Potential of Thermophilic Bioleaching, Effect of Temperature on the Process Performance

Karen H L Archer BSc (Eng) (Cape Town)

Submitted to the University of Cape Town in fulfillment of the requirements for the degree of Masters of Science in Engineering.

Bioprocess Engineering Research Group
Department of Chemical Engineering
University of Cape Town
Rondebosch, South Africa

September 1997

Acknowledgements

I would like to thank:

- *my supervisor Dr Susan Harrison for her support, advice and encouragement,*
- *Prof. Geoff Hansford for his open door and friendship*
- *Dr Mehdi Nemati for many hours of time and lunches on the roof*
- *MINTEK for their financial support*
- *Dr. Tony Pinches, John Neale, Marieckie Gericke and Dr Mike Dry from MINTEK*
- *Ritvar Muhlbauer, Dave Dew from Gencor*
- *Prof. Frank Robb from the University of Maryland Biotechnology Institute*
- *Prof. Peter Linder and Dr Martin Fey from the University of Cape Town*
- *Prof Tony Bryson and Dr Ellen Lawson from the University of Witwatersrand*
- *Sue Jobson and fellow members of the 'Bio's Group'*
- *Chemical Engineering staff members*
- *to my ex-digs members, I'm looking over my shoulder!*
- *my family and friends, thanks for all your prayers!*
- *Paul Archer, need I say more*
- *and God, with whom all things are possible.*

Abstract

Bioleaching is a biohydrometallurgical process whereby mineral sulphides are metabolically oxidised by microorganisms, releasing precious metals encapsulated in them. This pre-treatment is based on the action of microorganisms affecting oxidation of reduced sulphur species and ferrous iron to sulphate and ferric iron respectively. Conventionally *Thiobacillus ferrooxidans*, *Thiobacillus thiooxidans* and *Leptospirillum ferrooxidans* are implemented in this process in the region of 40-45°C and pH 1.8. A high temperature (65-80°C) process, utiltising thermophilic archaea such as *Sulfolobus* spp. can be considered as an alternative to current bioleaching practice. Literature indicates that there is an overall increase, 6 fold on average, in the rate of leaching due to the use thermophilic organisms.

Bioleaching involves nutrient transfer to microorganisms and interactions between several ionic species, including iron and sulphate. Thus, it is necessary to investigate the effect of the increased temperature on the gas-liquid mass transfer as well as ionic speciation of the system. Hence, the objectives of the present research were established as follows:

- to elucidate the effect of temperature on mass transfer from a theoretical point of view
- to establish whether ionic speciation is a contributing factor in thermophilic bioleaching
- to develop a generic and flexible means of representing ionic species

The three major factors which influence mass transfer are the mass transfer coefficient, k_L, interfacial area, a, and the driving force, C^*-C. In order to evaluate the effect of temperature on these factors, the variation of other physicochemical properties, including diffusivity, solubility, viscosity and density, with temperature were investigated. By considering several analytical and empirical mass transfer coefficient correlations, it was concluded that k_L is proportional to diffusivity to the power of $^1/_2$, $^2/_3$ or 1. Diffusivity increases with temperature, therefore, k_L increases with temperature. Excluding the effect of hydrodynamics, interfacial area (a) increases with temperature, based upon the contribution of surface tension. The solubility of oxygen and carbon dioxide decreases with temperature, thereby reducing the driving force. The relative proportions of these factors are such that the overall effect of temperature on mass transfer is mediated by the effect of temperature on k_L. Thus, the overall effect of an increase in temperature is improved mass transfer.

Ionic speciation is dependent upon the equilibrium constants of ionic reactions. Equilibrium constants are known to increase with temperature. Thus there is an increase in the extent of formation of complexes with increasing temperature, with a concomitant decrease in the available ferric iron. This has a negative effect on bioleaching and needs to be investigated more comprehensively. As a result, a modular simulation system which can take ionic speciation into account is proposed. This simulation should be flexible and generic such that further functionality can be included, as the algorithms become available. This is of value as it can be aimed directly and uniquely at bioleaching and ultimately include information about the kinetics of mesophilic and thermophilic operating conditions.

Current speciation packages are limited in their application as they cannot be amended to include information specific to bioleaching. As the first step towards a bioleaching simulation, a data model was developed to represent the thermodynamic and physical data of compounds involved in bioleaching as well as reactions including these compounds. This was achieved using two normalised SQL (Structured Query Language) databases. The data model was tested in a Delphi® 2.0 pascal simulation which calculates the equilibrium constants of given reactions. The simulation is able to accommodate reactions which include ionic species at temperatures up to 100°C using traditional thermodynamics and the Criss Cobble heat capacity correlation. In accordance with the criteria proposed, it is generic and flexible, such that further compounds, reactions and functionality can be included within the framework of the simulation.

Table of Contents

Appendices

Table of Figures

Table of Tables

Glossary

acetyl coenzyme A (acetyl-CoA) A combination of acetic acid and coenzyme A that is energy rich; it is produced by many catabolic pathways and is the substrate for the tricarboxylic acid cycle and other pathways.

acidophile A microorganism that has its growth optimum between about pH 1.0 and 5.5.

active transport The transport of solute molecules across a membrane against a concentration and/or electrical gradient; it requires a carrier protein and the input of energy.

adenosine 5'-triphosphate (ATP) The triphosphate of the nucleoside adenosine, which is a high energy molecule and serves as the cell's major form of energy currency.

adenosine diphosphate (ADP) The nucleoside diphosphate usually formed upon the breakdown of ATP when it provides energy.

aerobic respiration A metabolic process in which molecules, often organic, are oxidised with oxygen as the final electron acceptor.

anaerobic respiration An energy-yielding process in which the electron transport chain acceptor in an inorganic molecule other than oxygen.

archae, archaebacteria or archaeobacteria A kingdom of microorganisms that lack muramic acid in their cell walls, have membrane lipids with ether-linked branched chain fatty acids, and differ in many other ways from eubacteria.

Archaeobiota The name given to the archaeal kingdom.

arsenopyrite A sulphide mineral of composition FeAsS.

autotroph An organism that uses carbon dioxide as its sole or principal source of carbon.

biohydrometallurgy That branch of biotechnology dealing with the study and application of the economic potential of the interactions between the microbial world and the mineral kingdom.

biooxidation/bioleaching A hydrometallurgical process that involves the dissolution and recovery of metal values from minerals using microbial catalysts.

biotechnology The rational exploitation of the activities of living cells or part thereof.

bornite A sulphide mineral of composition Cu_5FeS_4.

caldariellaquinone A protein found in several archae. It has been used to distinguish species.

calvin cycle The main pathway for the fixation (or reduction and incorporation) of carbon dioxide into organic material during photosynthesis, it is also found in chemolithoautotrophs.

carotenoids Pigment molecules, usually yellowish in colour, that are often used to aid chlorophyll in trapping light energy during photosynthesis.

catabolism That part of metabolism in which larger, more complex molecules are broken down into smaller, simpler molecules with the release of energy.

cell envelope All the cellular structures outside the plasma membrane.

cell wall The strong layer or structure that lies outside the plasma membrane; it supports and protects the membrane and gives the cell shape.

chalcocite A sulphide mineral of composition Cu_2S.

chalcopyrite A sulphide mineral of composition $CuFeS_2$.

chemical leaching The action of a lixiviant without a contribution from microorganisms.

chemiosmotic hypothesis The hypothesis that a proton gradient and an electrochemical gradient are generated by electron transport and then used to drive ATP synthesis by oxidative phosphorylation.

chemoautotrophs, chemolitrophs, chemolithoautotrophs or chemolithotrophic autotrophs Microorganisms that oxidise reduced inorganic compounds to derive both energy and electrons; carbon dioxide is their carbon source.

chemoheterotrophs, chemoorganotrophs or chemoorganotrophic heterotrophs Organisms that use organic compounds as sources of energy, hydrogen, electrons and carbon for biosynthesis.

chemotrophs Organisms that obtain energy from the oxidation of chemical compounds.

coccus A roughly spherical microorganism.

coenzyme A loosely bound cofactor that often dissociates from the enzyme active site after product has been formed.

cofactor The nonprotein component of an enzyme, it is required for catalytic activity.

conformational change hypothesis The hypothesis that the energy from electron transport is used to induce changes in the shape of the enzyme that synthesises ATP and thereby drives ATP synthesis.

covellite A sulphide mineral of composition CuS.

Criss Cobble correlation An empirical correlation to estimate the heat capacity of ionic species. It is used in the prediction of thermodynamic properties of these ionic species at different temperatures.

cyanobacteria A large group of photosynthetic bacteria with a photosynthetic system like that present in eucaryotic photosynthetic organisms.

cytoplasm The protoplasm of a cell contained within the cell membrane, but excluding the nucleus.

cytosine A primidine 2-oxy-4-aminopyrimidine that is found in nucleosides, nucleotides, and nucleic acids.

database A collection of organised, inter-related data stored together without unnecessary redundancy.

Delphi® A pascal compiler developed by Borland™.

deoxyribonucleic acid (DNA) The nucleic acid that constitutes the genetic material of all cellular organisms. It is a polynucleotide composed of deoxyribonucleotides connected by phosphodiester bonds.

diffusion The net transport of material within a single phase in the absence of mechanical or convectional mixing.

diffusion coefficient, diffusivity A constant which quantifies the resistance afforded to a particular solute by a solvent.

driving force (C^*-C) The concentration gradient across the gas-liquid interface which drives mass transfer.

dump and heap leaching In the case of heap leaching, a heap of ore is designed and built to optimise bacterial oxidation reactions. Leach liquor and cell nutrients are percolated through this heap or a mine tailings dump to facilitate microbial ferrous iron and mineral sulphide oxidation

electron transport chain A series of electron carriers that operate together to transfer electrons from donors such as NADH and $FADH_2$ to acceptors such as oxygen.

electron transport chain A series of electron carriers that operate together to transfer electrons from donors such as NADH and $FADH_2$ to acceptors such as oxygen.

Embden-Meyerhof pathway or glycolytic pathway A pathway that degrades glucose to pyruvate; the six - carbon stage converts glucose to fructose 1,6-bisphosphate, and the three-carbon stage produces ATP while changing glyceraldehyde 3-phosphate to pyruvate.

Entner-Doudorff pathway A pathway that converts glucose to pyruvate and glyceraldehyde 3-phosphate by producing 6-phosphogluconate and then dehydrating it.

equilibrium The state of a system in which no net change is occurring and free energy is at a minimum; in a chemical reaction at equilibrium, the rates in the forward and reverse directions exactly balance each other.

equilibrium constant The constant that is characteristic of a chemical reaction at equilibrium and is based on the relative equilibrium concentrations or activities of products and reactants.

eubacteria The large majority of bacteria that have cell wall peptidoglycan containing muramic acid (or are related to such bacteria) and membrane lipids with ester-linked straight chained fatty acids.

eucaryotic cells Cells that have a membrane-delimited nucleus and differ in many other ways from procaryotic cells; protists, algae, fungi, plants, and animals are all eucaryotic.

extremophiles Microorganisms which inhabit or endure environments with some or all of the following characteristics: high salt, heavy metal concentrations, extremes of temperature, pressure or pH.

expert system A computer application which uses information databases and algorithms to foster artificial intelligence *e.g.* JESS (Joint Expert Speciation System).

facultative anaerobes Microorganisms that do not require oxygen for growth but do grow better in its presence.

fermentation An energy-yielding process in which organic molecules serve as both electron donors and acceptors.

flavin adenine dinucleotide (FAD) An electron carrying cofactor often involved in energy production, *e.g.* the tricarboxylic acid cycle.

fluid mosaic model The current accepted model of cell membranes in which the membrane is a lipid bilayer with integral proteins buried in the lipid and peripheral proteins more loosely attached to the membrane surface.

galena A sulphide mineral of composition PbS_2.

galvanic leaching A voltaic cell is produced with the current being produced chemically.

gas holdup The volume of gas contained within reactor liquid at any time, represented by ε.

gas-liquid interface The interface between the gas being transferred and the bulk liquid. This is the area of greatest resistance and hence determines the overall rate of mass transfer into the system.

GC content The percentage of guanine and cytosine in the cell's DNA. It is used to characterise species.

geothermal Relating to the heat in the interior of the earth.

glycoprotein A group of conjugated proteins containing small amounts of carbohydrates as prosthetic groups.

goethite A type of iron precipitate of the form (α-FeO·OH).

Gram stain A differential staining procedure that divides bacteria into gram-positive and gram-negative groups based on their ability to retain crystal violet when decolorised with an organic solvent such as ethanol. Conventionally a gram-positive stain implies that murein or, in the case of archae, a pseudomurein substance exists. Cells that show negative-staining cells do not possess this thick outer layer.

guanine A purine derivative, 2-amino-6-oxypurine, found in nucleosides, nucleotides and nucleic acids.

GUI Graphical user interface, *e.g.* Windows 3.1® and similar menu driven environments.

halophile A microorganism that requires high levels of sodium chloride for growth.

hard coded When specific information (*e.g.* functions or correlation coefficients) has been programmed directly into an application without giving the user access to change these parameters.

hematite An iron precipitate of the form (α-Fe_2O_3).

Henry's Law constant A constant, H, which characterises the solubility of a gas in a liquid according to $H=Px$ where P is the total pressure and x is the mole fraction of the gas in the liquid.

heterotroph An organism that uses reduced, performed organic molecules as its principal carbon source.

hydrodynamics The study of the mechanical properties of fluids.

hydrophilic A polar substance that has a strong affinity for water (or is readily soluble in water).

hydrophobic A nonpolar substance lacking affinity for water (or which is not readily soluble in water).

in situ leaching literally in place leaching.

inoculum The microbial population introduced into a nutrient solution.

interfacial area (a) A measure of the surface area of gas exposed to the bulk liquid per unit liquid volume.

jarosites Ferric iron hydroxide precipitates with the general formula of $XFe_3(SO_4)_2(OH)_6$, where X is an ion with a single positive charge.

lag phase A period following the introduction of microorganisms into fresh culture medium when there is no increase in cell numbers or mass. Other phases of growth are the exponential phase, stationary phase and the death phase.

leaching The process of dissolution of the value mineral(s) of an ore or concentrate, generally by an aqueous solution of the leaching agent. When microorganisms facilitate leaching, it is referred to as bioleaching.

lipoprotein A molecule containing both lipid and protein.

lithotroph An organism that uses reduced inorganic compounds as its electron source.

lixiviant leaching agent

mass transfer (gas-liquid) Mass transfer is governed by the quotient of the resistance and the driving force. This is represented according to $r_A = k_L a(C^*-C)$ where k_L is the transfer coefficient, a is the interfacial area and (C^*-C) is the driving force.

mesophiles Microorganisms which prefer an environment with a mild temperature, in the region of 30°C. Thermotolerant mesophiles exist which can endure temperatures in the vicinity of 45°C.

metabolism The total of all chemical reactions in the cell; almost all are enzyme catalysed.

metalliferous containing a metallic element

methanogenic bacteria Strictly anaerobic bacteria that derive energy by converting CO_2, H_2, formate, acetate and other compounds to either methane or methane and CO_2; they are archaebacteria.

microenvironment The environment in the immediate vicinity of the microorganism.

mitochondrion The eucaryotic organelle that is the site of electron transport, oxidative phosphorylation and pathways such as the Krebs cycle; it provides most of a nonphotosynthetic cell's energy under aerobic conditions. It is constructed of an outer and inner membrane, which contains the electron transport chain.

mixotrophic Organisms which combine autotrophic and heterotrophic metabolic processes.

morphology concerned with the form and structure of organisms

nicotinamide adenine dinucleotide (NAD^+) An electron-carrying coenzyme; it is particularly important in catabolic processes and usually donates its electrons to the electron transport chain under aerobic conditions.

nicotinamide adenine dinucleotide phosphate ($NADP^+$) An electron-carrying coenzyme that most often participates as an electron carrier in biosynthetic metabolism.

object oriented programming (OOP) A method of programming where data is encapsulated with the procedures that manipulate the data.

obligate aerobes Organisms that grow only in the presence of air or oxygen.

obligate anaerobes Microrganisms that cannot tolerate the presence of oxygen and die when exposed to it.

organotrophs Organisms that use reduced organic compounds as their electron source.

oxidative phosphorylation The synthesis of ATP from ADP using energy made available during electron transport.

oxidising agent or oxidant The electron acceptor in an oxidation-reduction reaction.

pentose-phosphate pathway The pathway that oxidises glucose 6-phosphate to ribulose 5-phoshate and then converts it to a variety of three to seven carbon sugars; it forms several important products (NADPH for biosynthesis, pentoses and other sugars) and can also be used to degrade glucose to carbon dioxide.

peptidoglycan A large polymer composed of long chains of alternating N-acetylglucosamine and N-acetylmuramic acid residues. The polysaccharide chains are linked to each other through connections between tetrapeptide chains attached to the N-acetylmuramic acids. It provides much of the strength and rigidity possessed by bacterial cell walls.

phenotypes Of an individual organism: the totality of observable structural and functional characteristics, or particular named characteristic(s) and/or properties; the phenotype of an individual is determined jointly by its genotype and environment.

phospholipid bilayer see fluid mosaic model

photolithotrophic autotrophs Organisms that use light energy, an inorganic electron source (*e.g.* H_2O, H_2, H_2S) and carbon dioxide as a carbon source.

photoorganotrophic heterotrophs Microorganisms that use light energy and organic electron donors and also employ simple organic molecules rather than carbon dioxide as their carbon source.

phototrophs Organisms that use light as their energy source.

molecular chronometers sequences of ribosomal RNA referred to as 5S, 16S or 23S rRNA

phylogenetic tree A diagrammatic representation of the development of species which indicates their interrelationships. Phylogenetic trees are constructed on the bases of amino acids in selected proteins, such as cytochrome C.

physiology concerned with the functioning of organisms

plankton Free-floating microorganisms that can be found in almost all waters.

plasma membrane The selectively permeable membrane surrounding the cell's cytoplasm; also called the cell membrane, plasmalemma or cytoplasmic membrane.

preg robbing A characteristic of ores whereby organic compounds interfere with gold recovery via conventional cyanidation processes.

pressure oxidation This involves the breakdown of sulphide minerals using steam and oxygen injection under pressure, exposing the gold for cyanide extraction.

procaryotic cells Cells that lack a true, membrane-enclosed nucleus; bacteria are procaryotic and have their genetic material located in a nucleoid.

pseudomurein A modified peptidoglycan lacking D-amino acids and containing N-acetylalosaminuronic acid instead of N-acetylmuramic acid; found in archae.

pulp density The mass of solids in grams, usually ore, per 100 ml of liquid, given as a percent

pyrite A sulphide mineral of composition FeS_2

pyrrhotite A sulphide mineral of composition $FeS_{(1-x)}$

refractory Traditionally confined to gold ores but recently extended to other ores such as chalcopyrite. Ores are considered refractory if the recovery of values (*e.g.* gold or copper) from conventional processes in the absence of a pretreatment step is less than 80%.

relationship modelling A strategy in designing databases to model the intrinsic relationships between data.

respiration An energy-yielding process in which an electron donor is oxidised using an inorganic electron acceptor. The acceptor may be either oxygen (aerobic respiration) or another inorganic acceptor (anaerobic respiration).

ribonucleic acid (RNA) A polynucleotide composed of ribonucleotides joined by phosphodiester bridges (Prescott *et al.*, 1993). Including ribosomal (rRNA), messenger (mRNA) and transfer (tRNA).

roasting A pretreatment step which removes harmful constituents from the ore by oxidation and vaporisation exposing gold for removal by cyanidation.

serotyping A technique or serological procedure that is used to differentiate strains (serovars or serotypes) of microorganisms that have differences in antigenic composition of a structure or product.

solfatara Hot acid soils, named after a region in Naples.

sphalerite A sulphide mineral of composition ZnS.

Structured Query Language (SQL) A command set used to automate database manipulation.

substrate The substance an enzyme acts upon.

surface layer A rigid, yet porous layer of glycoproteins constituting the cell wall of some archaebacteria.

surface mobility Surface mobility characterises the regeneration of the surface of the bubble. A bubble has two extremes of interfacial movement, no internal circulation within the particle and fully developed internal circulation.

surface tension A property of liquids caused by inter-molecular forces near the surface leading to the apparent presence of a surface film and to capillary.

taxonomy The science of biological classification; it consists of three parts: classification, nomenclature and identification. Species are classified into a kingdom, order, family and genus.

thermoacidophiles A group of microorganisms that grow best at acidic pHs and high temperatures.

thermophiles Microorganisms which tolerate temperatures above 40°C, further classification includes moderate thremophiles which have an optimum growth temperature in the region of 50°C, extreme thermophiles around 70°C and hyper thermophiles with an optimum growth temperature greater than 80°C.

transfer coefficient (k_L) A measure of the resistance offered to gas-liquid mass transfer by the bulk liquid.

tricarboxylic acid (TCA) cycle The cycle that oxidises acetyl coenzyme A to carbon dioxide and generates NADH and $FADH_2$ for oxidation in the electron transport chain; the cycle also supplies carbon skeletons for biosynthesis.

triose three carbon sugar

Introduction

1 Introduction

Bioleaching of sulphide minerals is becoming an established process for the extraction of copper and the pre-treatment of arsenical refractory ores. It is an environmentally friendly alternative to roasting and high-pressure leaching. Advantages of bioleaching plants include the absence of noxious off-gases or toxic effluents, simplicity of plant operation and maintenance, safety considerations, tolerance to a feed of a wide range of sulphur grade, and the production of a stable iron/arsenic residue. Biooxidation processes are, however, sensitive to water quality, particularly with respect to cyanide and thiocyanate. They may require sizeable power and neutralisation costs. Bioleaching plants require long residence times, of the order of days. This increases operational costs. Current commercial processes employ *Thiobacillus ferrooxidans* and *Leptospirillum ferrooxidans*, operating in the region of 10 to 45°C and a pH value of 1.8.

A major contributor to bioleaching is the chemical leaching action of ferric iron on the mineral sulphides. It is postulated that by increasing the operating temperature the rate of this chemical oxidation will increase, thereby increasing the overall rate of bioleaching. Hence, there is growing interest in operating at a higher temperature to take advantage of anticipated higher leach rates and reduced cooling costs. High temperature bioleaching may be achieved by implementing moderate or extreme thermophiles, including *Sulfolobus* spp. The consequences of temperature on mass transfer and ionic speciation need to be evaluated in order to assess the advantages and disadvantages of elevated operating temperatures.

The oxidation of mineral sulphides is an exothermic process, requiring extensive cooling to remove excess heat which adversely affects the activity of microorganisms if left uncontrolled. A study was conducted by Miller (1991) which indicated that the optimum operating temperature for bioleaching, in terms of heat removal, is 65°C. Another advantage of operating at a higher temperature is the increased driving force for cooling.

The role of the microorganisms is to re-oxidise the ferrous iron to ferric iron and maintain a high redox potential. Another role of the microorganisms is to oxidise elemental sulphur. Elemental sulphur formed or remaining after the pre-treatment is detrimental in the cyanide leach step as it produces thiocyanate ions, thereby increasing cyanide consumption.

Extreme and moderate thermophiles which oxidise iron, including *Sulfolobus* BC and *Acidianus brierleyi* have an optimum growth temperature between 50 and 80°C and pH between 1.3 and 2.0. Substantial work with the thermophiles has been conducted to investigate their potential for bioleaching. This indicates that an increase in leach rate can be expected. An increase in the extent of leaching by *Sulfolobus*-type organisms of copper and other base metal sulphide minerals is anticipated. Observations show that shear damage from high solids loading or toxicity from high metal concentrations may adversely affect the growth of these organisms.

The aim of this work was to look at the effect of temperature on the chemical environment of bioleaching systems. Physico-chemical factors which vary with temperature include gas solubility, diffusivity, viscosity and

density. These factors impinge upon mass transfer in the reactor. In bioleaching, important mass transfer steps include the transfer of nutrients, including oxygen, carbon dioxide and ferrous iron to the cells as well as the transfer of ferric iron to the mineral. Oxygen is required for the chemical oxidation of ferrous iron, and both carbon dioxide and oxygen are required for the respiration of extreme thermophiles. Mass transfer is influenced by the transfer coefficient (k_L), interfacial area (a) and the driving force (C^*-C). These three factors are dealt with in turn from a theoretical point of view.

In order to optimise and simulate bioleaching, thereby increasing our overall understanding of bioleaching, it would be useful to have a modular simulation. Modularity would ensure that modules concerning thermodynamics, kinetics, mass transfer and cell growth can be added as they become available. Because bioleaching involves interactions between several ionic species, including iron and sulphate, it is necessary to consider the effect of the increased temperature on iron speciation. As a first step in achieving this goal, a data model has been developed to ensure that thermodynamic and physical data of compounds can be represented generically.

This data model is implemented in a Delphi® 2.0 simulation which makes use of two relational, normalised databases. The stoichiometric information and the thermodynamic data of the components involved are stored in the 'species' database. The 'running' database enables the selection of several reactions and stores calculation variables. This data model has been tested with the calculation of equilibrium constants for generic reactions. Heat capacity data is predicted using a standard power series for non-aqueous species and by the Criss Cobble correlation for aqueous species.

This thesis is divided into three parts. **Part I** (Chapter 2) looks at the pretreatment of mineral sulphides, with particular emphasis on bioleaching and the commercial plants in operation. Other pretreatment alternatives, the mechanism of bioleaching, chemical leaching, jarosite formation and dump and heap leaching are discussed.

Part II (Chapters 3 & 4) focuses on the feasibility of thermophilic bioleaching. Do microorganisms exist which are capable of leaching mineral sulphides at temperatures above 60°C? Further are these microorganisms able to tolerate the low pH, high solids loading, and can they tolerate a similar level of metal concentrations to *Thiobacillus* spp. Of greatest importance is whether the recovery of gold and metal values by thermophilic bioleaching is economically efficient.

Part III (Chapters 5 - 10) concentrate on the effect of temperature on the bioleaching environment. Two main topics are covered, mass transfer and a data model to represent chemical compounds as a precursor for speciation calculations.

In Chapter 5, mass transfer correlations and the transfer coefficient, interfacial area and driving force are explained. Chapter 6 covers the theoretical investigation of the effect of temperature on the physico-chemical factors which in turn affect the transfer coefficient, interfacial area and driving force. The overall effect of temperature on mass transfer is contrasted with experimental results reported in literature.

Chapter 7 describes the thermodynamics of aqueous solutions and the Criss Cobble correlation. This forms the basis for the consideration of speciation in the bioleaching system (Chapter 8) and the data model which is implemented to calculate equilibrium concentrations of any given chemical reaction (Chapter 9).

Chapter 10 details the conclusions of Part III and includes recommendations for future work. This is followed by the appendices.

1.1 REFERENCES

Miller, D.M. (1991): "Effect of temperature on BIOX® operations", Colloquim, Bacterial Oxidation, South African Institute of Mining and Metallurgy, Johannesburg, 18 June

Part I

Pretreatment of Mineral Sulphides, Including Bioleaching

2. Pretreatment of Mineral Sulphides

Sulphide-refractory gold ores contain gold in intimate association with sulphide minerals, typically pyrite and arsenopyrite. Steps in gold recovery include milling, concentration and a pretreatment step to liberate occluded gold (Figure 2-1). The objective of the pretreatment step is to destroy the host matrix in order to render the gold amenable to contact with cyanide ions and/or to decompose species which combine with and deplete the cyanide and oxygen to dissolve the gold (Iglesias and Carranza, 1996).

Metal extraction from refractory sulphide minerals can be achieved by roasting, pressure oxidation and biooxidation (Poulin and Lawrence, 1996). Biooxidation has been considered for sulphide minerals other than those associated with gold recovery including zinc, copper, nickel, antimony, lead and tin (Barrett *et al.*, 1993; Karavaiko *et al.*, 1977). Metal toxicity and the availability of iron, ferrous iron as an energy substrate and ferric iron as an oxidising agent, need to be considered.

Biotechnology can be described as the rational exploitation of the activities of living cells or part thereof (Poulin and Lawrence, 1996). Biohydrometallurgy may be defined as that branch of biotechnology dealing with the study and application of the economic potential of the interactions between the microbial world and the mineral kingdom (Rossi, 1990). An increase in environmental awareness and the need for low cost treatment have led to serious investigation of biotechnology for environmental control (Poulin and Lawrence, 1996).

2.1 REFRACTORY ORES

Gold ores are considered refractory if gold extraction from a conventional cyanidation process in the absence of a pretreatment step is less than 80%, even after fine grinding. This low recovery eliminates economic recovery of the metal (Iglesias *et al.*, 1993). Physical or primary refractoriness is due to the inability of cyanide ions to

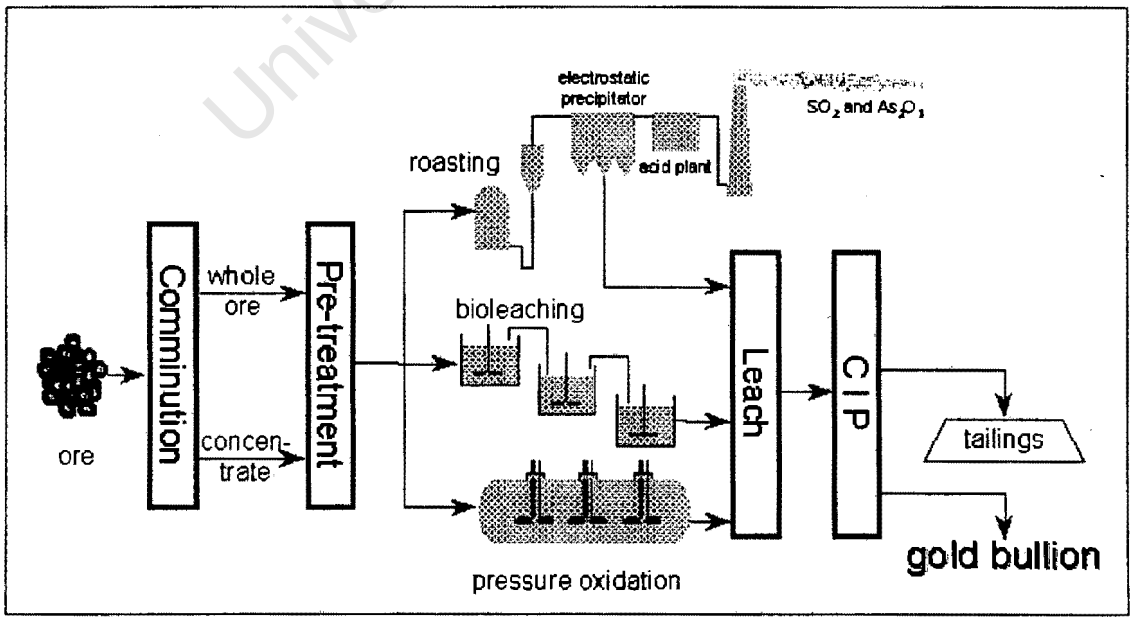


Figure 2-1: Schematic flow diagram of sulphide mineral pretreatment in gold production (Mintek, 1996).

access the gold particles inside the sulphide crystal lattice. As a result ore is milled to reduce this obstacle. When the refractoriness is mainly due to the reactive behaviour of the matrix, it is known as chemical refractoriness, *e.g.* mineral sulphides. Sulphide minerals decompose to sulphur or sulphate species. Sulphur combines with and depletes the cyanide available for gold dissolution. In addition intermediate species like ferrous iron, the sulphide ion, thiosulfate and arsenite can consume oxygen vital for gold dissolution in cyanide. Furthermore, these species tend to precipitate the gold already dissolved (Iglesias *et al.*, 1993).

Carbonaceous gold ores represent a unique class of refractory precious metal ores. These ores contain organic matter that interferes with gold recovery by cyanidation. Direct interference can be attributed to either occlusion of gold within organic matter or formation of a stable gold-carbon complex. A more common problem is indirect interference, whereby the aurocyanide complex formed during cyanidation is sorbed by the native organic matter and is not available for recovery. This phenomenon is referred to as "preg robbing" (Hutchins *et al.*, 1988).

2.2 ROASTING

Roasting is currently the most commonly used pre-treatment process (Bailey and Hansford, 1993). It removes harmful constituents by oxidation or vaporisation, and liberates gold from pyrite. In some cases, however this method of pre-oxidation can result in poorer metal recovery, particularly when compounds, which consume cyanide in the subsequent metal recovery stage are formed (Lawrence and Bruynesteyn, 1983).

Roasting is capital and operating cost intensive and has negative environmental implications (Iglesias *et al.*, 1993). It is applicable to ores containing organic carbon and is able to break down the preg-robbing characteristic of the ore. If arsenic is present, a two-stage roaster is often required to remove arsenic as arsenic trioxide and then degrade the remaining sulphide. Gas scrubbers are essential to capture sulphur dioxide and arsenic trioxide emissions. Two-stage roasters and emissions control devices greatly increase capital costs. If all costs relative to emissions are internalised by the company, operating costs are also increased (Poulin and Lawrence, 1996).

2.3 PRESSURE LEACHING

Autoclave processing involves the breakdown of sulphide minerals using steam and oxygen injection under pressure, exposing the gold for cyanide extraction. This is combined with chemical leaching as the activity of oxygen is increased at higher pressures, increasing its potential as a chemical leaching agent. While pressure autoclave treatment typically results in good recoveries, it is not appropriate when carbonaceous material is present. Pressure autoclaves are capital intensive because of their advanced materials of construction and the need for an oxygen plant. The high level of operator training and necessary skill, as well as increased safety requirements to handle the high pressures and temperatures, increase the operating costs of this process (Poulin and Lawrence, 1996).

2.4 BIOLEACHING

In bioleaching, sulphide mineral leaching is catalysed by microorganisms, typically *Thiobacillus ferrooxidans, Leptospirillum ferrooxidans* or extreme thermophiles (including sulfolobus-type organisms). This catalysis can be in the form of either direct or indirect leaching (Section 2.5). It can be applied to the treatment of low-grade ores as in heap leaching operations, or of higher-grade ores in stirred tank reactors (Lawrence and Bruynesteyn, 1983).

During the bioleaching of pyrite and arsenopyrite arsenic and sulphur are solubilised. The arsenic and sulphur contained in the effluent are precipitated as basic ferric arsenate ($FeAsO_4$) ($FeOH_3$) and calcium sulphate. Disposal of the arsenic is dependent on the stability of this precipitate. Tests show that when the Fe^{3+}:As^{5+} ratio exceeds 3:1, the precipitate will be compliant with current Environment Protection Agency (EPA) standards (van Aswegen, 1993; Brierley and Brans, 1994).

Biooxidation plants have advantages and disadvantages that influence their economic attractiveness. Advantages include the absence of noxious off-gases or toxic effluents, simplicity of plant operation, maintenance, safety considerations, tolerance to a wide range of sulphur grade feed, and the production of a stable iron/arsenic residue (Haines and van Aswegen, 1990). However, the bioxidation processes are sensitive to water quality, particularly with respect to cyanide and thiocyanate. They can entail sizeable power and neutralisation costs (Poulin and Lawrence, 1996). Biooxidation plants require long residence times, of the order of days which causes excessive operational costs. Thus, it is desirable to improve the kinetics of biooxidation (Iglesias and Carranza, 1996). The principle benefits of bioleaching are low operating costs and mitigation of air pollution. The lowering of operating costs allows the processing of lower grade ores and the scavenging of residual metal values from mine wastes (Brierley and Brierley, 1986, van Aswegen, 1993).

2.5 MECHANISMS OF THE LEACHING PROCESS

The role of microorganisms in leaching is complex and not precisely defined. Some researchers support the concept that the function of microbes may be 'indirect', whereby the microbes generate ferric iron, which oxidises the mineral. Other investigations indicate 'direct' leaching in which the microbes contact or adhere to the mineral surface, directly oxidising the mineral (Brierley and Brierley, 1986). The extent to which chemical leaching (oxidation without the presence of bacteria) can take place depends on the mineral. It is uncertain at which points each type of leaching dominates. The extent to which each mechanism occurs is dependent on various parameters such as temperature, pH, redox potential and solids concentration.

2.5.1 DIRECT BIOLEACHING

Direct bacterial leaching proceeds through the interaction of bacteria with the surface of the sulphide mineral ore, or any other sulphur depositions on the surface (Espejo *et.al.*; 1988). Pyrite may be oxidised directly by the microbes:

$$FeS_2 + 3\frac{1}{2}O_2 + H_2O \rightarrow FeSO_4 + H_2SO_4$$

This results in the solubilisation of iron. The iron is subsequently oxidised back to ferric iron by microbes, which then participates in the 'indirect' leaching process (Brierley and Brierley, 1986). The bacteria cause the release of the metal ions from the actual solid making use of carbon dioxide and oxygen and releasing sulphate ions, as illustrated in Figure 2-2 (Boon and Heijnen, 1993).

2.5.2 INDIRECT OR 'TWO-STEP' BIOLEACHING

The fundamental reaction for indirect leaching is the microbial oxidation of ferrous iron in acidic conditions for the purpose of energy generation:

$$4FeSO_4 + O_2 + 2H_2SO_4 \rightarrow 2Fe_2(SO_4)_3 + 2H_2O$$

The ferric sulphate generated serves to oxidise minerals such as chalcopyrite, chalcocite and covellite (Brierley and Brierley, 1986):

$$CuFeS_2 + 2Fe_2(SO_4)_3 \rightarrow CuSO_4 + 5FeSO_4 + 2S$$

$$Cu_2S + 2Fe_2(SO_4)_3 \rightarrow 2CuSO_4 + 4FeSO_4 + S$$

$$CuS + Fe_2(SO_4)_3 \rightarrow CuSO_4 + 2FeSO_4 + S$$

The reduced iron is then re-oxidised by the microorganisms to ferric iron making it available as the oxidising agent and continuing the cycle (Brierley and Brierley, 1986). Indirect leaching entails the oxidation of the metal sulphides by the ferric iron regenerated by the microbes. Hence ferric iron takes the role of a chemical-leaching agent, as illustrated in Figure 2-3. Another role of the microorganisms is to oxidise elemental sulphur (Boon *et al.*, 1995). Elemental sulphur formed or remaining after the pretreatment is undesirable in the cyanide leach step as it produces thiocyanate ions, thereby increasing cyanide consumption (Iglesias and Carranza, 1996).

Escobar *et al.* (1993) determined that the main action of microorganisms in bioleaching is to maintain a high redox potential, and thus a high ferric iron concentration in order to help the dissolution of other minerals present. This is in agreement with the 'Two-Step' hypothesis (Boon *et al.*, 1995), which postulates several

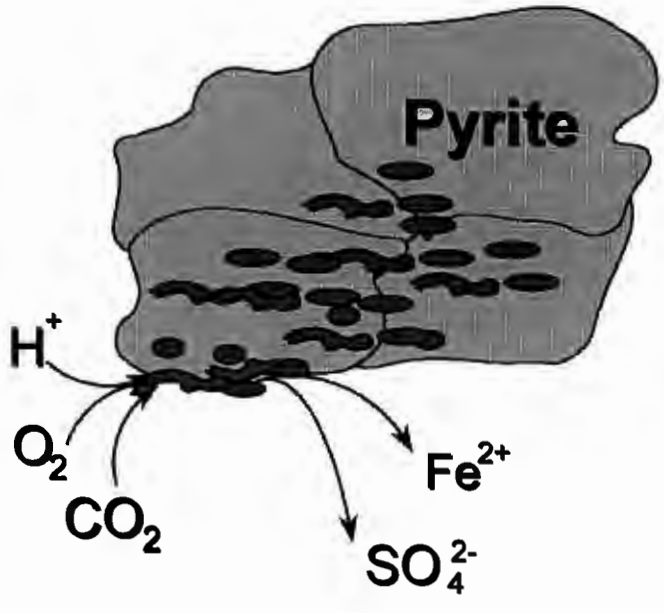


Figure 2-2: Direct bioleaching mechanism (Boon and Heijnen, 1993).

contributing sub-processes:

1. The major contribution to leaching is achieved by chemical leaching of sulphide minerals by ferric iron (Boon *et al.*, 1995), the case for pyrite is given below. It must be noted that Dutrizac and MacDonald (1974) and Konishi *et al.* (1990, 1995) state that chemical leaching of pyrite by ferric iron does occur to a great extent. A contributing factor is the unfavourable stoichiometry of the reaction as indicated below. Furthermore, the effectiveness of a powerful oxidising agent varies for different mineral sulphides (May, 1997).

$$FeS_2 + 14Fe^{3+} + 8H_2O \rightarrow 15Fe^{2+} + 2SO_4^{2-} + 16H^+$$

2. Microorganisms regenerate ferric iron. This reaction is also controlled by thermodynamic equilibrium (Boon *et al.*, 1995):

$$4Fe^{2+} + O_2 + 4H^+ \rightarrow 4Fe^{3+} + 2H_2O$$

3. Sulphur consumption by sulphur-metabolisers including *Thiobacillus thiooxidans* and *Sulfolobus acidocaldarius* reduces sulphur accumulation (Rossi, 1990).
4. Galvanic interactions occur between minerals of different rest potentials (Natarajan, 1988).

The production of sulphuric acid, or alternatively maintenance of the acidic medium is vital for the continuation of the leaching process. This maintains a pH which is optimum for the bacteria, as well as preventing the precipitation of jarosite, a complex ferric hydroxide (Lizama and Suzuki; 1988).

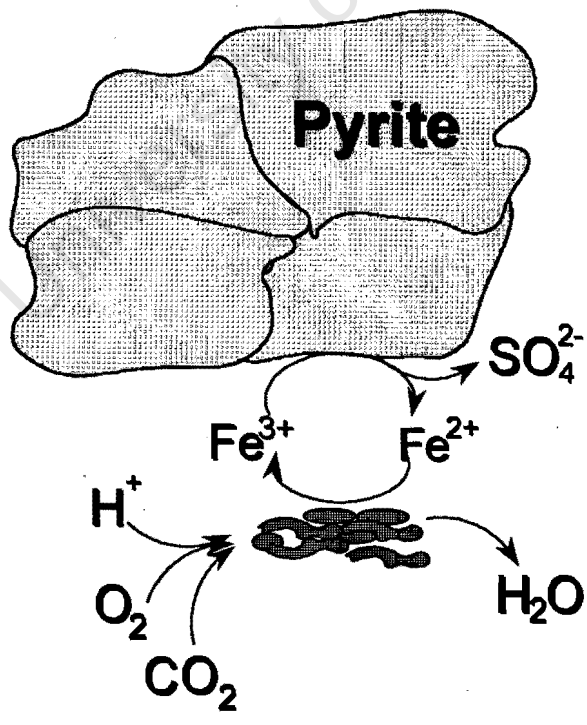


Figure 2-3: Indirect bioleaching mechanism (Boon and Heijnen, 1993).

2.5.3 CHEMICAL LEACHING

The 'Two-Step' hypothesis indicates that chemical oxidation of the mineral sulphide is the primary attack on the mineral. Hence it is important to understand what affects the rate of chemical oxidation of mineral sulphides by ferric iron. The reaction for pyrite is as follows (Zheng *et al.*, 1986):

$$FeS_2 + 14\,Fe^{3+} + 8\,H_2O \rightarrow 15\,Fe^{2+} + 2SO_4^{2-} + 16H^+$$

The rate of most chemical reactions increases exponentially with temperature in accordance with the Arrhenius equation. Further, Zheng *et al.* (1986) demonstrate that the rate of the above equation is a function of ferric iron concentration, the ratio of ferric to ferrous iron and the sulphate ion concentration.

Boogerd *et al.* (1991) assessed the relative contributions of biological and chemical leaching to the overall rate of pyrite oxidation at 30, 45 and 70 °C in shake flasks. At these temperatures they quantified the chemical oxidation of pyrite as 2, 8-17 and 43% respectively. Vitaya *et al.* (1994) found the contribution of chemical leaching to be 33% at 70°C. Hence the extent of ferric iron leaching of sulphide minerals increases with temperature. Iglesias and Carranza (1995) confirm this in the case of chalcopyrite and bornite. Iglesias and Carranza (1996) also conducted work on ferric sulphate leaching of arsenopyrite. As the temperature increases beyond 50°C, the rate-limiting step changes from chemical control to diffusional control, indicating the increase in chemical leaching kinetics.

2.5.4 JAROSITE AND IRON PRECIPITATES

Another aspect, which is important to the mechanism of leaching, is the formation of jarosites (Brierley *et. al.*, 1980). Jarosites are ferric iron hydroxide precipitates with the general formula of $XFe_3(SO_4)_2(OH)_6$, where X is an ion with a single positive charge. Two other iron precipitates that form are goethite (α-FeO·OH) and hematite (α-Fe_2O_3) (Dutrizac, 1985). Owing to the importance of the dissolved, free ferric iron concentration, excessive removal of ferric iron from solution is undesirable. Jarosite formation is dependent on pH as ferric iron has a low solubility at a pH greater than 2.5 (Dutrizac, 1985; Ferreria, 1975). Kametani (1955) found that the rate of decrease of the concentration of residual ferric iron in solution $\left(-\frac{dC}{dt}\right)$ is a function of pH and ferric iron concentration (C), according to:

$$\log\left(-\frac{dC}{dt}\right) = A + 3\cdot \mathrm{pH} + 2\log C \qquad \textit{Equation 2-1}$$

where A is a constant between –9 and –12.

Of the physical factors that affect jarosite formation, temperature and retention time are the most significant. Increasing temperature significantly increases the extent of iron precipitation, especially in the range 70-110°C. This is supported by Escobar *et al.*, 1993; Gomez *et al.*, 1996; Norris and Barr, 1988. The amount of precipitate increases

with prolonged retention times, up to 15 hours. Longer periods are without major effect as a pseudo-equilibrium has been reached (Dutrizac, 1983). As iron is continually solubilised during bioleaching, this pseudo-equilibrium will not be maintained and jarosite will form in proportion to the available ferric iron.

2.6 MICRO-ORGANISMS EMPLOYED IN BIOLEACHING

Microorganisms involved in bioleaching can endure acidic and generally unfavourable conditions. In mining environments, particularly in mine waters, it is possible to find large quantities of heavy metals, low pH and, in some cases, high temperatures to which some microorganisms adapt very easily. The energy needed for their metabolic reactions is obtained from the oxidation of reduced compounds of sulphur and ferrous iron (Torres *et al.*, 1995).

Conventionally mesophilic organisms such as *Thiobacillus ferrooxidans, T. thiooxidans* and *Leptospirillum ferrooxidans* are implemented at an operating temperature between 35 and 45°C and pH between 1.2 and 1.8. High temperatures have a negative effect on the activity of mesophilic organisms (Torres *et al.*, 1995). Several extreme and moderate thermophiles have been isolated which, like *Thiobacillus* spp. oxidise sulphur and iron in an acidic environment. Mesophiles, moderate and extreme thermophiles have an optimum growth temperature around 30, 50 and 70°C respectively (Norris, 1990).

Substantial work with the thermophiles, *e.g. Sulfolobus acidocaldarius* and *Acidianus brierleyi*, has been conducted to investigate their potential for bioleaching (Jordan *et al.*, 1996; Clark and Norris, 1996; Gómez *et.al.*, 1996). While the application of thermophiles in bioleaching has resulted in increased leaching rates (Konishi *et al.*, 1995; Norris and Barr, 1988) it is striking that in most cases a low pulp density, less than 10%, has been found to be the optimum. This shows that shear damage from high solids loading or toxicity from high metal concentrations may adversely affect the growth of these organisms (Torres *et. al.*, 1995; Escobar *et al.*, 1993; Jordan *et al.*, 1996; Clark and Norris, 1996).

To ensure successful implementation of thermophiles such as *Sulfolobus*-type microorganisms, the following aspects need to be demonstrated (Lawrence and Marchant, 1988):

- fast oxidation kinetics,
- operation at high pulp densities and
- tolerance to low pH and high metal concentration.

2.7 ENERGY AND ECONOMIC CONSIDERATIONS

Lawrence and Marchant (1988) conducted a comparative economic study on the bioleaching system operating at mesophilic and thermophilic temperatures. The primary benefits to be derived from operating at a higher temperature are reduced cooling costs and improved oxidation kinetics (Lawrence and Marchant, 1988).

Cooling considerations: Oxidation of sulphide minerals is an exothermic process, as such considerable cooling is required. The potential of operating at a higher temperature is the greater cooling gradient. If cooling water

is available at 10°C, then using *Sulfolobus* spp. at 70°C has a driving force twice as high as that of the conventional system operated at 40°C. As a result of this greater cooling gradient the efficiency of heat transfer is increased (Lawrence and Marchant; 1988, Brierley and Brans, 1994). Hence both the capital and operating costs will be reduced.

Oxidation kinetics: Improved oxidation kinetics reduces the residence time in bioleaching tanks. This would result in a lower capital and operating cost per unit product, assuming that the high temperature system can be maintained at comparable pulp densities (Lawrence and Marchant, 1988).

2.8 COMMERCIAL BIOLEACHING OPERATIONS AND PILOT PLANTS

Bioleaching is now a fully-fledged technology. Three commercial processes are available; Gencor's BIOX® process, MINTEK-AAC's MINBAC process and BacTech (Australia) Pty Ltd (Brierley and Brans, 1994). These processes have been implemented in several full-scale and pilot biooxidation plants.

Others that have reported varying degrees of success in the bacterial treatment of refractory gold ores include the Equity Silver Mine, British Columbia, Durea de Recherches Géologiques et Miniéres (BRGM) pilot plant, France and Transworld Mining and Minerals (TWMM), Australia. Several other gold ores in British Columbia have also been evaluated in pilot scale, *e.g.* Salmita Mine) (Lindström *et al.*, 1992).

2.8.1 BIOX® AND BIONIC® (GENCOR)

Gencor pioneered the BIOX® process for the biological oxidation of sulphide gold deposits. This bioleaching technology has been licensed to other producers, namely Ashton Mining Harbour Lights in Western Australia (1991), ASARCO's Wiluna plant in 1993, and ASHANTI Goldfield's Sansu Mine in 1994. At 1 Mt/y, Ashanti is the largest BIOX® installation (Warhurst and Bridge, 1996). Gencor is also involved in the operation of a 150-tpd biooxidation plant at Sao Bento in Brazil (Bailey and Hansford, 1993).

In 1994 Gencor entered into a joint venture with an Australian Producer, Maggie Hays Nickel, in which the BioNIC® process will be implemented. This process enables the treatment of nickel sulphides where the head grade is too low to allow sufficient concentration (Warhurst and Bridge, 1996).

2.8.1.i Fairview

At Fairview mine, near Barberton in the Eastern Transvaal, South Africa, arsenopyrite/pyrite concentrate was roasted in an Edwards roaster. A BIOX® demonstration plant was commissioned at Fairview mine to treat 40% of the concentrate. The full-scale plant treats 35 tpd and was commissioned in July,1991 (van Aswegen, 1993).

2.8.1.ii Sao Bento Mineração Mine

The Sao Bento Mineração Mine is located 100km northwest of Belo Horizante, Brazil. It was commisioned in 1987 to produce 185 kg/month of gold from 240 tpd of refractory sulphide flotation concentrate (Slabbert *et al.*, 1992). This operation uses a pressure oxidation circuit based on technology supplied by Sherritt Gordon. A biooxidation module was proposed to act as a pretreatment to the autoclave to reduce the pyrrhotite and siderite. These compounds are undesirable in the process as pyrrhotite results in the formation of elemental sulphur and siderite decomposes in acidic media to give carbon dioxide, which reduces the oxygen partial pressure in the autoclave (Slabbert *et al.*, 1992).

A BIOX®plant was designed to increase combined BIOX®/Pressox refractory sulphide concentrate treatment to 480 tpd. This included the installation of 8 x 550m^3 BIOX® reactors with two 35 000 NCMH compressors and two 3.4 MW cooling towers (Slabbert *et al.*, 1992). The BIOX® plant was commissioned to treat 150 tpd by October 1991 (van Aswegen, 1993).

2.8.1.iii Ashton Mining Harbour Lights

This operation is situated near Leonora in Western Australia. A refractory sulphidic orebody is mined with the gold mainly associated with the arsenopyrite and pyrite (van Aswegen, 1993). A 40tpd concentrate treatment BIOX® plant was designed to form part of the existing oxide ore treatment plant. Continuous operation commenced on 1 February 1992, full production was achieved by June 1992 (van Aswegen, 1993). This plant is currently operating at 115 tonnes of concentrate per day and was successfully commisioned in 1993. The plant will be expanded in 1998 to treat 155 tpd (Dew, 1997).

2.8.1.iv Asarco Wiluna Mine

The BIOX® plant at Wiluna, Western Australia processes a refractory pyrite/arsenopyrite concentrate. A mixed culture of *Thiobacillus ferrooxidans, Thiobacillus thiooxidans* and *Leptospirillum ferrooxidans* is employed. The plant consists of six 450 m^3 reactors, three primary reactors in parallel followed by three secondary reactors in series. All reactors are equipped with an agitator, sparge ring and cooling coil baffles. The plant was designed to handle 115 tpd of concentrate and currently operates at around 50 tpd (Brown *et al.*, 1995).

2.8.1.v Ashanti Sansu BIOX® Plant

Ashanti Sansu is located at Ashanti Goldfields Company's Obuasi mine in Ghana. Ashanti's sulphide ore is carbonaceous and highly refractory with 5 – 40% of the gold recoverable by direct cyanidation. The gold is generally fine-grained and occluded in sulphide minerals, mainly arsenopyrite and pyrite (Nicholson *et al.*,

1994). The continuous BIOX® test work was conducted in a fully automated mini-plant with a live volume of 127.2 l (Nicholson *et al.*, 1994).

The plant design consists of three identical modules. Each module consists of three primary reactors in parallel and three secondary reactors in series. The eighteen BIOX® reactors are equidimensional with a live capacity of 900m^3 each. Each reactor is equipped with cooling coil baffles to maintain the slurry temperature at 40°C. A total of 106 000 NMCH of low-pressure air, supplied by five blowers, is injected into the reactors via sparge rings. Axial flow agitators ensure optimum dispersion of the air (Nicholson *et al.*, 1994). Construction was completed in January 1994, and successfully commissioned to treat 720 tpd one month later. A fourth module was constructed and commissioned in 1995 increasing plant capacity to 960 tpd (Dew, 1997).

Slurry pH is maintained in the range 1.2 to 1.8. The neutralisation circuit comprises six 300m^3 contactors in series providing a residence time of six hours. The neutralised effluent is combined with the flotation tailings and BIOX® CIL residues prior to deposition onto the tailings dam (Nicholson *et al.*, 1994).

2.8.2 MINBAC™ (MINTEK - AAC)

The MINBAC™ process has been developed over the past ten years. The process is actively promoted by three international mineral organisations MINTEK, Anglo American Corporation of South Africa and Bateman Project Holdings Ltd.

Anglo American Corporation, South Africa and MINTEK have designed and commissioned a bacterial oxidation pilot plant at Vaal Reefs, South Africa. This plant treats 20 tpd of a backfill pyrite concentrate which has a sulphur and gold grade of 33% and 5g/t respectively. The plant consists of two 125m^3 oxidation tanks and a one 25m^3 tank. A similar 1 tpd pilot plant is currently operating at MINTEK (Dempsey *et al.*, 1990; Bailey and Hansford, 1993).

2.8.3 BACTECH

The Youanmi Deeps Project, owned by Gold Mines Australia Limited, is an underground mine situated 500km northeast of Perth in Western Australia. The underground ore is refractory with gold locked in arsenopyrite. The mine has a life expectancy of eight years. Bioleaching is an attractive process option as the size of the known resources available preclude installation of relatively high capital processing facilities, such as pressure oxidation vessels or a roaster.

The biooxidation plant associated with this mine was scheduled for commissioning in late 1994. The process employs moderately thermophilic, iron oxidising bacteria at 45-47° with excursions to 50°C. These cells have a rod shaped morphology similar to *Thiobacillus*. However, they are slightly larger in size, (1.5-4.3µm in length) and they have an optimum growth temperature of 50°C (Brierley and Brans, 1994). This culture, designated "M4", was enriched from a mine in Western Australia (Barret *et al.*, 1988).

2.9 HEAP, DUMP AND *IN SITU* LEACHING

The process of dump and heap leaching was recognised and commercially employed in the 1700s to extract copper in Spain. This technology is often used for treating low-grade ores, once again making use of the reduced operating costs (Brierley and Brierley, 1986).

Heap leaching involves a heap of ore, which has been specifically designed and built to optimise bacterial oxidation reactions. The ore may undergo preliminary crushing and screening, or even partial roasting before the heap is laid (Rossi, 1990). The heap is constructed by laying the ore on containment sheets and plumbing the heap to percolate the leach liquor.

Dump leaching is similar to heap leaching except that a dump is a pile of material which has not normally been constructed with any purpose other than to store either mine waste material, including untreated low grade ore, or tailings from previous extraction processes. The method allows very low-grade material to be treated and, as in the case of heap leaching, it is successful for copper recovery. This ties in with production of acid mine drainage where research is currently being carried out with the aim of preventing such bacterial activity (Barrett *et al.*, 1993).

Sulphuric acid is applied to a heap or dump, which has been seeded with appropriate microorganisms. These dumps and heaps can become heated because of the exothermic oxidation of pyrite (Murr and Berry, 1979). Because the immense size of these operations prevents dissipation of the heat, the internal temperatures rise. These elevated temperatures may prohibit the growth and activity of mesophilic organisms. Several moderate or extreme thermophilic microorganisms have been isolated from leach dumps (Brierley, 1978; Brierley and Brierley, 1986). It has been reported that thermophilic organisms pay a role in metal solubilisation at these elevated temperatures (Brierley and Brierley, 1986; Brierley, 1977; Brierley *et al.*, 1980).

In situ leaching is a form of chemical solution mining where a specific lixiviant is utilised to dissolve minerals en masse from within the confines of a deposit in or very close to its original geological position. The desired mineral values are dissolved under the appropriate conditions of pH, oxidation potential and complexation, where hydrologic flow is induced by gravity or pressure gradients between application and recovery points (Hiskey, 1994).

2.9.1 LEACHING OPERATIONS

2.9.1.i Newmont's Heap Bioleach

A 500 000 t/y two stage heap leach process has been used at Mt Leyshon Gold Mine, northern Queensland since August 1992, treating gold/copper ore (Warhurst and Bridge, 1996). Newmont Mining (Nevada) is applying bio-oxidation to 660-ton heaps of refractory ore at mining operations on Nevada's Carlin Trend deposit, the largest gold deposit outside of South Africa.

2.9.1.ii *In Situ Leaching*

Santa Cruz is a field experiment currently being conducted near Casa Grande, Arizona. The Santa Cruz ore body is owned jointly by ASARCO Inc. and Freeport-McMoRan Gold Co. An *in situ* leaching approach is being considered for this copper oxide deposit (Hiskey, 1994). Copper Range Company will also investigate *in situ* leaching at the White Pine operation. This system will utilise an acid ferric sulphate lixiviant in conjunction with microorganisms (Hiskey, 1994)

Gunpowder, another test program is conducted at the Mammoth Mine of Gunpowder Copper Limited in Queensland, Australia. This project consists of leaching both broken ore below an old open pit and fractured ore in a test stope. Test work has shown that a copper recovery of 70% would be achieved over three years and that acid generation was rapid, indicating that bioleaching was taking place (Potts and Webb, 1994, Hiskey, 1994).

2.10 REFERENCES

Bailey, A.D. and G.S. Hansford (1993): "Factors affecting biooxidation of sulphide minerals at high concentrations of solids: A review.", *Biotech. Bioeng.*, **42**(10), 1164-1174

Barrett, J, M.N. Hughes, G.I. Karavaiko and P.A. Spencer (1993): *Metal extraction by bacterial oxidation of minerals*, Ellis Horwood, England, p73-102

Barrett, J., M.N. Hughes, A.M. Nabar, D.J. O'Reardon, D.K.Ewart and R.K. Poole (1988): "The isolation and characterisation of a moderately thermophilic mixed culture of autotrophic bacteria: Application to the oxidation of refractory gold concentrates", *RANDOL*, Perth, Western Australia, p 148-150

Boon, M., G.S. Hansford and J.J. Heijnen (1995): "The role of bacterial ferrous oxidation in the bio-oxidation of pyrite", in *Biohydrometallurgyical Processing, Volume I*, Eds T. Vargas, C.A. Jerez, J.V. Wiertz and H.Toledo, University of Chile

Boon, M. and J.J. Heijnen (1993): "Mechanisms and rate limiting steps in bioleaching of sphalerite, chalcopyrite and pyrite with *Thiobacillus ferrooxidans*", in *Biohydrometallurgical Technologies* Eds A.E. Torma, J.E. Wey and V.L. Laksmanan, The Minerals, Metals and Materials Society, p 217-235

Boogerd, F.C., C. van den Beemd, T. Stoellwinder, P. Bos and J.G. Kuenen (1991): "Relative contributions of biological and chemical reactions to the overall rate of pyrite oxidation at temperatures between 30°C and 70°C", *Biotech. Bioeng.*, **38**, 109-115

Brierley, C.L. and R. Brans (1994): "Selection of BacTech's thermophilic biooxidation proscess for Youanmi mine", in *BIOMINE '94*, Australian Mineral Foundation, Perth, Western Australia, Chapter 5

Brierley, C.L., J.A. Brierley, P.R. Norris and D.P. Kelly (1980): "Metal-tolerant micro-organisms of hot, acid environments", in *Microbial growth and survival in extremes of environment* Eds G.W. Gould and J.E.L. Corry, Academic Press, New York, p 39-51

Brierley, C.L. (1977): "Thermophilic micro-organisms in extraction of metals from ores", *Dev. Ind. Microbiol.*, **18**, 273-283

Brierley J.A and C.L. Brierley (1986): "Microbial mining using thermophilic micro-organisms", in *Thermophiles: General, molechular and applied microbiology*, T.D. Brock, John Wiley & Sons, New York, p 279-305

Brierley, J.A. (1978): "Thermophilic iron-oxidising bacteria found in copper leaching dumps", *Appl. Environ. Micro.*, **36**(3), 523-535

Brown, A.R.G., W. Irvine and P.A.R. Odd (1994): "Bioelaching – Wiluna operating experience", in *BIOMINE '94*, Australian Mineral Foundation, Perth, Western Australia, Chapter 16

Clark, D.A. and P.R. Norris (1996): "Oxidation of mineral sulphides by thermophilic microorganisms", *Minerals Engineering*, **9**(11), 1119-1125

Dempsey, P., P. Human, A. Pinches and J.W. Neale (1990): "Bacterial oxidation at Vaal Reefs", *International Deep Mining Conference: Innovations in Metallurgical Plant*, SAIMM, Johannesburg, South Africa, p 111-123-

Dew, D. (1997): "Gencor's commercialisation of biotechnology for metals recovery from sulphide ores", *Biotech SA '97*, Grahamstown, South Africa, p 15 – 16

Dutrizac, J.E. (1985): "Recent advances in the leaching of sulphides and the precipitation of iron", *MINTEK 50*, South Africa, p 39-62

Dutrizac, J.E. (1983): "Factors affecting alkali jarosite precipitation", *Metallurgical Transactions B*, **14B** (December), 531-539

Dutrizac, J.E. and R.J.C. MacDonald (1974): "Ferric ion as a leaching medium", *Minerals Sc. Engng*, **2**, 59-100

Escobar, B., J.M. Cassas, J. Mamani and R.B. Ohlbaum (1993): "Bioleaching of a copper concentrate with *Sulfolobus* BC", in *Biohydrometallurgical Technologies*, edited by A.E. Torma, J.E. Wey and V.L. Lakshmanan, The Minerals, Metals & Materials Society, p195-204

Ferreira, R.C.H. (1975): "High-temperature E-pH diagrams for the systems S-H_2O, Cu-S-H_2O and Fe-S-H_2O", in *Leaching and reduction in hydrometallurgy*, Ed A.R. Burkin, Institution of Mining and Metallurgy, p67-83

Gómez, E., M.L. Blázquez, A. Ballester and F. González (1996); "Study by SEM and EDS of chalcopyrite bioleaching using a new thermophilic bacteria", *Minerals Engineering*, **9**(9), 985-999

Haines, A.K. and P.C. van Aswegen (1990): "Process and engineering challenges in the treatment of refractory gold ores", *International Deep Mining Conference: Innovations in Metallurgical Plant*, SAIMM, Johannesburg, South Africa, 103-110

Hiskey, J.B. (1994): "*In situ* leaching recovery of copper – what's next?", in *BIOMINE '94*, Australian Mineral Foundation, Perth, Western Australia, Chapter 1

Hutchins, S.R., J.A. Brierley and C.L. Brierley (1988): "Microbial pretreatment of refractory sulphide and carbonaceous ores improves the economics of gold recovery", *Mining Engineering*, April, 249-254

Iglesias, N. and F. Carranza (1996): "Treatment of a gold bearing arsenopyrite concentrate by ferric sulphate leaching", *Minerals Engineering*, **9**(3), 317-330

Iglesias, N. and F. Carranza (1995): "Bacterial leaching of a copper ore rich in gold and silver: Study of the chemical stage", *Minerals Engineering*, **8**(10), 1089-1096

Iglesias, N., I. Palencia and F. Carranza (1993): "Removal of the refractoriness of a gold bearing pyrite-arsenopyrite ore by ferric sulphate leaching at low concentration", *EPD Congress*, Edited by J.P. Hager, The Mineral, Metals and Materials Society

Jordan, M.A., S. McGinness and C.V. Phillips (1996): "Acidophilic bacteria - their potential mining and environmental applications", *Minerals Engineering*, **9** (2), 169-181

Kametani, H. (1955): "The hydrolysis of ferric sulfate solution", *J. Min. Met. Inst. Japan*, **71**, 351-356

Karavaiko, G.I., S.I. Kuznetsov and A.I. Golonizik (1977): *The bacterial leaching of metals from ores*, Chapter IV, Technicopy Limited, England

Konishi, Y., S. Toshido and S. Asai (1995): "Bioleaching of pyrite by acidophilic thermophile *Acidanus brierleyi*", *Biotech. Bioeng.*, **48**, 592-600

Konishi, Y. S. Asai and H. Katoh (1990): "Bacterial dissolution of pyrite by *Thiobacillus ferrooxidans*", *Bioproc. Eng.*, **5**, 231-237

Lawrence, R.W. and P.B. Marchant (1988): "Comparison of mesophilic and thermophilic oxidation systems for the treatment of refractory gold ores and concentrates", in *Biohydrometallurgy 1987*, Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Warwick, United Kingdom, p359-374

Lawrence, R.W. and A. Bruynesteyn (1983): "Biological pre-oxidation to enhance gold and silver recovery from refractory pyritic ores and concentrates", *CIM Bulletin*, September, **76**(857), 107-110

Lindström, E.B., E. Gunneriusson and O.H. Tuovinen (1992): "Bacterial oxidation of refractory sulfide ores for gold recovery", *Criticial reviews in Biotechnology*, **12**(1/2) 133-155

Lizama, H.M. and I. Suzuki (1988): "Bacterial leaching of a sufide ore by *Thiobacillus ferrooxidans* and *Thiobacillus thiooxidans*: I. Shake flask studies", *Biotech. Bioeng.* **32**, 110-116

May, N. (1997): "The ferric leaching of pyrite", MSc Thesis, University of Cape Town, South Africa, p12-14

Mintek (1996): "Minbac™ for the world", Internal publication

Murr, L.E. and V.K. Berry (1979): "Observations of a natural thermophilic micro-organism in the leaching of a large experimental copper-bearing waste body", *Metallurgical Transactions B*, **10B** (December), 523-531

Natarajan, K.A. (1988): "Electrochemical aspects of bioleaching of base metal sulfides", *Min. Metallurgical. Proc.*, May, 61-65

Nicholson, H.M., G.R. Smith, R.J. Stewart. F.W. Kock and H.J. Marais (1994): "Design and commissioning of Ashanti's Sansu BIOX® plant", *BIOMINE '94*, Australian Mineral Foundation, Perth, Western Australia, Chapter 2

Norris, P.R. (1990): "Acidophilic bacteria and their activity in mineral sulphide oxidation", in *Microbial Mineral Recovery*, Eds H.L. Ehrlich and C.L. Brierley, McGraw-Hill, New York, p3-27

Norris, P.R. and D.W. Barr (1988): "Bacterial oxidation of pyrite in high temperature reactors", in *Biohydrometallurgy*, Eds P.R. Norris and D.P. Kelly, , Science and Technology Letters, Surrey, United Kingdom, p 532-536

Potts, T.S. and W.K. Webb (1994): "Mining and in-place bioleaching at Gunpowder's Mammoth Mine", *BIOMINE '94*, Australian Mineral Foundation, Perth, Western Australia, Chapter 3

Poulin, R. and R.W. Lawrence (1996): "Economic and environmental niches of biohydrometallurgy", *Minerals Engineering*, **9**(8), 799-810

Rossi, G. (1990): *Biohydrometallurgy*, McGraw-Hill Book Company, Hamburg, Chapter 5

Slabbert, W., D. Dew, M. Godfrey, D. Miller and P. van Aswegen (1992): "Commissioning of a BIOX® Module at Sao Bento Mineração", *RANDOL*, Vancouver, p 447-452

Torres, F. , M.L. Blázquez, F. González, A. Ballester and J.L. Mier (1995): "The bioleaching of different sulphide minerals using thermophilic bacte4ia", *Metallurgical and Materials Transactions B*, **26B**, June, 455-465

van Aswegen, P.C. (1993): "Bio-oxidation of refractory gold ores, the Genmin experience", *BIOMINE '93*, Australian Mineral Foundation, Glenside, Southern Australia, Chapter 15

Vitaya, V.B., J. Loizumi and K. Toda (1994): "A kinetic assessment of substantial oxidation by *Sulfolobus acidocaldarius* in pyrite dissolution", *J. Fermentation Bioengineering*, 77(5), 528-534

Warhurst, A. and G. Bridge (1996): "Improving environmental performance through innovation", *Minerals Engineering*, **9**(9), 907-921

Zheng, C.Q., C.C. Allen and R.G. Bautista (1986): "Kinetic study of the oxidation of pyrite in aqueous ferric sulphate", *Ind. Eng. Chem. Process Des. Dev.* **25**, 308-317

Part II

Thermophilic Microorganisms and Their Implementation in Bioleaching

3. High Temperature Bioleaching and Thermophiles

A variety of acidophilic bacteria can participate in the oxidation of mineral sulphides with the consequent solubilisation of some metals. The microorganisms used in bioleaching are able to oxidise both or either ferrous iron and sulphur. In industrial mineral leaching operations the combination of the different oxidation capacities of the various organisms is important (Norris, 1990).

The rate of biochemical reactions in mineral sulphide oxidation are likely to be limited by the availability of substrate from the solid, with the rate of mineral dissolution controlled by the particle size, diffusion rates at the mineral surface and the temperature. An increase in the temperature, for example, allows more rapid mineral dissolution by thermophilic bacteria growing at temperatures above the limit for *Thiobacillus ferrooxidans* activity, even though the rates of autotrophic growth of the thermophiles on ferrous iron in solution are similar to that of the mesophiles (Norris, 1989).

3.1 CONCEPT OF HIGH TEMPERATURE LEACHING

Biological leaching systems for sulphide minerals involve oxidation reactions, which are highly exothermic, releasing excess energy as heat and causing elevated temperatures in interiors of leach piles. Considerable heat generation also occurs in stirred tank reactors operating at high pulp density of sulphide minerals (Niemelä *et al.*, 1994). Tests of 16 *Thiobacilli*-type organisms cultured on ferrous iron demonstrate that mesophilic thiobacilli are confined to temperature ranges of 37 to 43°C (Niemelä *et al.*, 1994). Hence increases in temperature above this range would lead to the reduced activity of *Thiobacillus*-type organisms.

Several moderately and extremely thermophilic organisms, which may be amenable to bioleaching have been isolated from geothermal environments and dump leach operations. As mentioned in Section 2.8, the primary benefits to be derived from higher temperature operations are reduced cooling costs and improved kinetics (Lawrence and Marchant, 1988). Because of the long residence time of conventional bioleaching operations there is substantial interest in employing these advantages in a high temperature bioleaching process. Figure 3-1 illustrates the variation of cooling requirements for a 150 tpd BIOX® process (Miller, 1991). It is clear that an optimum temperature exists in the region of 65°C when considering cooling costs alone.

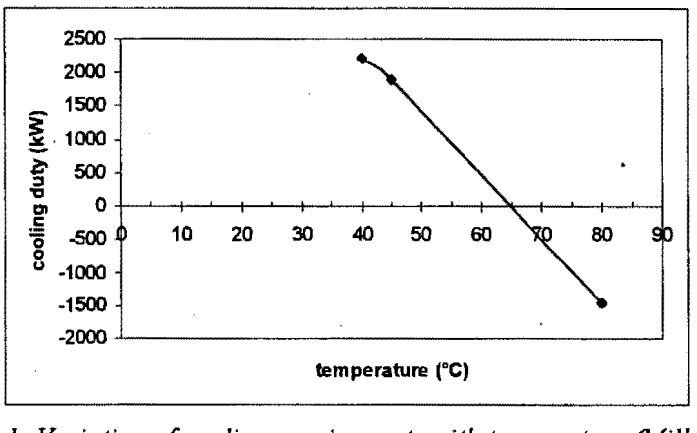


Figure 3-1: Variation of cooling requirements with temperature (Miller, 1991).

3.2 THERMOPHILIC ISOLATES

The heterogenous microenvironments and temperature gradients which exist in some mineral sulphide operations and geothermal natural habitats ensure that a variety of microorganisms will be present and active (Norris, 1990). These microorganisms employed are often categorised as mesophiles, moderate and extreme thermophiles. The respective optimum temperatures are in the region of 30, 50 and 70°C (Norris, 1990). Some mesophiles can more accurately be described as 'thermotolerant' with an optimum growth temperature around 40°C (Barrett *et al.*, 1988: Hallberg and Lindström, 1994). Further extreme thermophiles which grow above 70°C are being isolated (Stetter *et al.*, 1983, Adams, 1995). Thermophiles which grow significantly above these temperatures are termed hyper-thermophiles.

Thermoacidophilic organisms have been isolated from a variety of natural acidic thermal habitats, including geothermal springs, dump leach operations and burning coal tips. Table 3-1 details isolation information for various meso and thermophiles. For ease of reference in subsequent tables, each isolate is coded here by either 'T-' or 'S-' to denote whether they are *Thiobacillus*- or *Sulfolobus*- type organisms. This is followed by the first three letters of the primary author and the year of publication.

The geographical distribution of the areas of isolation is very diverse, as illustrated in Figure 3-2. Acidophilic thermophiles have been described since 1966: however, their taxonomy is continually being adjusted. For convenience, these thermophiles can be treated as two groups (Brierley *et.al.*, 1980):

1. the extreme thermophiles of the *Sulfolobus*-type (Brock *et al.*, 1972), having unusual morphology and growing at temperatures up to 84°C and
2. the moderate thermophiles and thermotolerant *Thiobacillus*-type organisms, generally rod-shaped eubacteria, growing at temperatures up to 55°C (Brierley, 1978).

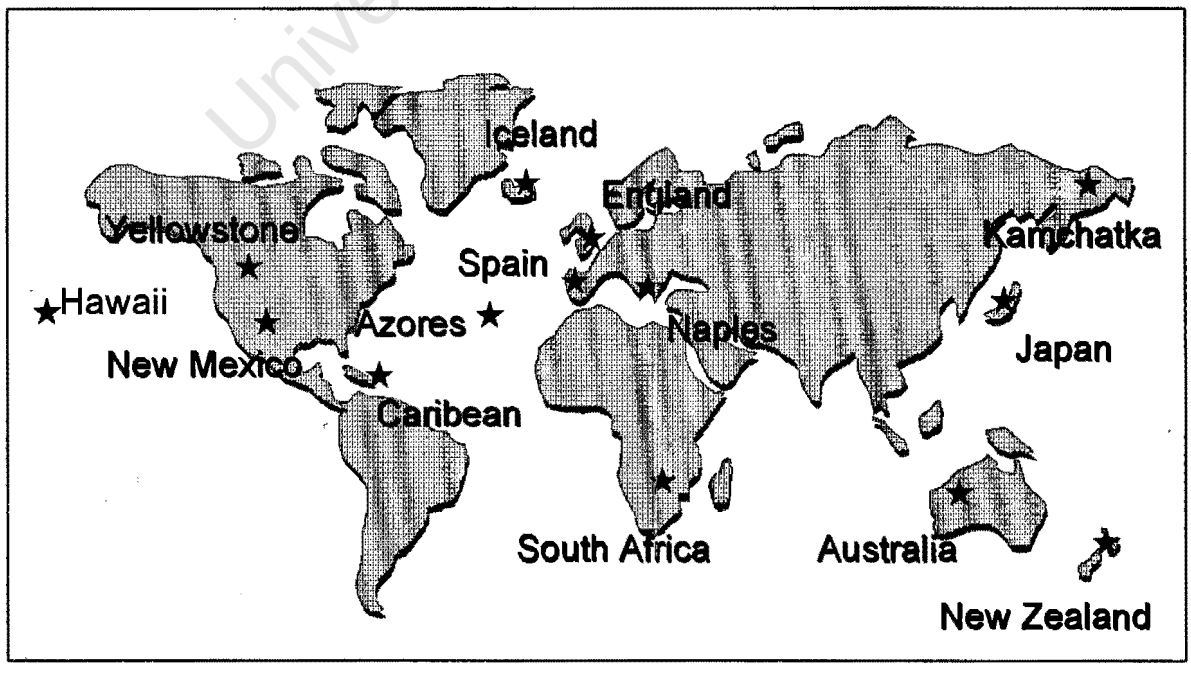


Figure 3-2: Geographical distribution of thermophilic isolates.

3.2.1 EXTREME THERMOPHILES

The extreme thermophiles are mainly comprised of archaebacteria of the *Sulfolobacae* family. The taxonomy is continually developing as more organisms are isolated and characterised.

3.2.1.i Sulfolobacae *Family*

Archae have recently been recognised as a phylogenetically distinct group of organisms. They are as distinct from the eubacteria as they are from the eucaryotes (Woese, 1987). The different isolates are classified according to their guanine-cytosine (GC) content, morphology, physiology and cell envelope characteristics (Zillig *et al.*, 1980) in Table 3-3. Nutrient media and energy sources used by these cultures are detailed in Appendix A.

All thermoacidophiles of the archaeal crenarchaeota kingdom are members of the *Sulfolobaceae* family (order Sulfolobales) as depicted in Figure 3-3. Up to now the genera *Sulfolobus*, *Acidianus*, *Desulfurolobus*, *Sulfurococcus* (Karavaiko *et.al.*, 1993) also known as *Metallosphaera* (Huber et al., 1989) have been described. They are characterised by, a coccoid shape, tolerance of the thermoacidophilic environment and the ability to oxidise elemental sulphur and ferrous iron (de Rosa and Gambacorta, 1975). In addition *Sulfolobaceae* possess glycoprotein sub-unit cell envelopes and caldariellaquinone (Zillig *et al.*, 1980), which has been used in the classification of these microorganisms (de Rosa and Gambacorta, 1975). The different genera within *Sulfolobaceae* can be distinguished from each other by their metabolic and biochemical properties as demonstrated in Table 3-3.

Members of *Sulfolobus* exhibit a GC-content of around 37 mol% and are able to utilise sugars, amino acids, complex organic substances and carbon dioxide as energy and carbon sources (Brock, 1977). Huber *et al.* (1989) claim that *S. acidocaldarius, S. shibatae* and *S. solfaraticus* are not able to extract metals from mineral sulphides. *Acidianus* and *Desulfurolobus* are closely related to each other, they show a GC-content around 31 mol% and a facultatively aerobic metabolism with elemental sulphur as electron donor or acceptor. They are weak ore leachers (Huber and Stetter, 1991). The first ore leaching acidophile, which had been described as *Sulfolobus brierleyi*, has recently been placed to the genus *Acidianus* (Zillig *et al.*, 1980). *Metallosphaera* is characterised by a GC-content of 45 mol% and a strong ore leaching capacity (Huber and Stetter, 1991).

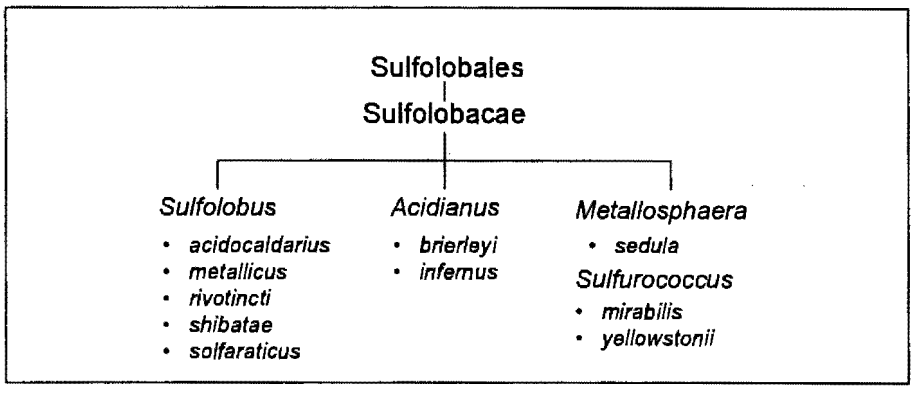


Figure 3-3: Sulfolobales order

Table 3-1 Thermophilic isolates

Code	Reference	Designated	Place	Temp. (°C)	pH	Agitation/Co_2	Substrates Tested
S-Boh75	Bohlool, 1975	NZ-74-1,3,4,6,8,9,13,15	Rototua Taupo region (hot springs), New Zealand	70	2	periodically by hand	yeast extract or elemental sulphur
S-Bri77	Brierley, 1977	Ferrolobus – *Acidianus brierleyi*	Yellowstone National Park	60°C	2.5		chalcopyrite and ferrous iron
S-DeR75	De Rosa and Gambacorta, 1975		Pisciarelli solfatara, Naples, Italy	74-89	1.4-2.6		sulphur, organic substrates, ferrous iron
S-Fur77	Furuya, et.al., 1977	TA-1	Owaku-dani hot spring, Japan	70	2-3	20 mm shaking 100 times per min	yeast extract and ferrous iron
S-Ger97	Gericke, 1997, pers. comm.	Thermophiles	Witbank, (coal dump) South Africa	70	1.6-1.8		pyrite and copper sulphide minerals
S-Góm96	Gómez *et al.*, 1996	*Sulfolobus rivotincti*	Rio Tinto mines, Huelva, Spain	68	1.8		Chalcopyrite, ferrous iron, 9K medium
S-Hub89	Huber *et al.*, 1989	*Metallosphaera sedula*	Pisciarelli Solfatara, Italy		2	shaking	sulphur, yeast extract supplement (0.2g/l), pyrite, chalcopyrite, sphalerite and various synthetic sulphides
S-Hub91	Huber and Stetter, 1991	*Sulfolobus metallicus*	Krafla, Kerlingarfjöll, Namarskarth and Hveragerthi (solfataric fields), Iceland	55-75	1-4.5	0.03% pulp density	pyrite, chalcopyrite or ore mixture
S-Kar93	Karavaiko *et al.*, 1993	*Sulfurococcus mirabilis, Sulfurococcus yellowstonii*	Yellowstone National Park (hot springs), USA and Kamchatka (volcanoes), Russia	70-75 60-65	2.0-2.6	rotary shaker	sulphur, Fe^{2+}, Cu-Zn-pyrite
S-Law97	Lawson, 1997.	*Sulfolobus* spp.	Ingwe coal mine, Natal, South Africa				
S-Mos74	Mosser, *et al.*,	*Sulfolobus* three	Yellowstone National Park (hot	63-80	3		elemental sulphur, 0.1% yeast extract

Code	Reference	Designated	Place	Temp. (°C)	pH	Agitation/Co_2	Substrates Tested
	1974	temperature strains	springs) , USA				
S-Muh95	Muhlbauer, 1997	Sulfolobus-type	Witbank coal dumps, South Africa	65-70	1.5		Pyrite and Arsenopyrite in the presence of yeast extract
S-Nor-86	Norris and Parrot, 1986	*Sulfolobus* BC *Sulfolobus* LM	Birch Coppice Colliery (UK), Lake Myvam, Iceland	70	2	150rpm, 0.5% (v/v) CO_2	pyrite, chalcopyrite, pyrrhotite and nickel: concentrates
S-Nor97	Norris, 1997	moderate and extreme thermophiles	Montserrat Volcano, Caribbean				
S-Seg86	Segerer *et. al*,, 1986	*Acidianus brierleyi* *Acidiamus infernus*	Solfatara, Italy: Caldeira de Velha, Azores: Krafla, Kerlingarfjöll, Namarskarth and Hveragerthi, Iceland and Yellowstone National Park, USA (solfataric fields)	70	2.5		S° (2g/l), organics
S-Vit94	Vitaya and Toda, 1994	*Sulfolobus acidocaldarius*	Beppu Hot springs, Japan	70	2	300rpm, 5% CO_2 in air	pyrite, ferrous iron and organic substrates
T-Bri78	Brierley, 1978	TH3 TH1	United States copper leach dump operation Iceland (thermal spring)	50-60	2.6		$FeSO_4$, 0.02% (w/v) yeast extract supplement
T-Dar71	Darland and Brock, 1971	15 different isolates	Yellowstone National Park, Hawaiian Volcano National Park, USA (hot springs)	45-70	2-5		yeast extract, glucose or ribose
T-Fli72 S-Fli72	Fliermans and Brock, 1972	*Thiobacillus thiooxidans*, *Sulfolobus acidocaldarius*	Yellowstone National Park (hot acid soils), USA	30-55 55-70	3		elemental sulphur, CO_2 enriched
T-Hall94	Hallberg and	*Thiobacillus caldus*	Coal spoils - UK (Paul Norris) ,	45	2-5	2% (v/v) CO_2	tetrathionate, 0.5% flowers of

Code	Reference	Designated	Place	Temp. (°C)	pH	Agitation/Co_2	Substrates Tested
	Lindström, 1994		Kingsbury (Marsh and Norris)				sulphur, ferrous sulphate, yeast extract
T-LeR77	Le Roux *et al.*, 1977	*Thiobacillus*-type	Hot springs, Iceland	50-75	3.5	200 rev/min shaking incubator	sulphide minerals, sulphur, ferrous iron
T-Mar83	Marsh and Norris, 1983	TH1, isolates from Warwickshire UK	Lake Myvam, Iceland: Warwiskshire coal heap, UK			120 rpm, 5% (v/v) CO_2	5% (w/v) pyrite, yeast extract
T-Nor80	Norris *et al.*, 1980	TH1, TH3		30	1.5		$FeSO_4$, mineral sulphides, yeast extract supplement

Table 3-2: Thermophilic mineral sulphide leaching

Metal Sulphide or Substrate	Strains examined	CO_2 in air	pH	Pulp Density	Reference
pyrite (FeS_2)	*Acidianus brierleyi* (60 and 68°C)			15%	Ngubane and Baecker, 1988
	Acidianus brierleyi (65°C), stirred reactor (500rpm)		1.5	10%	Konishi *et al.*, 1995
	Thiobacillus-type isolates (TH1) (50°C)	5%	1-2	5%	Marsh and Norris, 1983
	Thiobacillus ferrooxidans (33°C) and *Sulfolobus* BC (64°C)		1.6 - 3.0	2, 4, 8, 15, 20%	Lawrence and Marchant, 1988
	Sulfolobus acidocaldarius, S. solfataricus and *Acidianus brierleyi* (70°C)		2.0	2%	Larsson *et al., 1990*
	Sulfolobus BC (70°C) and mesophilic culture (30C), stirred reactors (100rpm) and air-lift reactors	0.5%	1.5	1-15%	Norris and Barr, 1988
	Enrichment from Iceland, *Sulfolobus* BC, *Metallosphaera sedula, Acidianus brierleyi* and	0-1%	1.75-2	1-5%	Norris and Owen, 1993

Metal Sulphide or Substrate	Strains examined	CO_2 in air	pH	Pulp Density	Reference
	Sulfobacillus thermosulfidooxidans (BC1)				
	Sulfolobus acidocaldarius and BC (70°C) (airlift rxor)	0.1-5%	2		Norris, 1989
	Sulfolobus BC (68°C) and mesophilic enriched culture	1%	1.4-2	5%	Barr *et al.*, 1992
	Sulfolobus acidocaldarius(70°C)	5%	2	1,2 and 4 g/600ml	Vitaya and Toda, 1994
	Sulfolobus BC (70°C)			1, 10 35%	Ngubane and Baecker, 1988
pyrite/sphalerite/ chalcopyrite♠	*Metallosphaera sedula*			0.5g/30ml	Huber *et al.*, 1989
	Sulfolobus BC (70°C)	1%	2	1,3,5%	Torres *et al.*, 1995
pyrite/arsenopyrite♣	*Thiobacillus ferrooxidans*, mixed mesophilc culture (30°C), *Sulfobacillus sulfidooxidans* (50°C), *Sulfolobus* BC and *Sulfolobus*-like ICHT (>70°C)	1%	2	1, 2, 5, 10%	Clark and Norris, 1996
pyrite/arsenopyrite‡	facultative thermophile (S16I) (50°C)	5%	1.5	5%	Hutchins *et al.*, 1988
pyrite/arsenopyrite*	*Thiobacillus ferrooxidans* (30°C), facultative thermophile (50°C) and *Sulfolobus* (60°C)	5%	1.5	5%	Hutchins *et al.*, 1988
pyrrhotite (Nickel present)	*Sulfolobus* LM (70°C)	0.5%	1.5	1, 4, 9%	Norris and Parrot, 1986
iron concentrate ♥	*Sulfolobus* BC (70°C)	1%	2	1,3 & 5%	Torres *et al.*, 1995
chalcopyrite ($CuFeS_2$)	**Flask tests**: *Acidianus brierleyi* ('Ferrolobus') (60°C), *Sulfolobus acidocaldarius* (60°C)		monitored	0.01%	Brierley, 1977
	Column Tests: *Acidianus brierleyi* ('Ferrolobus') (60°C), *Sulfolobus acidocaldarius* (60°C)		monitored	packed	Brierley, 1977
	Sulfolobus rivotincti (68°C)		1.8	1%	Gómez *et al.*, 1996

Metal Sulphide or Substrate	Strains examined	CO_2 in air	pH	Pulp Density	Reference
	Sulfolobus BC (68°C), *Thiobacillus ferrooxidans* (30°C) stirred and air-lift reactors up to 10l	1%	1.0 – 1.8	10, 15, 20, 25, 30%	Le Roux and Wakerley, 1988
	Sulfolobus BC and *Sulfolobus* LM (70°C)	0.5%	1.5	1,4,9%	Norris and Parrot, 1986
	Sulfolobus-comparable microorganism (60-75°C, 0.01% yeast extract)				Wyckoff and Davidson, 1977
	Enrichment from Iceland, *Sulfolobus* BC, *Metallosphaera sedula*, *Acidianus brierleyi* and *Sulfobacillus thermosulfidooxidans* (BC1)	0-1%	1.75-2	1-5%	Norris and Owen, 1993
	Sulfolobus BC, *Acidianus brierleyi*, other mesophiles and moderate thermophiles		1.2-1.6	3%	Jordan *et al.*, 1996
	Thiobacillus-type thermophile (60°C)		2.5		Brierley, 1974
	Extreme thermophiles (55°C)		2.4	10%	Chakraborti and Murr, 1980
copper concentrate[Ψ]	*Sulfolobus* BC (BC65), *Acidianus brierleyi*(DSM 1651), *Metallosphaera sedula*(DSM 5348) (68°C), *Thiobacillus ferrooxidans* (DSM 583, ATCC 11820) and a mesophilic uncharacterised enrichment culture	2%		5%	Jordan *et al.*, 1993
copper concentrate [†]	*Sulfolobus* BC (70°C)		1.6 - 1.8	0.5, 2 & 5%	Escobar *et al.*, 1993
copper concentrate[♣]	*Sulfolobus* BC (70°C)	1%	2	1,3 & 5%	Torres *et al.*, 1995
copper/zinc concentrates[§]	*Sulfolobus* BC (68°C) and mesophilic enriched culture	1%	1.4-2	5%	Barr *et al.*, 1992
zinc concentrate [♦]	*Sulfolobus* BC (70°C)	1%	2	1,3 & 5%	Torres *et al.*, 1995
molybdenite (98.5% MoS_2)	*Thiobacillus*-type thermophile (60°C)		2.5		Brierley, 1974

Metal Sulphide or Substrate	Strains examined	CO_2 in air	pH	Pulp Density	Reference
ferrous iron	thermophilic *Thiobacillus*-type (TH3) (50°C)				Bierley, 1978
	Acidianus brierleyi (65°C), stirred reactor (500rpm)		1.5		Konishi *et al.*, 1995
elemental sulphur	*Sulfolobus acidocaldarius* (70°C)	5%	2.5-3.2		Shivvers and Brock, 1973

† 31.82% pyrite, 18.31% covellite, 15.29% chalcopyrite, 14.22% chalcocite and 13.02% enargite (by weight)
‡ 50% pyrite, 15% arsenopyrite and 2% pyrrhotite
• 36% pyrite, 23% arsenopyrite, 4% pyrrhotite, 1% chalcopyrite and 1% sphalerite
♣ 15.8% Cu, 4.01% Zn, 17.53% Fe and 5.39% Pb
• 0.47% Cu, 51.12% Zn, 9.47% Fe and 2.31% Pb
▼ 0.18% Cu, 0.53% Zn, 55.86% Fe and 1.01% Pb
♦ Fe 6800 ppm, Zn 4500ppm, Cu 800 ppm, Pb 250 ppm, U 95 ppm and As 58 ppm
ξ 22.5% Cu, 0.3% Zn, 32.2% Fe and 29.8% S (one ore example)
▼ 21.2% Cu, 12.9% Zn, 34.25% Fe and 31.3% S (Wheal Jane mine, Cornwall)
ξ 19.1% Fe, 19.4% S, 3.8%As and 0.1%Cu and 30.2% Fe, 36.3% S, 10.2%As and 0.1%Cu

3.2.1.ii *Genus* Sulfolobus

Sulfolobus, lobed sulphur oxidisers, have been found in acid thermal springs at temperatures above 60°C in Yellowstone National Park, USA (Brock *et.al.*, 1972, Brierley and Brierley, 1978) and also in hot sulphur-rich, acidic soils of the same area (Fliermans and Brock, 1972). They have also been isolated from an Italian thermal spring near Naples (De Rosa and Gambacorta, 1975), Owaku-dani hot spring in Japan (Furuya *et al.*, 1977) and from New Zealand hot springs (Bohlool, 1975). *Sulfolobus* BC was isolated from a coal tip near Birch Coppice colliery, UK and *Sulfolobus* LM from Lake Myvam in Iceland (Marsh and Norris, 1983). *Sulfolobus* reportedly occurs in environments with a pH between 2-3 and temperatures above 55°C where an oxidisable energy source of either sulphur or ferrous iron is present (Brierley *et al.*, 1980).

Sulfolobus acidocaldarius was originally isolated from Yellowstone National Park (USA) by Brock *et al.* (1972). The derivation of the name is from '*Sulfo*', Latin sulphur, '*lobus*', Latin for lobe, '*acidio*' is Latin for acid and '*caldarius*' is Latin for hot (Brock, 1977). They characterised the organism as a thermoacidophilic, facultative sulphur-oxidising autotroph with an unusual cell wall structure. Brierley and Brierley (1978) isolated *Sulfolobus*-like organisms which were initially classified as *S. brierleyi*, however this species was subsequently re-classified as *Acidianus brierleyi* (Segerer *et al.*, 1986). De Rosa and Gambacorta (1975) isolated several organisms from Solfatara, Italy. These organisms were shown to be very similar to *Sulfolobus acidocaldarius* and *Acidianus brierleyi*. It was able to grow autotrophically on sulphur and ferrous iron and heterotrophically on organic carbon sources. The cells were spherical of approximately 1 µm diameter and lacked a rigid cell wall (De Rosa and Gambacorta, 1975). A study of the GC-content of the DNA showed that it was sufficiently

Table 3-3 Distinguishing characteristics - Extreme Thermophiles

	S-Fur77	S-Bohl75	S-Fli72	S-Hub91	S-Kar93	S-Seg86	S-Ger97	S-Hub89
optimum growth temp (°C)	70	70	50-85	55-75		70	70°C	75°C
optimum growth ph	2-3		2.5-3	1-4.5		1.5-2	1.6-1.8	1-4.5
cell shape	lobed spheres	lobed spheres	lobed spheres	irregular, lobed cocci	coccal cells, small chains	irregular coccoid, sometimes lobed	lobed spheres	irregular cocci, slightly spherical
cell size ((µm)	0.8-1.2			1.5µm	0.8-1.7	0.5-2		0.8-1.2
cell wall sub-unit structure	present			present	present	present		present
gram stain	negative			negative	negative	negative		negative
GC-content (mol %)	44.2			38	43-44.6	31		44
% NaCl				0-3		0.1-4		
Pili					present	non-motile		up to 3µm

See Table 3-1 for details of reference.

distinct from *Sulfolobus acidocaldarius*. This was later classified as *Sulfolobus solfataricus* (Zillig *et al.*, 1980).

Fliermans and Brock (1972) conducted a study of solfatara soils in the Yellowstone National Park. Across a depth of 20cm, the temperature ranged from 20 to 90°C. *Thiobacillus thiooxidans* and *Sulfolobus* spp. were distributed in soil samples according to the temperature. The pH ranged from 1-6, but this did not noticeably affect the bacterium distribution. Serologically different strains of *Sulfolobus acidocaldarius* coexist in both flowing and non-flowing springs (Bohlool and Brock, 1974). Different temperature strains of *S. acidocaldarius* coexist in different springs. These cultures had temperature optima of 65, 70 and 80°C (Mosser *et al.*, 1974). Thus it is evident that a significant diversity of organisms coexist in solfatara regions.

Gericke (1997) isolated spherical, lobed thermophiles from burning coal dumps near Witbank, South Africa. They grow at 70°C between pH 1.6 and 1.8 on pyrite and copper sulphides. The exact classification of these organisms has not yet been clarified. Muhlbauer (1997) also isolated *Sulfolobus*-type organisms from Witbank. These organisms grew on pyrite and arsenopyrite in the presence of yeast extract. Further, *Sulfolobus* species have been isolated from Ingwe coal waste heap in Natal, South Africa (Lawson, 1997).

3.2.1.iii *Genus* Acidianus

Acidianus is derived from Latin for acid and Ianus (Latin), a mythical roman figure with two faces looking in opposite directions. Thus it can be translated to acidic bifaced bacterium, reflecting the ability of these organisms to grow both aerobically and anaerobically (Segerer *et al.*, 1986).

Table 3-4 Distinguishing characteristics - Moderate Thermophiles and Mesophiles

	T-Fli72	TH1	TH2	T-Dar71	T-Hall94	TH3
optimum growth temp (°C)	20-55	37-55°C		45-70	45	37-55°C
growth ph	2.5			2-5	1-4	
cell shape	rods	rod shaped	rod shaped	rods	short rods	rod shaped
cell size ((µm)		1.6-3.2	1.6-4.2		1.2-1.8	1.1-1.6
cell wall sub-unit structure						
gram stain		negative	negative	gram variable	negative	negative
GC-content (mol %)				62	63	
flagellum					1, polar	
See Table 3-1 for details of reference.						

Work was conducted by Brierley and Brierley (1978) on a *Sulfolobus* type organism classified as *S. brierleyi*, and later reclassified as *Acidianus brierleyi*. Isolates were collected from Solfatara Crater and Pisciarelli Solfatara (Naples) and Vulcano, Porti di Levante in Italy: Caldeira de Velha in Azores: Krafla, Kerlingarfjöll, Namarskarth and Hveragerthi in Iceland and Yellowstone National Park in USA. 26 different isolate cultures were maintained. These isolates were all facultative aerobes growing by means of two contrary modes of chemolithotrophy. At extremely anaerobic conditions, the organisms grew autotrophically forming H_2S. Under aerobic conditions in the presence of elemental sulphur, they were able to grow by oxidation of S°, forming sulphuric acid (Segerer *et al.*, 1986). Unlike *Sulfolobus acidocaldarius* and *S. solfataricus*, these isolates were unable to grow without elemental sulphur by oxidising organic compounds and are therefore strictly chemolithotrophic (Segerer *et al.*, 1986). A new species *A. infernus* was classified (Segerer *et al.*, 1986).

3.2.1.iv Metallosphaera *and* Sulfurococcus *Genera*

Metallosphaera, 'the metal-mobilising sphere' (Huber *et al.*, 1989), is an aerobic, facultatively chemolithoautotrophic organism. Lithotrophic growth on sulphidic ores like pyrite, sphalerite and chalcopyrite and on elemental sulphur is known to occur. There is evidence of organotrophic growth on yeast extract but no evidence of sugar utilisation. A species was classified as *M. sedula*. The Latin adjective 'sedulus' implies busy, describing the efficient metal mobilisation (Huber *et al.*, 1989). Based on metabolism and GC-content, these organisms resemble the thermophiles isolated by Furuya *et al.* (1977) in Japan (Table 3-1), as well as the Golovacheva *et al.* (1987) and Karavaiko *et al.* (1993) isolates mentioned below.

Karavaiko *et al.* (1993) isolated a new genus, *Sulfurococcus* from Yellowstone National Park, USA and Kamchatka, Russia. These organisms exhibited the ability to utilise elemental sulphur, ferrous iron and pyrite as an energy source. The maximum specific growth rates, obtained for growth on these substrates were 0.033, 0.023, and 0.004 h^{-1} respectively in the presence of 0.02% yeast extract (Karavaiko *et al.*, 1993). This illustrates that the organisms have a preference for growth on elemental sulphur.

3.2.2 MODERATE THERMOPHILES (INCLUDING THERMOTOLERANT MESOPHILES)

Bacteria belonging to the genus *Thiobacillus* are able to oxidise reduced sulphur compounds. Other substrates for these bacteria include ferrous iron, molecular hydrogen and various organic compounds and sulphide minerals (Hallberg and Lindström, 1994). While *Thiobacillus* are typically termed mesophiles, Fliermans and Brock (1972) claim the existence of *Thiobacillus thiooxidans*-type organisms, which grow at temperatures up to 55°C. For the purposes of comparison, these microorganisms will be included in the group of moderate thermophiles. Details of some of these isolates are contained in Table 3-1 and 3-4.

Natural habitats of moderate thermophiles include acidic thermal springs and hot, acid soils of volcanic solfatara. Artificial habitats include coalmine wastes, metalliferous ores and copper leach dumps (Table 3-1). These artificial habitats are of particular interest because of the association of thermophilic microbes with metal

leaching operations. In one instance, moderate thermophiles were found in association with *Sulfolobus* spp. in the Birch Coppice colliery (Brierley and Brierley, 1986). Studies of growth on pyrite show that these moderate thermophiles oxidise metal sulphides as well as ferrous iron (Brierley and Brierley, 1986).

Three *Thiobacillus*-like strains have been described (Table 3-1): TH1 isolated from a thermal spring in Iceland (Brierley *et al.*, 1978), TH2 isolated from an experimental copper leach system and TH3 isolated from a copper leach dump in southwestern New Mexico, USA (Brierley *et al.*, 1980: Brierley, 1978: Norris *et al.*, 1980). These organisms are reputed to have a broad temperature range (30 - 55°C) and use either iron or metal sulphides as an energy source (Brierley *et al.*, 1980). Some moderate thermophiles require yeast extract for growth (Brierley, 1978).

Another *Thiobacillus*-like thermophile, classified as *Thiobacillus thermosulfidooxidans* (Brierley, 1978) was isolated in Russia. It has a temperature optimum of 50°C and is able to oxidise sulphur, ferrous iron, pyrite and other minerals in the presence of yeast extract. Marsh and Norris (1983) contrasted moderately thermophilic *Thiobacillus*-type organisms from Lake Myvam, Iceland and Warwickshire and Alvecote, United Kingdom. Copper solubilisation at 37°C was similar in the cases of the Lake Myvam isolate and *Thiobacillus ferrooxidans*. As the temperature increased beyond the tolerable range of *T. ferrooxidans*, so the extent and rate of copper solubilisation in the presence of the Lake Myvam isolate increased beyond that by *T. ferrooxidans*.

Darland and Brock (1971) isolated 15 cultures from Yellowstone National Park and Hawaiian Volcano National Park, USA. These cultures were gram-variable spore-forming rods. Two strains isolated from Warwickshire, United Kingdom (Hallberg and Lindström, 1994), classified *Thiobacillus caldus*, were examined in terms of their amenability to implementation in bioleaching. This strain has been identified by immunobinding assay as a significant substituent in pilot scale leaching operations (Amaro *et al.*, 1994). Although these microorganisms are unable to oxidise ore, they can form a significant part of bioleaching cultures in reactors (Hallberg and Lindström, 1994). A strain, designated M4, isolated by Barrett *et al.* (1988) has been implemented in a BacTech commercial process at Youanmi Mine in Western Australia (Section 2.9.3).

Some of these moderate thermophiles have the ability to grow heterotrophically on yeast extract. There was evidence that suggested a slight inhibition of pyrite oxidation by yeast extract. It is apparent that when using yeast extract as the primary energy source, the organisms revert to a fermentative mode of metabolism (Brierley and Brierley, 1986). They seem to require supplemented carbon dioxide (1-10% v/v in air) as do *Sulfolobus*-like organisms (Brierley and Brierley, 1986: Norris and Owen, 1993). The taxonomy of these organisms remains unresolved, however it has been suggested that they form a new genus *Sulfobacillus,* including *S. thermosulfidooxidans* and *S. acidophilus* (Brierley and Brierley, 1986; Norris *et al.*, 1996).

3.3 MINERAL SULPHIDE THERMOPHILIC BIOLEACHING

Sulfolobus BC, an organism frequently implemented in bioleaching studies (Table 3-2) is reputed to be *Sulfolobus metallicus* (Lawson, 1997), which has demonstrated ore leaching ability (Huber and Stetter, 1991). Other species which are capable of weak ore leaching belong to the genera *Acidianus, Metallosphaera* also

referred to as *Sulfurococcus* (Karavaiko *et al.*, 1993) and *Desulfurolobus* (Huber and Stetter, 1991: Huber *et al.*, 1989). The dissolution of sulphide mineral concentrates by some *Sulfolobus*-type organisms in the region of 70°C has been described as more rapid, sometimes three to six fold faster than that by mesophilic metal mobilisers (Norris and Barr, 1988: Konishi *et al.*, 1995: Le Roux and Wakerley, 1987).

Since the isolation of *Sulfolobus acidocaldarius* (Brock *et al.*, 1972), the metallurgical potential of the organism has been discussed and assessed for the leaching of copper (Chakraborti and Murr, 1980: Brierley, 1980: Le Roux and Wakerley, 1987), molybdenum (Brierley, 1974), and other minerals including pyrite and arsenopyrite (Marsh and Norris, 1983: Norris and Parrot, 1986). Application in the desulfurisation of coal has also been studied (Kargi and Robinson, 1985; 1986). Characteristics such as tolerance to high acidity, high metal concentrations, and rapid oxidation of pyrite have been claimed (Chakraborti and Murr, 1980: Kargi and Robinson, 1985: Norris and Parrott, 1986). Jordan *et al.* (1996) comment that 'the question still remaining to be answered ... is whether the higher temperature acidophiles have the ability to function under a wide range of very dynamic conditions.' Table 3-2 summarises the implementation of various thermophiles for different sulphide minerals and energy substrates.

Table 3-5: Sulfolobus leaching experiments in comparison with other microorganisms.

CULTURE	CONDITIONS	MAXIMUM SOLUBILISATION RATE	REFERENCE
Sulfolobus BC	5% (w/v) pyrite in an air-lift vessel, 70 °C	200 mg iron/l/h	Norris and Barr,1988
Sulfolobus BC	1% (w/v) pyrite in a stirred reactor (100 rpm), further pyrite additions to 9%, 70 °C	40 mg iron/l/h	Norris and Barr, 1988
mesophiles	1% (w/v) pyrite in a stirred reactor (100 rpm), further pyrite additions to 9%, 30 °C	40 mg iron/l/h	Norris and Barr, 1988
Sulfolobus BC	35 g chalcopyrite in 700 ml medium, stirred reactor (500 rpm), 68°C	36 mg copper/l/h 83% copper extraction	Le Roux and Wakerley, 1988
Thiobacillus	35 g chalcopyrite in 700 ml medium, stirred reactor (500 rpm), 30 °C	8 mg copper/l/h 19% copper extraction	Le Roux and Wakerley, 1988
Thiobacillus ferrooxidans	440 ml air lift reactors2% (w/v) pyrite/arsenopyrite (30 °C)	13 mg iron/l/h 38 mg sulphate/l/h	Clark and Norris, 1996
Sulfobacillus thermosulfidooxidans	440 ml air lift reactors2% (w/v) pyrite/arsenopyrite (48 °C)	30 mg iron/l/h 75 mg sulphate/l/h	Clark and Norris, 1996
Sulfolobus BC	440 ml air lift reactors2% (w/v) pyrite/arsenopyrite (70 °C)	70 mg iron/l/h 225 mg sulphate/l/h	Clark and Norris, 1996

Among the thermophilic microorganisms, the most widely studied are those of the genus *Sulfolobus*, which oxidise sulphur at temperatures between 55°C and 80°C and from pH 0.9 to 5.8. Although they have not been used in an industrial scale, laboratory experiments in stirred reactors have shown appreciable leaching kinetics in attacking especially refractory sulphides, such as chalcopyrite (Le Roux and Wakerley, 1983) and in the bio-

desulphurisation of coal (Torres *et al.*, 1995). Maximum iron solubilisation rates are of the order of 200 mg/l/h (Norris and Barr, 1987), as demonstrated in Table 3-5.

Brierley (1977) experimented with *Sulfolobus acidocaldarius* and *Acidianus brierleyi* in reactors and shake flasks. Flask tests indicated that *A. brierley* preferred ferrous iron to chalcopyrite as an energy substrate. Ultimately Brierley concluded that the presence of *A. brierleyi* enhanced chalcopyrite oxidation. This is in agreement with the 'Two-Step' mechanism (Section 2.6.2) as the chemical action of ferric iron would continue the solubilisation of copper from the chalcopyrite. Vitaya and Toda (1994) modelled the kinetics of ferrous iron oxidation and direct pyrite leaching by a *Sulfolobus* isolate from Beppu Hot Springs, Japan. They observed successful pyrite leaching. The yield of cells per gram of ferrous iron and per gram of pyrite were of the same order of magnitude (~3-4x10^{11} cells /g substrate), not indicating a preference for either substrate.

Huber *et al.* (1989) observed growth of *Sulfolobus acidocaldarius* (DSM 639), *S. solfataricus* (DSM 1616), *Acidianus brierleyi* (DSM 1651), *A. infernus* (DSM 3191), and *Metallosphaera sedula* sp. nov. in the presence of various sulphidic ores. With the *Sulfolobus* strains examined, no microbial metal extraction was evident after three weeks, when compared with the sterile control tests. Larsson *et al.*, 1990 examined the capability of *Sulfolobus acidocaldarius*, *S. solfataricus* and *Acidianus brierleyi* to oxidise pyrite. Neither *S. acidocaldarius* nor *S. solfataricus* showed evidence of pyrite oxidation. These experiments were conducted in 500ml shake flasks with the DSM recommended nutrient medium.

Huber *et al.* (1989) grew *Acidianus brierleyi* autotrophically on ore mixtures. Final cell concentrations were generally low (~1 x 10^7 cells/ml) but increased with the addition of 0.02% yeast extract. Metal ion extraction by *A. brierleyi* continued in the presence of yeast extract (Huber *et al.*, 1989). *A. infernus* is able to grow autotrophically on ore mixtures, although metal mobilisation with or without an organic substrate is rather low (Huber *et al.*, 1989). Table 3-6 displays a comparison of gold extraction from a refractory sulphide mineral enhanced by three cultures at 30, 50 and 60°C. The maximum rate of iron solubilisation is achieved by *Acidianus brierleyi*. The exact experimental conditions were not given in the review (Brierley, 1990).

Table 3-6: Comparison of biologically enhanced gold extraction from refractory sulphidic gold ore (Brierley, 1990)

Parameter	*Thiobacillus ferrooxidans*	Moderate thermophiles	*Acidianus brierleyi*
Incubation temperature (°C)	30	50	60
Maximum rate of iron solubilisation (mg Iron/h)	52.1	61.0	177.1
Total iron extraction (%)	29.4	47.1	84.4
Gold recovery by cyandiation of bioleach residue	55.5	56.2	91.0

Metallosphaera sedula exhibits aerobic growth on sulphidic ores including pyrite, chalcopyrite, sphalerite and combinations of these ores (Huber *et al.*, 1989). They are also able to grow on elemental sulphur, forming sulphate. The presence of yeast extract (0.005%) did not alter the rate of sulphate formation significantly (Huber *et al.*, 1989). It was observed that *M. sedula* was capable of leaching sulphide ores ten times faster than *A. brierleyi*. In column leaching experiments on pyrite at 65°C, their metal extraction was more than 50 times

higher than values of mesophilic metal mobilisers (*e.g. Thiobacillus ferrooxidans* ATCC 23270 at 30°C). It is clear that *Metallosphaera* are good candidates for mineral sulphide bioleaching.

3.3.1 CHALCOPYRITE LEACHING

Chalcopyrite leaching is singled out by virtue of the fact that it is widely studied within the context of extreme thermophiles. Copper recovery from chalcopyrite by mesophilic strains of bacteria, characterised by *Thiobacillus ferrooxidans*, is typically less than 50% (Le Roux and Wakerley, 1988). Implementing either a silver (Ag^+) catalyst or thermophilic micro-organisms, such as *Sulfolobus*, increases copper recovery to greater than 90% in laboratory scale investigations (Jordan *et al.*, 1993; Le Roux and Wakerley, 1988; Escobar *et al.*, 1993). Increased leaching of copper from non-refractory copper minerals at 60°C was found not to be an inherent microbial characteristic, but rather a response to temperature. However, in the case of chalcopyrite minerals, the difference between the extent of leaching achieved by *Thiobacillus ferrooxidans* (8%) and *Sulfolobus* spp. (78%) could not be attributed only to an increase in temperature (Brierley and Brierley, 1986).

Work by Le Roux and Wakerley (1988) considered the potential of *Sulfolobus* BC in the leaching of chalcopyrite. It demonstrated that *Sulfolobus* BC, adapted to high copper concentrations (27g Cu/l), leached chalcopyrite with higher efficiency of copper extraction than obtained when using *Thiobacillus ferrooxidans* as illustrated in Figure 3-4 and Table 3-5. The high extraction obtained indicated that the problem of incomplete chalcopyrite oxidation associated with the use of *Thiobacillus ferrooxidans* does not occur when *Sulfolobus* is used. Suggesting that *Sulfolobus* could offer a more economically attractive route for the bioleaching of chalcopyrite (Le Roux and Wakerley, 1988). As microbially assisted solubilisation of chalcopyrite is the result of a combination of chemical and microbially mediated reactions enhanced chemical leaching (Section 2.6.3) must not be excluded.

Gómez *et al.* (1996) implemented a novel thermophile *Sulfolobus rivotincti*, isolated from the Rao Tinto mines in Huelva, Spain. This organism had an optimum growth temperature of 68°C and pH 1.8. It was claimed that the metallic sulphide mineral leaching capacity was high. It was adapted to acidic environments and high metallic cation concentrations (Gómez *et al.*, 1996). Experiments were conducted in 250ml shake flasks with massive and powdered chalcopyrite samples. Their results showed evidence of direct attack by the microorganisms on the mineral in the absence of an alternative energy source. The experimental set-up chosen did not include or pre-empt the occurrence of shear damage as the pulp density of the chalcopyrite concentrate was 1% and the stirrer speed used was 150rpm.

Sulphur is a product of chalcopyrite chemical leaching according to:

$$CuFeS_2 + 4Fe^{3+} \rightarrow Cu^{2+} + 5Fe^{2+} + 2S$$

Norris (1997) states that an additional role of the microorganisms in chalcopyrite bioleaching is to reduce elemental sulphur concentrations. Hence, implementation of extreme thermophiles capable of utilising sulphur is advantageous.

3.3.2 THERMOPHILE SENSITIVITY TO PULP DENSITY

The inhibition effect of agitation on the activity of *Sulfolobus,* which is more acute in the presence of high concentrations of minerals, restricts its potential industrial application (Norris and Barr, 1988). Jordan *et al.*, (1993) use air lift reactors as opposed to CSTR'S to reduce inactivation of the organisms possibly due to shear effects. Norris and Barr (1988) commented that at 70°C, a significant ferrous iron concentration, which was not present at 30°C was found in solution. Barr *et al.* (1992) found a concomitant release of ferrous iron into solution with chalcopyrite leaching. This is generally taken as an indication of a limitation in microbial oxidation of ferrous iron (Jordan *et al.*, 1993; Barr *et al.*, 1992).

Extremely thermophilic archae such as *Sulfolobus* and *Acidianus* spp. lack a rigid peptidoglycan cell wall (König and Stetter, 1986: König, 1988: Michel *et al.*, 1980). Further the fluidity of cellular membranes increases with temperature (Kelly and Deming, 1988). A combination of these factors results in a tendency for archae to be sensitive to shear. There is presently no clear understanding of the mechanism or onset of shear damage.

Studies using *Sulfolobus* generally implement low mineral concentrations or pulp densities. (up to 5% (w/v)). Norris and Barr (1988) conducted work with *Sulfolobus* BC at 70°C and a mixed, pyrite-enriched culture at 30°C and 10 % mineral concentration. The maximum rate of iron solubilisation by *Sulfolobus* BC in a stirred

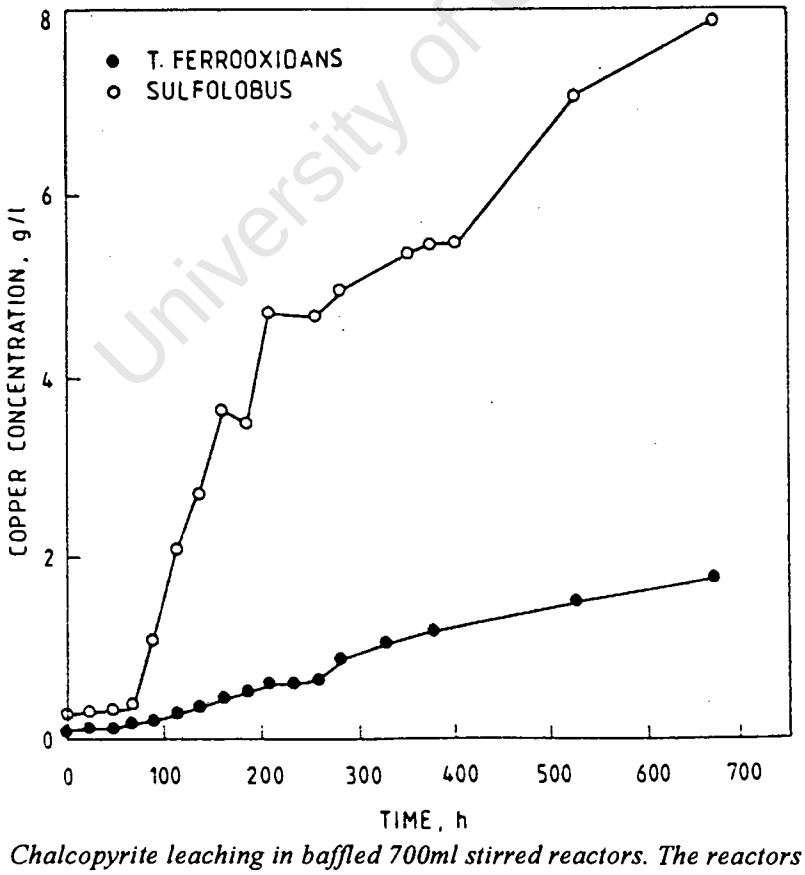


Figure 3-4: Chalcopyrite leaching in baffled 700ml stirred reactors. The reactors were agitated at 500rpm and run at 50g/l chalcopyrite (Le Roux and Wakerley, 1988).

tank reactor was 20% of that achieved in an air lift reactor (Table 3-5). The reactors were operated at 1% and 5% w/v pyrite respectively. *Sulfolobus* was observed to be inhibited by the high mineral concentration. A lag phase between an initial high, non-biological, iron solubilisation stage and the onset of biologically catalysed leaching was observed (Norris and Barr, 1988). *Sulfolobus* was implemented in air lift reactors with a central draft tube at higher mineral concentrations without incurring a lag phase before the onset of bioleaching. Pyrite was added stagewise. Inhibition was observed when the mineral concentration was increased to 15% (Norris and Barr, 1988).

A study was conducted looking at the effect of pulp density by Le Roux and Wakerley (1988). It was found that the greatest copper solubilisation rate was achieved at 15% (w/v). This dropped rapidly until no additional cell growth was observed at a pulp density of 30% (w/v) (Le Roux and Wakerley, 1988). Torres *et al.* (1995) conducted similar work with *Sulfolobus* BC strain at pulp densities of 1, 3 and 5% (w/v). They noticed an increase in the induction period with an increase in pulp density of a copper ore. Further, there was a reduction in the rate of dissolution of copper, zinc and iron from the respective ores, with an increase in pulp density. They attribute this to the greater sensitivity of *Sulfolobus* to agitation in the presence of the mineral.

Escobar *et al.*, (1993) implemented *Sulfolobus* BC in the bioleaching of a copper concentrate. Batch experiments were conducted at different pulp densities: 0.5%, 2% and 5% (w/v), Table 3-7. A reduced copper dissolution rate accompanied by lower redox potential and cell population was observed at higher pulp densities, as indicated in Table 3-7. This was attributed to the abrasive effect of the mineral. At higher pulp densities it was noted that the organisms were not able to maintain a high redox potential. When the rate of ferric iron consumption by the concentrate is faster than the rate of ferric iron regeneration by the bacteria, the ratio of ferric to ferrous iron decreases. As a result the redox or electrochemical potential of the solution decreases. Thus, it can be concluded that the rate of leaching is controlled (or limited) by the bacterial generation of ferric iron. This situation occurs when the bacterial activity is low, which may be induced by a higher than optimal pulp density (Escobar *et al.*, 1993).

Table 3-7: Variation of copper extraction by Sulfolobus BC with pulp density.

CULTURE	CONDITIONS	COPPER EXTRACTION			REFERENCE
Sulfolobus BC	batch reactor (500rpm), 70°C, various pulp densities of a copper concentrate affected copper recovery and max redox potential	solids (w/v %)	Eh (mV)	Cu (%)	Escobar *et al.*, 1993
		0.5	580	85	
		2	500	72	
		5	400	32	

Thus, there is clear indication of a negative effect of high pulp densities on the activity of thermophiles. It remains to be conclusively demonstrated why *Sulfolobus* strains and other extreme thermophiles are unable to tolerate higher concentrations of minerals in agitated culture: detrimental shear forces, physical attrition, chemical toxicities and unfavourable mass transfer properties can be considered as possible reasons (Clark and Norris, 1996).

3.3.3 METAL TOLERANCES:

The moderate thermophiles show a tolerance to metal concentrations similar to that of *Thiobacillus ferrooxidans* (Brierley *et al.*, 1980). *Sulfolobus* spp. can leach copper from chalcocite, chalcopyrite (Le Roux and Wakerley, 1987), molybdenum from molybdenite (Brierley, 1974). Further, they can grow at molybdenum concentrations exceeding that which is tolerated by *Thiobacillus, e.g. Acidianus brierleyi* tolerated 2000ppm hexavalent molybdenum (Brierley, 1974). The extent and rate of chalcopyrite leaching bySulfolobus spp. were greater than those with *Thiobacillus* spp. (Section 3.3.1).

Clark and Norris (1996) found that an arsenic concentration of over 4g/l did not inhibit the iron oxidising capacity of *Sulfolobus* BC. Lindström and Gunneriusson (1990) showed that the leaching rate of arsenopyrite by *Sulfolobus* BC decreased with increasing pulp density. They attributed this reduction to the toxicity of arsenic. By operating the process semi-continuously, and removing the leachate, arsenate did not increase beyond 30mM (Lindström and Gunneriusson, 1990). Jordan *et al.* (1996) also found that arsenic played a detrimental role in bioleaching by *Sulfolobus* BC and *Acidianus brierleyi*.

Brierley (1974) conducted copper concentration studies with a microbe isolated from Yellowstone National Park which grew in a temperature range between 45 and 75°C. This organism is likely to be a type similar to *Sulfolobus* spp. It was observed that the organism was able to respire in copper concentrations greater than 10g/l, however cell growth was inhibited at concentrations greater than 1g/l. Le Roux and Wakerley (1988) used a progressive adaptation technique to increase the tolerance of *Sulfolobus* BC from 3 to 27 g Cu/l over 18 months.

Miller *et al.* (1992) investigated the tolerances of several *S. acidocaldarius* and *S. solfataricus* strains to transition metals. Specific metals such as cadmium (Cd), cobalt (Co), copper (Cu), nickel (Ni), zinc (Zn) and magnesium (Mg) were examined. A definite tolerance pattern arose with the different strains. The main conclusion was that the tolerances pattern was a function of the acidic environment and therefore the response of the metal to the solution and not necessarily gene encoded. Contrary to Brierley (1980) and Norris (1990), Miller *et al.* (1992) concluded that in general the *Sulfolobus* spp. have a lower tolerance than *Thiobacilli*, that is ~0.3mM for Mg, ~1.0mM for Cd, Zn, Co and Ni and less than 1.0mM for Copper.

No differences between autotrophic and heterotrophic growth of *Metallosphaera sedula* was observed when arsenic, cadmium, cobalt, copper, antimony, uranium and zinc ions were present (Huber *et al.*, 1989). To compare with *Thiobacillus ferrooxidans* ATCC 23270, a higher resistance against cadmium and molybdenum was observed. The resistance to arsenic and uranium ions was similar for both organisms. *M. sedula* was more sensitive to the presence of silver, copper, mercury, antimony and zinc ions.

3.3.3.i *Adaptation to High Metal Concentrations*

It has been shown that the copper tolerance of *Sulfolobus* spp. can be increased through adaptation. The level achieved was such that it enabled *Sulfolobus* BC to be considered as a viable alternative to *Thiobacillus* organisms (Le Roux and Wakerley, 1987). Torres *et al.*(1995) observed that *Sulfolobus* growth in the presence of a copper concentrate was more efficient after progressive adaptation to greater pulp densities (up to 5% w/v). *Metallosphaera sedula* isolated by Huber *et.al.* (1989) was capable of adapting to grow on sulphidic ores by the following process:

- incubation in a medium containing elemental sulphur (1%), ferrous sulphate (1%) and yeast extract (0.1%):
- transfer and incubation in a medium containing a pyrite, sphalerite, chalcopyrite-ore mixture (3.3%), ferrous sulphate (1%) and yeast extract (0.005%):
- transfer and incubation in a medium containing the ore mixture (3.3%) and ferrous sulphate (1%):
- transfer and incubation in a medium containing only ore mixture.

The final cell densities were about 8×10^7 cells/ml for the first three steps and 2×10^8 /ml for the final step.

3.4 COAL DESULPHURISATION

Coal is a major energy source in several countries including South Africa. *Sulfolobus* and *Thiobacillus* spp. are considered in the implementation of coal desulphurisation (Kilbane, 1989: Olsson *et al.*, 1994: Karavaiko and Lobyreva, 1994: Brierley and Brierley, 1986). Sulphur content in coals is known to vary from 0.5 to 11%. Inorganic sulphur is present mainly in the form of pyrite and to a lesser extent as elemental sulphur and sulphides of other minerals. Pyrite occurs as finely dispersed intrusions in the coal (Karavaiko and Lobyreva, 1994).

The combustion of coal results in the emission of sulphur oxides. Several chemical and physical processes for removing sulphur before combustion have been proposed (Kilbane, 1989). The chemical processes operate at high temperatures (100-400°C) and are energy intensive, whereas physical methods such as flotation result in energy loss by removing coal particles containing finely disseminated pyrite. A microbial coal desulphurisation process would occur under mild reaction conditions compared to the chemical processes. The methods involved in microbial desulphurisation :

- require low capital and operating costs,
- are more energy efficient, and
- can remove finely distributed pyrite without any loss of coal.

Therefore it is believed that a microbial coal desulphurisation process may be the basis for an economically feasible pre-combustion coal desulphurisation commercial process (Maka and Cork, 1990). This is very feasible for the removal of inorganic sulphur (pyrite) but not for organic sulphur which forms a more integral part of the molecular coal matrix and is not readily accessible to microbial attack (Maka and Cork, 1990). However, Kargi and Robinson (1986) found that *Sulfolobus acidocaldarius* was capable of removing nearly 44% of all the

organic sulphur in coal. Other organisms including *Acidianus brierleyi* and a *Thiobacillus* type moderate thermophile have shown potential in this implementation (Maka and Cork, 1990).

3.5 DISCUSSION

Thermophilic bioleaching can be considered a feasible process if the following criteria are demonstrated:

- oxidation of ferrous iron and /or reduced sulphur
- fast oxidation kinetics
- operation at high pulp densities
- tolerance to low pH and high metal concentration
- efficient gold or valuable metal recovery

Sulfolobus BC (*a.k.a. Sulfolobus metallicus*), *Acidianus brierleyi* and *Metallospheara* have shown good ore leaching capability for a range of different mineral sulphides. Bioleaching kinetics achieved with *Sulfolobus* BC and *A. brierleyi* are three to six fold faster than those achieved with *Thiobacillus*-type organisms. Hence, faster oxidation kinetics are achievable with these microorganisms. In conjunction with faster chemical oxidation rates, this would lead to a faster overall rate of bioleaching and a concomitant improvement in economic efficiency.

Maximum leach rates achieved with *Sulfolobus* BC in stirred tank reactors at high pulp densities (15%) are similar if not less than that achieved with mesophiles. However, when air-lift reactors are employed at similar pulp densities, the maximum leaching rates achieved are significantly increased. From this it is evident that *Sulfolobus* BC are shear sensitive, yet implemented in reactor configurations which reduce shear, high pulp densities can be tolerated. As will be seen, Archae have a different cell wall structure to bacteria. Instead of a rigid peptidoglycan cell wall they have a porous, crystalline glyco-protein surface layer. Whether this surface layer affords similar protection to the cells in a slurry environment and the mechanism of damage still needs to be investigated.

There is some discrepancy as to the metal concentrations which *Sulfolobus* BC and other extreme thermophiles can tolerate. The general opinion, with the exception of arsenic metal, is that these microorganisms tolerate a similar level of metal concentrations in comparison with *Thiobacillus* spp. Further, the fact that these thermophiles successfully leach a variety of sulphide minerals at pulp densities up to 30% (Table 3-2) indicates an adequate metal tolerance. Further, *Sulfolobus* BC is readily adapted to high (27 g/l) copper concentrations.

In connection with gold or metal value recovery from sulphide minerals, there is no evidence that *Sulfolobus*-type organisms are less efficient than *Thiobacillus*-type mesophiles. To the contrary, the extent of chalcopyrite leaching by *Sulfolobus* spp. at 60°C was 70% greater than that achieved by *Thiobacillus ferrooxidans* at 30°C and otherwise similar operating conditions. This increase can be attributed to both an increase in temperature and the implementation of thermophiles.

Based on the criteria put forward, *Sulfolobus*-type extreme thermophiles, are capable of leaching sulphide minerals at higher temperatures. A possible downfall is the need to minimise shear damage. Another important factor is the need for increased dissolved carbon dioxide concentrations, generally 0.5-1% in air. As the solubility of oxygen and carbon dioxide decreases with temperature, the effect of temperature on gas-liquid mass transfer is significant.

3.6 REFERENCES

Adams, M.W.W. (1995): "Thermophilic archaea: An overview", in *Archaea, A loaboratory Manual, Thermophiles*, Eds F.T. Robb, A.R. Place, K.R. Sowers, H.J, Schreier, S. DasSarma and E.M. Fleischmann, Cold Spring Harbor Laboratory Press, United States of America, p3-8

Amaro, A.M., K.B. Hallberg, E.B. Lindström and C.A. Jerez (1994): "An immunological assay for detection and enumeration of thermophilic biomining microorganims", *Appl. Env. Micro.*, **60**(9), 3470-3473

Barr, D.W., M.A. Jordan, P.R. Norris and C.V. Phillips (1992): "An investigation into bacterial cell, ferrous iron, pH and Eh interctions during thermophilic leaching of copper concentrates", *Minerals Engineering*, **5**(3-5), 557-567

Barrett, J., M.N. Hughes, A.M. Nabar, D.J. O'Reardon, D.K.Ewart and R.K. Poole (1988): "The isolation and characterisation of a moderately thermophilic mixed culture of autotrophic bacteria: Application to the oxidation of refractory gold concentrates", *RANDOL*, Perth, Western Australia, p 148-150

Bohlool, B.B. (1975): "Occurrence of *Sulfolobus acidocaldarius*, an extremely thermophilic acidophilic bacterium, in New Zealand Hot Springs", *Arch. Microbiol.*, **106**(3), 171-174

Bohlool, B.B. and T.D. Brock (1974): "Population ecology of *Sulfolobus acidocaldarius* II. Immunoecological studies", *Arch. Microbiol.*, **97**, 181-194

Brierley, C.L., J.A. Brierley, P.R. Norris and D.P. Kelly (1980): "Metal-tolerant micro-organisms of hot, acid environments", in *Microbial growth and survival in extremes of environment*, Eds G.W. Gould and J.E.L. Corry, Academic Press, New York, p 39-51

Brierley, C.L. (1974): "Leaching: Use of a high temperature microbe", *Solution Mining Symposium*

Brierley, C.L. (1977): "Thermophilic micro-organisms in extraction of metals from ores", *Dev. Ind. Microbiol.*, **18**, 273-283

Brierley, J.A. (1990): "Acidophilic thermophilic archaebacteria: potential application for metals recovery", *FEMS Microbiology Reviews*, **75**, 287-292

Brierley, J.A. and C.L. Brierley (1986): "Microbial mining using thermophilic microorganisms", in *Thermophiles: General, molecular and applied microbiology*, Ed T.D. Brock, John Wiley & Sons, New York, Chapter 12

Brierley, J.A. (1978): "Thermophilic iron-oxidising bacteria found in copper leaching dumps", *Appl. Environ. Micro.*, **36**(3), 523-535

Brierley, J.A. and C.L. Brierley (1978): "Microbial leaching of copper at ambient and elevated temperatures", in *Metallurgical applications of bacterial leaching and related microbiological phenomena"*, Eds L.E. Murr, A.E. Torma and J.A. Brierley, Academic Press, New York, p477-490

Brierley, J.A., P.R. Norris, D.P. Kelly and N.W. Le Roux (1978): "Characteristics of a moderately thermophilic and acidophilic iron-oxidising *Thiobacillus*", *European J. Appl. Microbiol. Biotechnol.*, **5**, 291-299

Brock, T.D. (1977): "Thermophilic Microorganisms and Life at High Temperatures", Springer Verlag, New York, p 1-66, 117-178

Brock, T.D., K.M. Brock, R.T. Belly and R.L. Weiss (1972): "*Sulfolobus*: A new genus of sulphur-oxidising bacteria living at low pH and high temperature", *Arch. Mikrobiol.*, **84**, 54-68

Chakraborti, N. and L.E. Murr (1980): "Kinetics of leaching of chalcopyrite-bearing waste rock with thermophiles and mesophilic micro-organisms", *Hydrometallurgy*, **5**, 337-354

Clark, D.A. and P.R. Norris (1996): "Oxidation of Mineral Sulphides by Thermophilic Microorganisms", *Minerals Engineering*, **9** (11), 1119-1125

Darland, G. and T.D. Brock (1971): "*Bacillus acidocaldarius* sp. nov., an acidophilic thermophilic spore-forming bacterium", *J. Gen. Microbiology*, **67**, 9-15

De Rosa, M. and A. Gambacorta (1975): "Extremely thermophilic acidophilc bacteria convergent with *Sulfolobus acidocaldarius*", *J. Ind. Microbiology*, **86**, 156-164

Escobar, B., J.M. Cassas, J. Mamani and R.B. Ohlbaum (1993): "Bioleaching of a copper concentrate with *Sulfolobus* BC", in *Biohydrometallurgical Technologies*, edited by A.E. Torma, J.E. Wey and V.L. Lakshmanan, The Minerals, Metals & Materials Society, p195-204

Fliermans, C.B. and T.D. Brock (1972): "Ecology of sulphur oxidising bacteria in hot acid soils", *J. Bacteriology*, **111**(2), 343-350

Furuya, T., T. Nagumo, T. Itoh and H. Kaneko (1977): "A thermophilic acidophilic bacterium from hot springs", *Agric. Biol. Chem.*, **41**(9), 1607-1612

Gericke, M. (1997): Personal Communication, MINTEK, South Africa

Golovacheva, R.S., K.M. Valiejo-Roman and A.V. Troitsky (1987): "*Sulfurococcus mirabilis* gen. nov. sp. nov., a new thermophilic archaebacterium oxidising sulphur.", *Mikrobiologiya*, **56**, 100-107: cited in Huber *et al.*, 1989

Gómez, E., M.L. Blázquez, A. Ballester and F. González (1996): "Study by SEM and EDS of chalcopyrite bioleaching using a new thermophilic bacteria", *Minerals Engineering*, **9**(9), 985-999

Hallberg, K.B. and E.B. Lindström (1994): "Characterisation of *Thiobacillus caldus* sp. Nop., a moderately thermophilic acdophile", *Microbiology*, **140**, 3451-3456

Huber, G. and K.O. Stetter (1991): "*Sulfolobus metallicus*, sp. nov., a novel, strictly chemolithoautotrophic thermophilic archaeal species of metal mobilisers", *System. Appl. Microbiol.*, **14**, 372-378

Huber, G., C. Spinnler, A. Gambacorta and K.O. Stetter (1989): "*Metallosphaera sedula* gen. and sp. nov. represents a new genus of aerobic, metal mobilising, thermoacidophilic archaebacteria", *System. Appl. Microbiol.* **12** 38-47

Hutchins, S.R., J.A. Brierley and C.L. Brierley (1988): "Microbial pre-treatment of refractory sulphide and carbonaceous ores improves the economics of gold recovery", *Mining Engineering*, April, 249-254

Jordan, M.A., S. McGinness and C.V. Phillips (1996): "Acidophilic bacteria - their potential mining and environmental applications", *Minerals Engineering*, **9** (2), 169-181

Jordan, M.A., D.W. Barr and C.V. Phillips (1993): "Iron and sulphur speiation and cell surface hydrophobicity during bacterial oxidation of a complex copper concentrate", *Minerals Engineering*, **6**(8-10), 1001-1011

Karavaiko, G.I. and L.B. Lobyreva (1994): "An overview of the bacteria and archaea involved in removal of inorganic and organic sulphur compounds from coal", *Fuel Processing Technology*, **40**, 167-182

Karavaiko, G.I., R.S. Golovacheva, O.V. Golyshina, K.M Valiejo-Roman, V.K. Bobrova, A.V. Troitsky and T.A. Pivovarova (1993): "*Sulfurococcus*- a new genus of thermoacidophilc archae-bacteria oxidising sulphur, ferrous iron and sulphide minerals", in *Biohydrometallurgical Technologies,* edited by A.E. Torma, J.E. Wey and V.L. Lakshmanan, The Minerals, Metals & Materials Society, p 685-694

Kargi, F. and J.A. Robinson (1986): "Removal of organic sulphur from bituminous coal, Use of thermophilic organism *Sulfolobus acidocaldarius*", *FUEL*, **65**, 397-399

Kargi,F. and J.A. Robinson (1985): "Biological removal of pyritic sulfur from coal by the thermophilic organism *Sulfolobus acidocaldarius*", *Biotech. Bioeng.*, **27**, 41-49

Kelly, R.M. and J.W. Deming (1988): "Extremely thermophilic archaebacteria: Biological and engineering considerations", *Biotechnology Progress*, **4**(2), 47-62

Kilbane, J.J. (1989): "Desulfurization of coal: the microbial solution", *TIBTECH*, **7**(April), 97-101

König, H. (1988): "Archaebacterial cell envelopes", *Can. J. Microbiol.*, **34**, 395-406

König, H. and K.O. Stetter (1986): "Studies on archaebacterial S-layers", *System. Appl. Microbiol.*, **7**, 300-309

Konishi, Y., S. Yoshida and S. Asai (1995): "Bioleaching of pyrite by acidophilic thermophile *Acidianus brierleyi*", *Biotech. Bioeng.*, **48**, 592-600

Larsson, L., G. Olsson, O. Holst and H.T. Karlsson (1993): "Oxidation of pyrite by *Acidianus brierleyi*: Importance of close contact between pyrite and the microorganisms"

Lawrence, R.W. and P.B. Marchant (1988): "Comparison of mesophilic and thermophilic oxidation systems for the treatment of refractory gold ores and concentrates", in *Biohydrometallurgy '87,* Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Warwick, United Kingdom, p359-374

Lawson, E. (1997): Personal communication, Microbiology, University of Witwatersrand, South Africa

Le Roux N.W. and D.S Wakerley (1988): "Leaching of chalcopyrite ($CuFeS_2$) at 70°C using *Sulfolobus*", in *Biohydrometallurgy '87*, Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Surrey, United Kingdom, p 305-317

Le Roux, N.W., D.S. Wakerley and S.D. Hart (1977): "Thermophilic *Thiobacillus*-type bacteria from icelandic thermal areas", *J. Gen. Microbiology*, **100**, 197-201

Lindström, E.B. and L. Gunneriusson (1990): "Thermophilic bioleaching of arsenopyrite using *Sulfolobus* and a semi-continuous laboratory procedure", *Journal of Industrial Microbiology*, **5**, 375-382

Maka, A. and D.J. Cork (1990): "Introduction to the sulphur micro-organisms and their applications in the environment", *Dev. Ind. Mirobiology*, β31β(5), 99-102

Marsh, R.M and P.R. Norris (1983): "Mineral sulphide oxidation by moderately thermophilic acidophilic bacteria", *Biotech. Letters*, **5**(9), 585-590

Michel, H., D.-Ch. Neugebauer and D. Oesterhelt (1980): "The 2-D crystalline cell wall of *Sulfolobus acidocaldarius*: Structure, solubilisation and reassembly", in *Electron Microscopy at Molecular Dimension*, Eds W. Baumeister and W. Vogell, Springer-Verlag, Berlin Heidelberg, p27-35

Miller, D.M. (1991): "Effect of temperature on BIOX® operations", Colloquim, Bacterial Oxidation, South African Institute of Mining and Metallurgy, Johannesburg, 18 June

Miller, K.W., S.S. Risanico and J. B. Risatti (1992): "Differential tolerance of *Sulfolobus*strains to transition metals", *FEMS Microbiology Letters*, **93**, 69-74

Mosser, J.L., A.G. Mosser and T.D. Brock (1974): "Population ecology of *Sulfolobus acidocaldarius* I. Temperature strains", *Arch. Microbiol.*, **97**, 169-179

Muhlbauer, R. (1997): Personal Communication, Gencor Process Research, South Africa

Niemelä, Sivelä, Luoma and Tuovinen (1994): "Maximum temperature limits for acidophilic mesophilic bacteria in biological leaching systems", *Applied Environmental Microbiology*, **60**(9), 3444-3446

Ngubane, W.T. and A.A.W. Baecker (1988): "Thermophilic pyrite leaching by *Sulfolobus brierleyi*", in *Biohydrometallurgy '87*, Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Surrey, United Kingdom, p 528-529

Norris (1997): Personal Communication, University of Warwick, United Kingdom

Norris, P.R., D.A. Clark, J.P. Owen and S. Waterhouse (1996): "Characterisitics of *Sulfobacillus acidophilus* sp. nov. and other moderately thermophilic mineral-sulphide-oxidizing bacteria", *Microbiology*. **142**, 775-783

Norris, P.R. and J.P. Owen (1993): "Mineral sulphide oxidation by enrichment cultures of novel thermoacidophilic bacteria", *FEMS Microbiology Reviews*, **11**, 51-56

Norris, P.R. (1990): "Acidophilic bacteria and their activity in mineral sulphide oxidation", in *Microbial Mineral Recovery*, Eds H.L. Ehrlich and C.L. Brierley, McGraw-Hill, New York, p 3-27

Norris, P.R. (1989): "Factors affecting bacterial mineral oxidation: The example of carbon dioxide in the context of bacterial diversity", in *Biohydrometallurgy 1989*, Eds J Salley, R.G.L. McCready and P.L. Wichlacz, Jackson-Hole Wyoming, p 3-14

Norris, P.R. and D.W. Barr (1988): "Bacterial oxidation of pyrtie in high temperature reactors", in *Biohydrometallurgy '87*, Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Surrey, United Kingdom, p 532-536

Norris, P.R. and L. Parrot (1986): "High temperature, mineral concentrate dissolution with *Sulfolobus*", in *Fundamental and Applied Biohydrometallurgy*, Elsevier, Amsterdam, Netherlands, p 355-365

Norris, P.R., J.A. Brierley and D.P. Kelly (1980): "Physiological characteristics of two facultatively thermophilic mineral-oxidising bacteria", *FEMS Microbiology Letters*, 7, 119-122

Olsson, G., B.-M. Pott, L.Larsson, O. Holst and H.T. Karlsson (1994): "Microbial desulfurisation of coal by *Thiobacillus ferrooxidans* and thermophilic archaea", *Fuel Processing Technology*, **40**, 277-282

Segerer, A., A. Neuner, J.K. Kristjansson and K.O. Stetter (1986): "*Acidianus infernus* gen. Nov.,sp. nov., and *Acidianus brierleyi* comb. nov.: Facultatively aerobic, extremely acidophilic thermophilic sulphur-metabolising archaebacteria", *International Journal of Systematic Bacteriology*, **36**(4), 559-564

Shivvers, D.W. and T.D. Brock (1973): "Oxidation of elemental sulphur by *Sulfolobus acidocaldarius*", *J. Bacteriology*, **114**(2), 706-710

Stetter, K.O., H. König and E. Stackebrandt (1983): "*Pyrodictium* gen. nov., a new genus of submarine disc-shaped sulphur reducing archaebacteria growing optimally at 105°C", *System. Appl. Microbiol.*, **4**, 535-551

Torres, F. , M.L. Blázquez, F. González, A. Ballester and J.L. Mier (1995): "The bioleaching of different sulphide minerals using thermophilic bacte4ia", *Metallurgical and Materials Transactions B*, **26B**, June, 455-465

Vitaya, V.B., J. Loizumi and K. Toda (1994): "A kinetic assessment of substantial oxidation by *Sulfolobus acidocaldarius* in pyrite dissolution", *J. Fermentation Bioengineering*, 77(5), 528-534

Woese, C.R. (1987): "Bacterial Evolution", *Microbiology Reviews*, **51**, 221-271

Wyckoff, R.W.G. and F.D. Davidson (1977): "The composition , morphology and action upon chalcopyrite of autotrophs recovered from fumaroles" in *GBT Conference of Bacterial Leaching*, Ed. W. Schwartz, Verlag Chemie Weinheim, New York

Zillig, W., K.O. Stetter, S. Wunderl, W. Schulz, H. Priess and I. Scholz (1980): "The *Sulfolobus-"Calariella"* Group: Taxonomy on the basis of the structure of DNA-dependent RNA polymerases", *Arch. Microbiol.*, **125**, 259-269

4. Archae and *Sulfolobus* spp.

An interesting aspect of the order *Sulfolobales* (Section 3.2.1) is that they are members of the Archaeaobiota kingdom. This kingdom is phylogenetically distinct from the eucaryotes and procaryotes (Woese, 1987). There are three biological kingdoms: Archaeobiota, Eucaryotes and Eubacteria, the relationships between these kingdoms are represented in Figure 4-1. The names of the kingdoms all have greek derivatives. Archae is derived from the Greek word *archaios* meaning ancient, *eu* is Greek for true, *karyo*, radical form for nucleus and *bakterion*, small rod (Prescott *et al.*, 1993). The archae inhabit very extreme conditions, including high temperatures, acidic conditions and high halide concentrations. From this it is proposed that they existed 3-4 billion years ago, hence their name and the substantial interest in their phylogenetic position (Danson, 1988; Fewson, 1986).

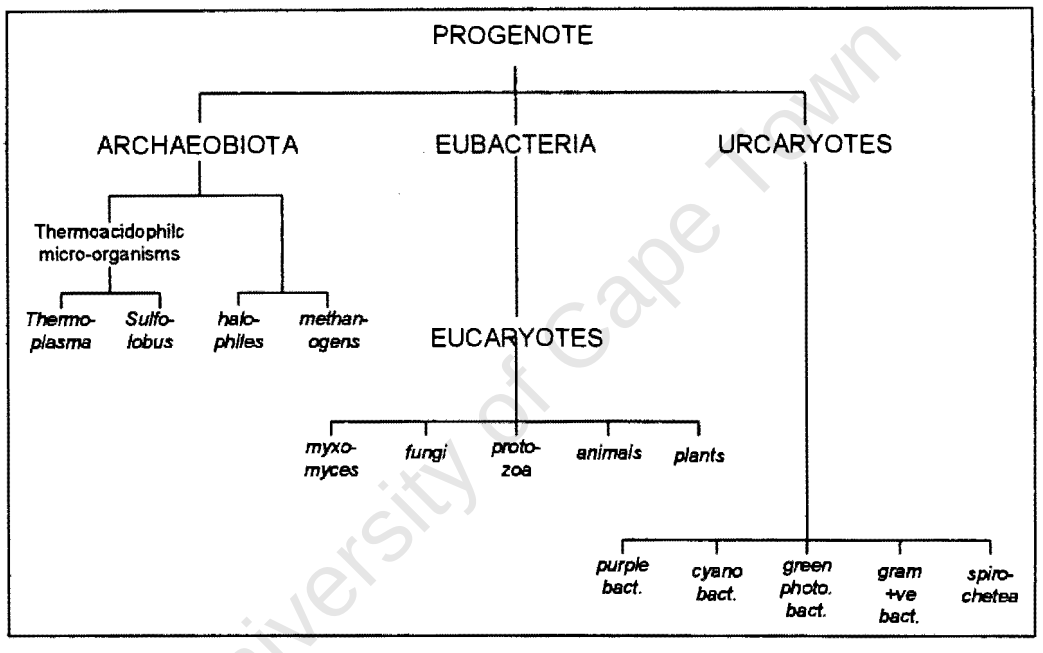


Figure 4-1: The three primary kingdoms according to Woese (1987).

Previously it was accepted that there were only two kingdoms, the procaryotes and the eucaryotes. Carl Woese and George Fox (Woese, 1981) were instrumental in demonstrating the existence of a third grouping, using RNA (Ribosomal Nucleic Acid) sequences. 16S/18S sequences of rRNA (ribosomal RNA) act as molecular chronometers. It is argued that they are able to measure both large and small phylogenetic distances based on the supposition that random changes, which occur in genetic sequences, are cumulative. Large phylogenetic distances are therefore denoted by large variations in these RNA molecules. Various archae, eubacteria and eucaryotes have been sequenced. The complete sequence data supports a clear distinction between the three kingdoms. Phylogenetic trees have been estimated on the basis of this data (Woese, 1987).

This chapter illustrates the characteristics of the archaeobiota kingdom and elaborates on the metabolic pathways and structure, particularly of the *Sulfolobus* spp. Furthermore it endeavors to investigate the

significance of this information in terms of attachment, carbon dioxide fixation and general functioning of these organisms in the mineral slurry environment.

4.1 ARCHAE MICROORGANISMS

As a group, the archae are diverse both in morphology and physiology. Morphologically speaking, they can be rod shaped, spherical, spiral, lobed or plate shaped. They can stain either gram negative or gram positive, the significance of which is mentioned in Section 4.2.2. Their size varies from 0.1 to 1.5 μm in diameter. They vary physiologically in that they can be aerobic, facultatively anaerobic or strictly anaerobic. Nutritionally they range from chemolithoautotrophs to organotrophs. Moreover they can be either mesophilic or thermophilic. As illustrated in Figure 4-1, there are several different phenotypes: methanogens, halophiles, thermophiles and sulphur reducing organisms. Figure 4-2 illustrates the relative grouping of the archae into genera. Table 4-1 summarises the different groups, as defined by Bergey's manual (Stanley *et al.*, 1989) within the archae (Prescott *et al.*, 1993). Only the more significant groups will be discussed in this chapter.

Table 4-1: Characteristics of the major archae groups (Prescott et al., 1993).

Group of Archae	*General Characteristics*	Representative Genera
Methanogenic	Methane is the major metabolic end product. S^0 may be reduced to H_2S without energy production. Cells possess coenzyme M, factors 420 and 430 and methanopterin.	*Methanobacterium* *Methanococcus* *Methanomicrobium* *Methanosarcina*
Sulfate reducers	H_2S is formed from sulfate by dissimilatory sulfate reduction. Traces of methane also formed. Extremely thermophilic and strictly anaerobic. Possess factor 420 and methanopterin, but not coenzyme M or factor 430.	*Archaeoglobus*
Extreme Halophiles	Rods and regular to very irregular cells. Gram-negative or gram-positive, aerobic or facultatively anaerobic chemoorganotrophs. Require high sodium chloride concentrations for growth (≥1.5M). Neutrophilic or alkalophilic. Mesophilic or slightly thermophilic. Some species contain bacteriorhodopsin and use light for ATP synthesis.	*Halobacterium* *Halococcus* *Natronpbacterium*
Cell wall-less archaeobacteria	Coccoid cells lacking a cell envelope. Thermoacidophilic. Aerobic. Plasma membrane contains a mannose-rich glycoprotein and lipoglycan.	*Thermoplasma*
Extremely thermophilic S°-metabolisers	Gram-negative rods, filaments or cocci. Obligately thermophilic (optimum growth temperature between 70-105°C). Aerobic, facultatively aerobic, or strictly anaerobic. Acidophilic or neutrophilic. Autotrophic or heterotrophic. Most are sulphur metabolisers.	*Desulfuroccocus* *Methanopyrus* *Pyrodictium* *Sulfolobus* *Thermococcus* *Thermoproteus*

4.1.1 METHANOGENIC ARCHAE

These microorganisms are strict anaerobes, which utilise CO_2, H_2, formate, methanol, acetate and other compounds. Methane is the major metabolic end product. As a result of this, their metabolism is rather unusual. They constitute the largest group of archae, consisting of at least three orders and 13 genera. Within the group there is a great diversity with respect to structure such as cell walls and morphology (Prescott *et al.*, 1993).

4.1.2 EXTREMELY HALOPHILIC ARCHAE

These organisms also constitute a major group of archae, with six genera in one family. They are aerobic chemoheterotrophs with respiratory metabolism which require complex nutrients, usually proteins and amino acids for growth. They can be motile or non-motile (Prescott *et al*,. 1993).

A distinctive trait of this group is their dependence on a high concentration of sodium chloride (NaCl), greater than 1.5 M. They have a growth optimum in the range of 3 - 4M NaCl. The cell wall of some organisms disintegrates in the absence of a high concentration of NaCl. These organisms grow in areas such as the Dead Sea or the Great Salt Lake in Utah. They contain carotenoids, resulting in a red-yellow pigmentation, which are suspected to provide protection against strong sunlight (Prescott *et al.*, 1993).

4.1.3 EXTREMELY THERMOPHILIC SULPHUR-METABOLISERS

These organisms are extremely thermophilic, many of which are thermoacidophiles and sulphur dependent. Isolation of several extremely thermophilic sulphur metabolisers is covered in Section 3.2. The sulphur may be used either as an electron acceptor in anaerobic respiration or as an electron donor. They inhabit geothermal heated water or soils that contain elemental sulphur, *e.g.* the hot sulphur springs in Yellowstone National Park,

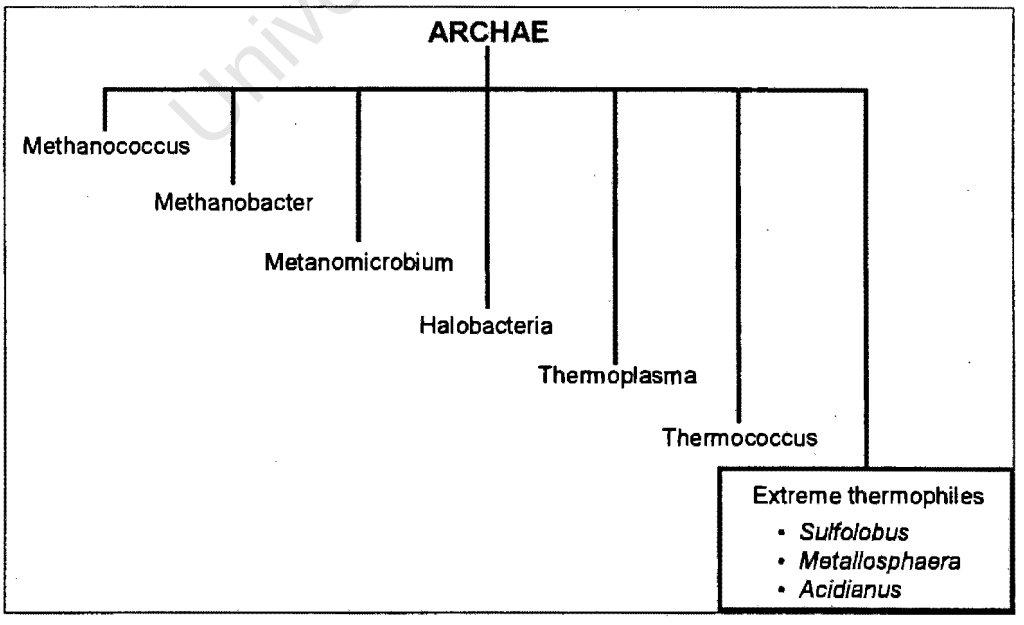


Figure 4-2: Relative grouping within the archae (Stanley et al., *1989).*

Wyoming. Both organotrophic or lithotrophic growth occurs in this group, where sulphur and H_2 are the most common electron sources. There are three orders within this group, *Thermococcales*, *Thermoproteales* and *Sulfolobales* (Stanley *et al.*, 1989).

Members of the genus *Sulfolobus* are gram-negative, aerobic, irregularly lobed spherical bacteria with a growth temperature optimum around 70-80°C and an optimum pH 2-3. Hence they are classified as thermoacidophilic (Prescott *et al.*, 1993). Their cell wall contains lipoprotein and carbohydrates, but lacks peptidoglycan (Berry and Murr, 1980; König and Stetter, 1986). They grow lithotrophically on sulphur granules (Weiss, 1973; Shivvers and Brock, 1973), ferrous iron and mineral sulphides (Karavaiko *et al.*, 1993; De Rosa and Gambacorta, 1975; Gómez, *et al.*, 1996). Oxygen is the normal electron acceptor, but ferric iron may also be used (Prescott *et al.*, 1993).

4.2 DISTINGUISHING FEATURES OF ARCHAE MICRO-ORGANISMS

The first thing to note when dealing with archae is that no aspect either of their morphology, physiology or biochemistry can be assumed to be similar to those of either eucaryotes or eubacteria. Furthermore, these assumptions cannot be made within the archae either. As suggested by Figure 4-1 and Figure 4-2, a large diversity exists amongst the archae. Subsequently it is expected that the differences between the archae and eubacteria or eucaryotes will not be unanimous within the archae. As a result, it is important to realise that any information presented may not be confirmed for the entire kingdom or even the order for that matter (Danson, 1988).

There are several features of the archae, which distinguish them from other microorganisms in general, these are listed below: (König and Stetter, 1986)

- differences in central metabolic pathways
- presence of glycerol isopranyl ether lipids in the cell membrane
- lack of murein in their cell walls (hence insensitive to β-lactam antibiotics such as penicillin)
- general cell envelope diversity resulting in either gram positive or negative staining
- no ribothmidine in the 'common arm' of the tRNA
- differences in 5S, 16S, 23S rRNA'S.

Several of these features will be discussed in further details.

4.2.1 CENTRAL METABOLIC PATHWAYS

It is accepted that the central metabolic pathways in all organisms involve sugar catabolism. A basic understanding of these systems in eucaryotes and eubacteria is essential prior to the description of the central metabolic pathways of the Archae (Danson, 1988).

4.2.1.i Glucose Catabolism

The pathways concerning glucose catabolism of the eucaryotes and eubacteria will be reviewed: that is the Embden-Meyerhof or glycolytic pathway; Entner-Doudorff pathway (with mention of the pentose-phosphate pathway) and the tricarboxylic acid (TCA) cycle. Thereafter the corresponding pathways of the archae will be contrasted.

As illustrated in Figure 4-3, glucose is phosphorylated in the first few steps to fructose 1,6-bisphosphate, which in turn undergoes an aldol cleavage to two interconvertible trioses; dihydroxyacetone phosphate and

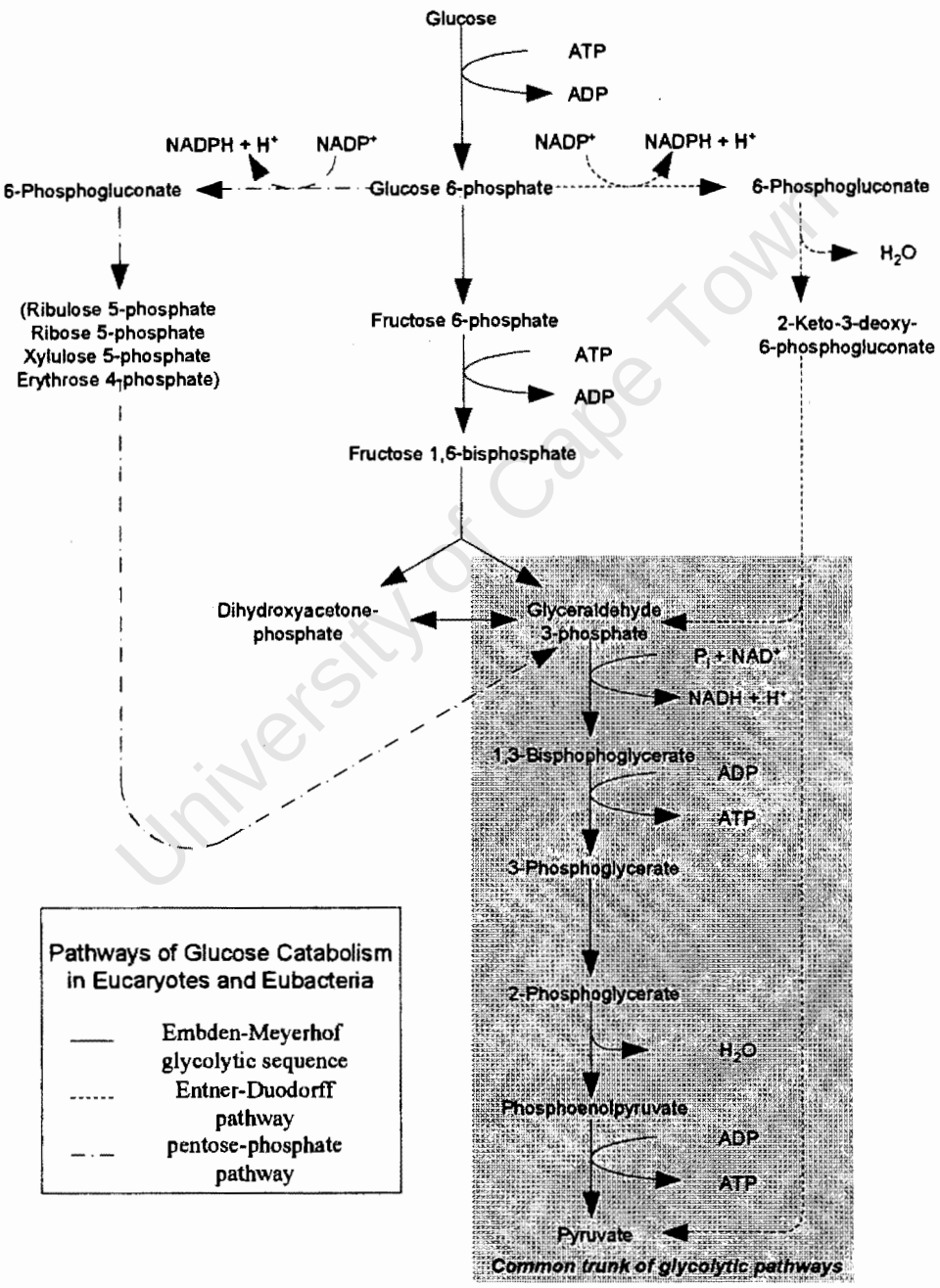


Figure 4-3: Eucaryotic and eubacterial glycolytic pathways (Danson, 1988).

glyceraldehyde 3-phosphate. The glyceraldehyde 3-phosphate is then converted to pyruvate as demonstrated (Danson, 1988).

The Entner-Doudorff pathway (Figure 4-3) can be considered as a divergence from the Embden-Meyerhof pathway as it only differs by several intermediates. Glucose 6-phosphate is oxidised to 6-phosphogluconate before being dehydrated to 2-keto-3-deoxy-6-phosphogluconate, which then undergoes aldol cleavage to yield glyceraldehyde 3-phosphate and pyruvate. At this stage the Entner-Doudorff pathway rejoins the Embden-Meyerhof pathway (Danson, 1988).

It is clearly evident that there is a 'common trunk' amongst glycolytic pathways, that is, the sequence from glyceraldehyde 3-phosphate to pyruvate is common to several glucose catabolic pathways. It has been postulated that this is in fact the true central pathway in glucose catabolism (Danson, 1988).

The tricarboxylic acid cycle (TCA) continues the metabolism of the carbohydrates. The most common fate of pyruvate is acetyl-CoA, via oxidative carboxylation, which in turn enters TCA cycle, and follows the general sequence as illustrated in Figure 4-4. The TCA cycle exhibits considerable diversity in the portions of the cycle which are used, as well as in the patterns of regulation and catalytic activity. This is particularly important in terms of carbon dioxide fixation or assimilation (Section 4.2.4).

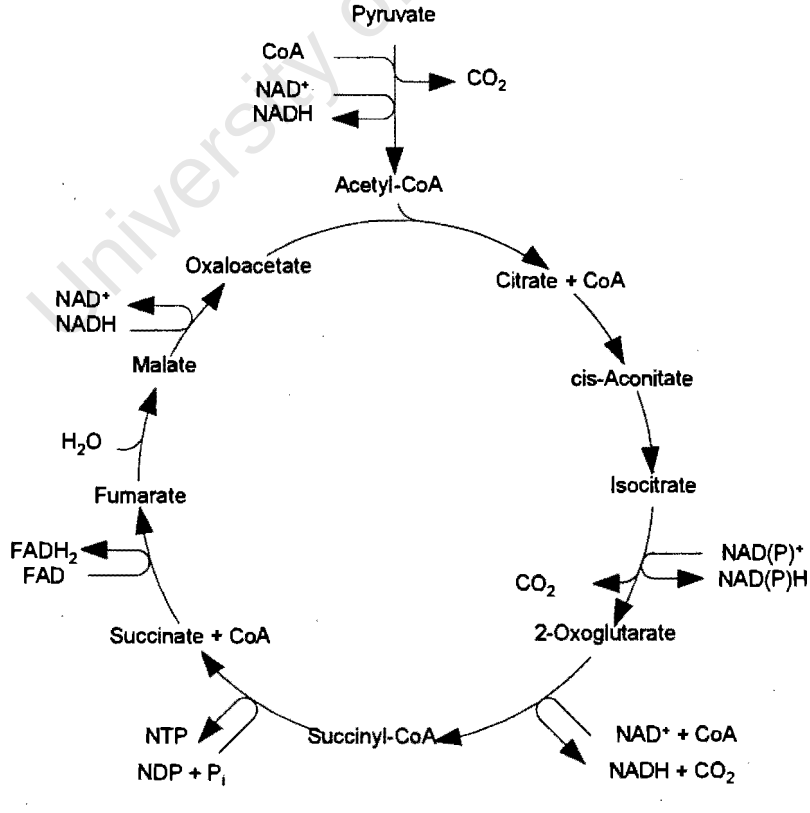


Figure 4-4: Tricarboxylic acid cycle (Bailey and Ollis, 1991).

4.2.1.ii Archae Hexose Catabolism

The archae can be divided into the following three main groups, from a phylogenetic and biochemical point of view (Danson, 1988):

- thermophilic and sulphur dependent, referred to as the thermoacidophiles
- methanogens and halophiles
- the order *Thermococcales.*

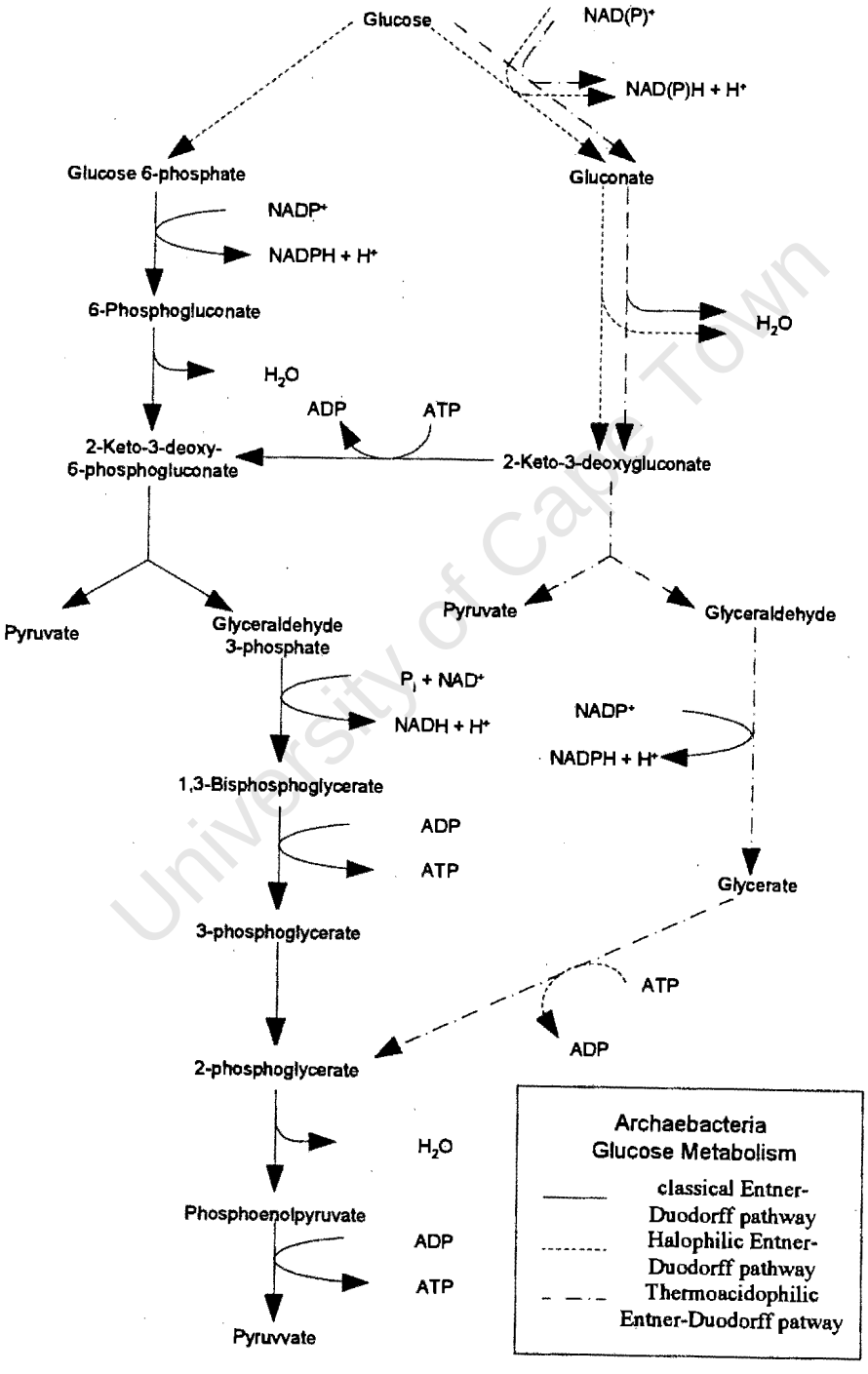


Figure 4-5: Archae modified Entner-Duodorff pathway (Danson, 1988).

The pathways will be divided into distinct categories and discussed and patterns identified. An important point is that archae have different nutrient sources. As is the case with eubacteria, some are lithotrophic, heterotrophic, chemolithotrophic, and autotrophic or even mixotrophic. Hexose catabolism is obviously important, but the origin of the hexoses may vary (Fewson, 1986). These will therefore be discussed and developed relative to halophiles and thermoacidophiles.

4.2.1.ii.a The Halophiles

Firstly, many of the halophiles utilise amino acids and proteins rather than carbohydrates. Where glucose is utilised, a modified Entner-Doudorff pathway is implemented, as illustrated in Figure 4-5. Glucose is initially oxidised to gluconate by $NAD(P)^+$, or by NAD^+ at a significantly reduced rate. Thereafter, it is dehydrated to 2-keto-3-deoxygluconate, where the halophile pathway rejoins the conventional glycolytic pathway. The significant differences are that $NAD(P)^+$ or NAD^+ are utilised from the start, thereby delaying the phosphorylation step, 2-keto-3-deoxy-6phosphogluconate is the first phosphorylated metabolite in the halophilic sequence. The significance is that there may not be an absolute requirement for ATP (Danson, 1988).

4.2.1.ii.b The Thermoacidophiles

Amongst these microorganisms, there is a further modification of the Entner-Doudorff pathway, the phosphorylation step is omitted all together in the formation of the first molecule of pyruvate. 2-Keto-3-deoxygluconate is cleaved directly to pyruvate and glyceraldehyde which is then oxidised to glycerate by an $NAD(P)^+$ dependent dehydrogenase. Glycerate is then converted into 2-phosphoglycerate, which then follows the remainder of the 'common trunk' generating the second molecule of pyruvate. Even at this point, there is no net yield of ATP, although this is not the pathway that contributes the bulk of the energy yield (Danson, 1988).

In conventional glycolysis, 6-phosphofructokinase is the enzyme which catalyses the phosphorylation of glucose and this is the rate-limiting step. There is no evidence of this enzyme in *Sulfolobus solfataricus* microorganisms, although they readily grow on glucose as their sole carbon source. This alludes to the fact that ATP may not be an absolute requirement (Danson, 1988). Because of their acidophilic nature, *Sulfolobus* species require active proton extrusion for the regulation of internal pH (Lübben and Schäfer, 1989). Hence they depend energetically on respiration-coupled oxidative phosphorylation. Further, a plasma-membrane associated ATPase has been identified as a proton-translocating agent (Vitaya and Toda, 1991).

This first hypothesis involves a transport mechanism that requires the carrier undergo a conformational change. Energy released by electron transport induces changes in the shape or conformation of the enzyme that synthesises ATP. These changes drive the ATP formation. The chemiosmotic hypothesis supposes that the proton concentration gradient across the inner membrane is created directly by the electron transferred across the membrane. Therefore the high-energy state is the proton gradient, which represents a difference not only in concentration but also in electrical potential across the membrane. ATP is generated when the protons flow back into the mitochondria (eukaryotes) or across the cell membrane (prokaryotes) down the concentration

gradient (Ingledew, 1990). The application of this hypothesis to *Sulfolobus* respiration will be discussed in Section 4.2.3.

4.2.2 CELL ENVELOPE DIVERSITY

There is considerable structural and chemical diversity in the cell envelope within the archae. As mentioned previously, the cell envelope lacks a peptidoglycan layer and can exhibit a positive or negative gram stain. Conventionally a gram-positive stain implies that murein or, in this case a pseudomurein substance exists (Prescott *et al.*, 1993). Cells that show negative-staining cells do not possess this thick outer layer. These cells have a surface layer of (glyco-) protein subunits, as is the case with *Sulfolobus* spp. (König and Stetter, 1989; Prescott *et al.*, 1993; Inatomi *et al.*, 1983).

One of the most distinctive features of archae is the cell membrane lipid structure. They differ from other microorganisms in that they have branched hydrocarbons attached to glycerol by ether links as opposed to fatty acids connected by ester links. The result is the formation of rather long (usually containing 40 carbons) tetraethers. These C40 tetraethers result in the formation of a very rigid monolayer, which resembles a bilayer membrane in structure, helping to increase the stability of the membranes of the extreme thermophiles (Prescott, *et al.*, 1993).

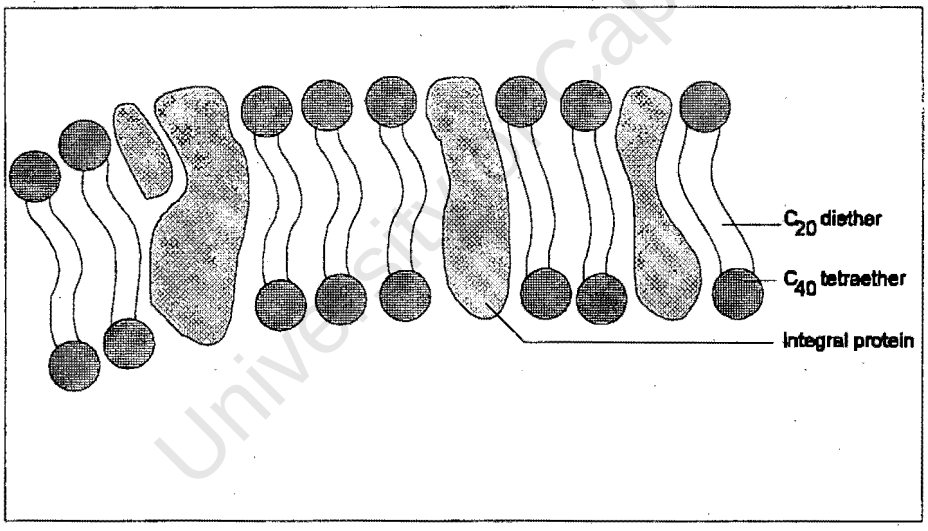


Figure 4-6: Cell membrane molecular structure (Prescott, et al., 1993).

Enzymes and carrier molecules can be embedded within the membrane as integral proteins, in a similar way to the phospholipid bilayer, as illustrated in Figure 4-6 (Prescott *et al.*, 1993). These molecules are also known to agglomerate to form ports, or carrier complexes, which may be involved in conformational changes to bring about active transport across the membrane.

All *Sulfolobus* spp. and other extreme thermophilic groups possess cell envelopes composed of single surface layers (König, 1988). The thickness of these layers varies from 20 to 40 nm (Inatomi *et al.*, 1983; König and Stetter, 1989). The negative Gram reaction confirms that these cells lack an outer layer of pseudo-murein, as do

some methanogens (König and Stetter, 1989). The surface layer of *Sulfolobus acidocaldarius* is composed of a hexagonal array of glycoprotein, with regular holes of diameter 4.5 to 5 nm and a centre to centre distance of 20nm (Michel *et al.*, 1980; Inatomi *et al.*, 1983; König, 1988). While being rigidly structured, this surface layer is very porous, resembling a sponge with channels running through it, (Figure 4-7). Figure 4-8 displays a

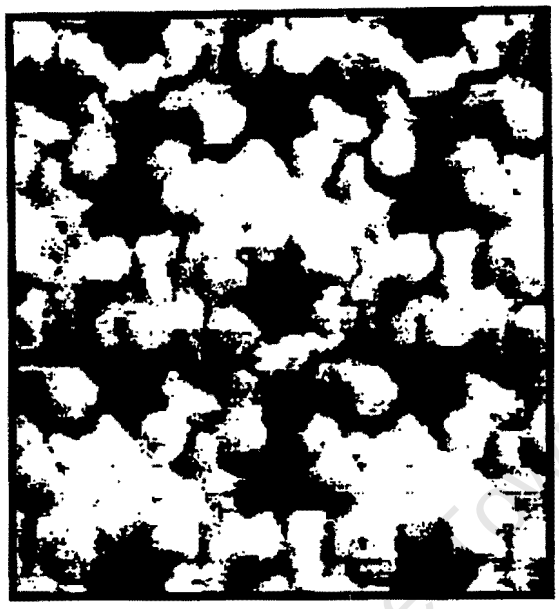


Figure 4-7: Surface reconstruction of Sulfolobus solfataricus *cells (Konig, 1988).*

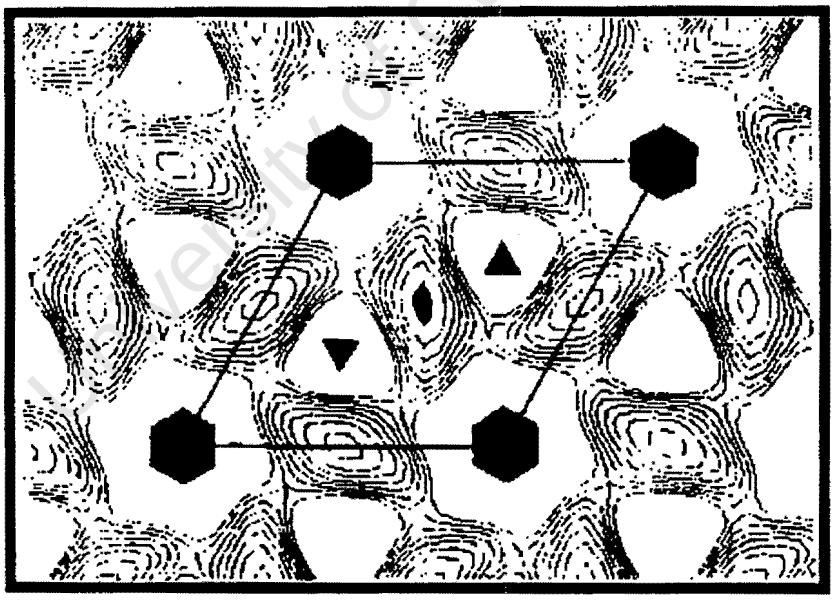


Figure 4-8: Computer representation of the cell envelope surface layer (Michel et al., *1980.).*

computer-generated representation of this surface layer cell membrane (Michel *et al.*, 1980). The surface proteins create a highly negatively charged layer. The low pH (in the region of 2) results in a high concentration of hydrogen ions which stabilises the cell envelope by neutralising this layer (Berry and Murr, 1980).

Hence, the cell wall of *Sulfolobus* spp. lends itself to being rigid for support and porous to facilitate transport of necessary nutrients through to the cell membrane. It forms an outer layer of hydrophobic amino acids which may be related to attachment (Michel *et al.*, 1980; van Loosdrecht *et al.*, 1987; Weiss, 1973).

4.2.2.i Attachment

Several studies have been conducted to determine the ease of attachment of *Sulfolobus* spp. onto different substances. It is postulated that microorganisms can be both attached and in suspension to carry out the leaching (Section 2.5). *Sulfolobus* spp. readily become attached to sulphur particles (Weiss, 1973), pyrite and other mineral sulphides (Huber *et al.* 1989, Konishi *et al.*, 1995; Murr and Berry, 1976; Chen and Skidmore, 1988; Brierley *et al.*, 1980; Vitaya and Toda, 1991), coal (Vitaya and Toda, 1991; Chen and Skidmore, 1988) or activated carbon with a high surface area (Chen and Skidmore, 1988).

Attachment on to ore or immobilisation particles, may be an effective means to counteract susceptibility to shear (Section 3.3.2). Iglesias and Caranza (1996) suggest a 'separate generator' process where the cells are immobilised and used to generate ferric iron, which is recycled to achieve leaching of mineral sulphides. However, Larsson *et al.* (1993) give evidence that physical contact between pyrite and the *Acidianus brierleyi* is the most favourable condition for good microbial growth and fast pyrite oxidation. It is possible that the microorganisms are growing on other substrates apart from ferrous iron in the presence of mineral sulphides.

Brock *et al.* (1972) reported the attachment of *Sulfolobus acidocaldarius* to sulphur crystals. Shivvers and Brock (1973) examined *Sulfolobus* attachment using fluorescent staining. Results showed that during the initial exponential growth phase there was a 1:1 relationship between attached and planktonic cells. As time passed the relative number of attached cells increased. After 17 days of growth on sulphur crystals, there was a 10:1 relationship. Weiss (1973) reported that attachment is not a prerequisite for sulphur oxidation by the organisms in culture, but attachment to sulphur in flowing springs enables *Sulfolobus* to colonise these areas. Hence, sulphur is a good growth substrate for *Sulfolobus* spp. If sulphur is a by-product of bioleaching, then *Sulfolobus* spp. can be used to 'mop' this up to eliminate or reduce the problem of increased cyanide consumption in the cyanidation step (Norris, 1997).

Huber *et al.* (1989) isolated *Metallosphaera sedula* organisms. They observed 50% attachment of cells to mineral sulphide particles in experiments. Konishi *et al.* (1995) found that the quotient of free cells to adsorbed cells increased linearly with free cell concentration. At a free cell concentration of $5x10^{14}$ cells/m^3 the ratio of free to absorbed cells is 10:1. This this information was used to correct errors in cell enumeration owing to increased adsorption.

Chen and Skidmore (1988) observed that the extent and rate of attachment varied considerably with the type of particle. A high selectivity for cell attachment to pyritic surfaces within coal was noted (Chen and Skidmore, 1988; Vitaya and Toda, 1991). Desorption studies showed that an irreversible attachment developed through longer particle contact (Chen and Skidmore, 1988). At this stage it is difficult to make comparisons with

Thiobacillus ferrooxidans and *T. thiooxidans* owing to inconsistent agitation effects and coal types reported in the *Sulfolobus* work.

Vitaya and Toda (1991) proposed that adsorption occurred in two stages: a rapid initial stage of adsorption controlled by the force of hydrophobicity, followed by desorption to achieve dynamic equilibrium between adsorbed and desorbed cells. This desorption or equilibrium stage was physiologically controlled, but the dominant contributing factor is still unknown. It was suggested that hydrodynamic conditions created by shaking may contribute to desorption. Two other effects investigated were the effect of the electrical properties (zeta potentials) of the coal and cell surfaces as well as the inhibition of the cell membrane-bound ATPase using sodium fluoride (NaF) (Vitaya and Toda, 1991). Vitaya and Toda (1991) did not concur with Chen and Skidmore's (1988) observation of irreversible attachment.

Chen and Skidmore (1988) proposed that the surface condition of the particles could play an important role in attachment after short-term contact with the cells. Therefore, information regarding the surface properties of both the microbial cell wall and solid particles is crucial to further understanding of attachment. This includes factors such as total external surface area, surface charge, hydrophobicity and surface appendages to the cell wall (Chen and Skidmore, 1988; Konishi *et al.*, 1995).

4.2.3 *SULFOLOBUS* RESPIRATION AND HOW IT RELATES TO ADSORPTION

Sulfolobus grows in an acidic environment, optimally pH 2-3. As a result, it needs to be able to maintain a high pH gradient, with an intracellular pH of 5.6 (Lübben and Schäfer, 1989). This is energetically linked to cellular respiration. Experimentation has found that endogenous respiration does occur in *Sulfolobus acidocaldarius*, and it can be enhanced by supplementing metabolic intermediates such as succinate, isocitrate and fumarate. However, it is unlikely that this occurs to the exclusion of oxidative phosphorylation, and it is most likely that these are coupled, with a greater emphasis on oxidative phosphorylation (Anemüller *et al.*, 1985; Lübben and Schäfer, 1989).

Further experimentation has shown that the electrochemical activity gradient for protons, Δp, is defined as follows:

$$\Delta p = \Delta\Psi - 2.3 \cdot \left(\frac{RT}{F}\right) \cdot \Delta pH \qquad \textit{Equation 4-1}$$

where R is the gas constant, T the absolute temperature, F is the Faraday constant. The first term ($\Delta\Psi$) is the potential resulting from the membrane potential or the electrical potential difference across the membrane (mV), the second term (ΔpH) is the potential resulting from the pH difference (mV). It is postulated that the potential developed by the chemiosmotic process can be used to drive cellular activities such as synthesis of extracellular slimes and ATP required for adsorption onto mineral or coal surfaces (Vitaya and Toda, 1991b).

4.2.4 CARBON DIOXIDE FIXATION AND ASSIMILATION

Many strains of *Sulfolobus* are facultative autotrophs, which can grow autotrophically with carbon dioxide as their only source of carbon, or heterotrophically on some organic carbon source (Norris, 1989). A factor that indirectly affects mineral oxidation by microbes and which may be controlled in active bioleaching processes is the availability of carbon dioxide. As with iron oxidation, the efficiency of carbon dioxide assimilation by various mineral-oxidising acidophiles might be expected to differ given the diversity of the organisms (Norris, 1989).

Norris (1989) conducted work with *Thiobacillus ferrooxidans*, *Sulfolobus acidocaldarius* and *S.* BC strain in the presence of pyrite (40% (w/w) iron) in 500ml airlift reactors with a central draught tube. It was observed that the carbon dioxide limitation of the activity of the thermophiles was largely alleviated with only 0.1% (v/v) CO_2 in air. It was concluded that the use of carbon dioxide enriched air would be required to maintain the optimum microbial activity, particularly with the extreme and moderate thermophiles (Norris, 1989).

Shivvers and Brock (1973) conducted work on *Sulfolobus acidocaldarius* strains isolated from Yellowstone National Park (USA) and Italy. It was shown that these strains required carbon dioxide supplementation for optimum growth rates. They used intermittent 5% (w/v) carbon dioxide-enriched air sparging. When the carbon dioxide was delivered continuously, sulphur oxidation ceased (Shivvers and Brock, 1973). Using $^{14}CO_2$ with *Thiobacillus* spp it has been observed that the main part of carbon fixed from $^{14}CO_2$ (about 50-80%) is incorporated into bacterial cells while the rest (20-50%) is released into the medium as organic substrates (Karavaiko and Lobyreva, 1994). Hence it is possible that excess carbon dioxide results in organic carbon,which serves as a more suitable substrate for *Sulfolobus* spp. Shivvers and Brock (1973) concluded that carbon was growth limiting for *Sulfolobus acidocaldarius.* When yeast extract was made available, growth was greatly stimulated and the total quantity of oxidised sulphur was reduced. The rate of sulphur oxidation is inhibited up to 30-fold by the presence of yeast extract. They attribute this to the fact that the organic supplement to the growth medium suppresses the enzymes in the sulphur oxidation pathway (Shivvers and Brock, 1973).

Carbon dioxide is assimilated by *T. ferrooxidans*, primarily via the Calvin reductive pentose phosphate pathway, as with many other autotrophic eubacteria. The enzyme catalysing the carbon dioxide fixation, ribulose-1,5-bisphosphate carboxylase/oxygenase, has been purified from *T. ferrooxidans.* Activity of the same enzyme has been found in the moderately thermophilic iron-oxidising organisms. However, in comparison with *T. ferrooxidans*, the autotrophic growth of aerated cultures of the moderate thermophiles is severely restricted unless the carbon dioxide concentration in the gas supply is increased above that in air. In the absence of such carbon dioxide enrichment, the rate of pyrite oxidation during cell growth of these organisms is found to be reduced (Norris, 1990).

Sulfolobus acidocaldarius does not lack any specific enzymes that would preclude the natural pathway of carbon dioxide assimilation to take place, particularly the enzymes and intermediates involved in the TCA cycle (Danson, 1988). However there is little evidence of a large-scale oxidative TCA metabolism. Carbon dioxide

fixation is presumed to occur through the carboxylation of pyruvate and phosphoenolpyruvate (Danson, 1988). Further evidence of this is the existence of both pyruvate and phosphoenol carboxylase enzymes in several different strands of *Sulfolobus* (Wood *et.al.*, 1987). Both carbon dioxide fixation and assimilation are known to occur concurrently. However, it is known that heterotrophic growth represses carbon dioxide fixation and sulphur (or ferrous) oxidation (Kargi and Robinson, 1985; Shiverrs & Brock, 1973; Wood *et. al.*, 1987; Brierley and Brierley, 1986). This evidence does not contradict the proposed reductive carboxylic acid pathway. *Sulfolobus* spp. have been shown to give faster growth and sulphur oxidation under increased partial pressures of carbon dioxide (Shivvers and Brock, 1973).

The methods of carbon dioxide assimilation by *Acidianus brierleyi* were determined to be different for heterotrophic and autotrophic metabolism. For heterotrophic metabolism, carbon dioxide fixation occurs by an exchange between the α-carboxyl group of α-ketoglutaric acid and carbon dioxide. The autotrophic fixation occurs via a reductive carboxylic acid pathway. The Calvin-Benson cycle is apparently not present in these extreme thermophiles (Brierley and Brierley, 1986).

The pathway of carbon dioxide fixation by *Sulfolobus* and other extreme thermophiles has not been resolved but a reductive carboxylic acid cycle (Stetter, 1986) and a possible role for an acetyl CoA carboxylase have been suggested (Norris, 1989). The capacity of *Sulfolobus* to respond to a limiting carbon dioxide concentration appeared to reside in the increased production of an enzyme catalysing the carboxylation of acetyl coA. The Calvin cycle does not operate in *Sulfolobus* and archaebacteria in general (Norris, 1990). Previous demonstrations of the rapid and efficient oxidation of finely-ground mineral sulphides in the absence of organic nutrients have generally involved gassing with 1-5% (v/v) CO_2 in air (Norris and Parrot, 1986; Le Roux and Wakerley, 1988; Marsh and Norris, 1983).

4.3 REFERENCES

Anemüller, S., M. Lübben, and G. Schäfer (1985): " The respiratory system of *Sulfolobus acidocaldarius*, a thermoacidophilic archaebacterium." *FEBS Letters.* **193** (1), 83-87.

Berry, V.K. and L.E. Murr (1980): "Morphological and ultrastructural study of the cell envelope of thermophilic and acidophilic microorganisms as compared to *Thiobacillus ferrooxidans*", *Biotech. Bioeng.*, **22**, 2543-2555

Brierley, J.A. and C.L. Brierley (1986): " Microbial mining using thermophilic microorganisms", John Wiley & Sons, Canada, p.279-305

Brierley, C.L., J.A. Brierley, P.R. Norris, and D.P. Kelly (1980): " Metal-tolerant micro-organisms of hot, acid environments." Academic Press, New York, p.39-51

Brock T.D., K.M. Brock, R.T. Belly and R.L. Weiss (1972): "*Sulfolobus*: A new genus of sulfur-oxidizing bacteria living at low pH and high temperature." *Arch. Mikrobiol.* **84**, 54-68.

Chen, C. and D.R. Skidmore (1988): "Attachment of *Sulfolobus acidocaldarius* cells on coal particles." *Biotech. Prog.* **4**, 25-30.

Danson, M.J. (1988): " Archaebacteria: The comparative enzymology of their central metabolic pathways." *Advances in Microbial Physiology.* **29**, 165-231

De Rosa, M. and A. Gambacorta (1975): "Extremely thermophilic acidophilc bacteria convergent with *Sulfolobus acidocaldarius*", *J. Ind. Microbiology*, **86**, 156-164

Fewson, C.A. (1986): "Archaebacteria". *Biochemical Education*, **14**, 103-115.

Gómez, E., M.L. Blázquez, A. Ballester and F. González (1996): "Study by SEM and EDS of chalcopyrite bioleaching using a new thermophilic bacteria", *Minerals Engineering*, **9**(9), 985-999

Huber, G., C. Spinnler, A. Gambacorta and K.O. Stetter (1989): "*Metallosphaera sedula* gen. and sp. nov. represents a new genus of aerobic, metal mobilising, thermoacidophilic archaebacteria", *System. Appl. Microbiol.* **12** 38-47

Iglesias, N. and F. Carranza (1996): "Treatment of a gold bearing arsenopyrite concentrate by ferric sulphate leaching", *Minerals Engineering*, **9**(3), 317-330

Inatomi, K., M. Ohba and T. Oshima (1983): "Chemical properties of proteinaceous cell wall from an acido-therophile, *Sulfolobus acidocaldarius*", *Chemistry Letters*, Chemical Society of Japan, 1191-1194.

Ingledew, W.J. (1990): "Acidophiles", in *Microbiology of extreme environments*, Ed. C. Edwards, Open University Press, Milton Keynes, Chapter 2, p. 33-54

Karavaiko, G.I. and L.B. Lobyreva (1994): "An overview of the bacteria and archaea involved in removal of inorganic and organic sulphur compounds from coal", *Fuel Processing Technology*, **40**, 167-182

Karavaiko, G.I., R.S. Golovacheva, O.V. Golyshina, K.M Valiejo-Roman, V.K. Bobrova, A.V. Troitsky and T.A. Pivovarova (1993): "*Sulfurococcus*- a new genus of thermoacidophilc archae-bacteria oxidising sulphur, ferrous iron and sulphide minerals", in "*Biohydrometallurgical Technologies*", edited by A.E. Torma, J.E. Wey and V.L. Lakshmanan, The Minerals, Metals & Materials Society, p 685-694

Kargi, F and K.M. Robinson (1985): "Biological removal of pyritic sulfur from coal by the thermophilic organism *Sulfolobus acidocaldarius*" *Biotech. Bioeng.* **27**, 41-491

König, H. (1988): "Archaebacterial cell envelopes", *Can. J. Micro.* **34**, 395-406.

König, H. and O. Stetter (1986): "Studies on Archaebacterial S-layers." *System. Appl. Microbiol.* **7**, 300-309.

Konishi, Y., S. Yoshida and S. Asai (1995): "Bioleaching of pyrite by acidophilic thermophile *Acidianus brierleyi*", *Biotech. Bioeng.*, **48**, 592-600

Larsson, L., G. Olsson, O. Holst and H.T. Karlsson (1993): "Oxidation of pyrite by *Acidianus brierleyi*: Importance of close contact between the pyrite and the microorganisms" *Biotech. Letters*, **15**(1), 99-104.

Le Roux N.W. and D.S Wakerley (1988): "Leaching of chalcopyrite ($CuFeS_2$) at 70°C using *Sulfolobus*", in *Biohydrometallurgy '87*, Eds P.R. Norris and D.P. Kelly, Science and Technology Letters, Surrey, United Kingdom, p 305-317

Lübben, M. and G. Schäfer (1989): "Chemiosmotic energy conversion of the Archaebacterial thermoacidophile Sulfolobus acidocaldarius: Oxidative phosphorylation and the presence of an F0- Related N,N'-Dicyclohexylcarbodiimide-binding proteolipid." *J. Bact.* **171**, 6106-6116

Marsh, R.M and P.R. Norris (1983): "Mineral sulphide oxidation by moderately thermophilic acidophilic bacteria", *Biotech. Letters*, **5**(9), 585-590

Michel, H., D.-C. Neugebauer and D. Osterhelt (1980): "The 2-D crystalline cell wall of *Sulfolobus acidocaldarius*: Structure, solubilization and reassembly.", in *Electron microscopy at molecular dimension"*, Eds W. Baumeister and W. Vogell, Berlin. Springer-Verlag, p27-35

Murr, L.E. and V. K. Berry (1976): "Direct observation of selective attachment of bacteria on low-grade sulfide ores and other mineral surfaces", *Hydrometallurgy*, **2**, 11-24

Norris (1997): Personal Communication, University of Warwick, United Kingdom

Norris, P.R. (1990): "Acidophilic bacteria and their activity in mineral sulphide oxidation", in *Microbial Mineral Recovery*, Eds H.L. Ehrlich and C.L. Brierley, McGraw-Hill, New York, p 3-27

Norris, P.R. (1989): "Factors affecting bacterial mineral oxidation: The example of carbon dioxide in the context of bacterial diversity", in *Biohydrometallurgy 1989*, Eds J Salley, R.G.L. McCready and P.L. Wichlacz, Jackson-Hole Wyoming, p 3-14

Norris, P.R. and L. Parrot (1986): "High temperature, mineral concentrate dissolution with *Sulfolobus*", in *Fundamental and Applied Biohydrometallurgy*, Elsevier, Amsterdam, Netherlands, p 355-365

Prescott, L.M., J.P. Harley and D.A. Klein (1993): "Microbiology 2nd edition", Wm. C. Brown Communications Inc., USA, p.492-505.

Shivvers, D.W. and T.D. Brock (1973): "Oxidation of elemental sulfur by *Sulfolobus acidocaldarius*". *J. Bact.* **114**, 706-710.

Stanley, J.T., Williams and Wilkins *et al.*, (1989): "Bergey's manual of systematic Bacteriology Volume 3, Baltimore, USA

Stetter, K.O. (1986): " Diversity of extremely thermophilic Archaebacteria", John Wiley & Sons, Canada, p.39-74.

van Loosdrecht, M.C.M., J. Lyklema, W. Norde, G. Schraa and A.J.B. Zehnder (1987): "The role of bacterial cell wall hydrophobicity in adhesion" *Appl. Env. Micro.* **53**, 1893-1897.

Vitaya, V.B. and K. Toda (1991): "Physiological adsorption of *Sulfolobus acidocaldarius* on coal surfaces". *Appl. Micro.Biotech.* **35**, 690-695

Vitaya, V.B. and K. Toda (1991b): "Kinetics and mechanism of the adsorption of *Sulfolobus acidocaldarius* on coal surfaces" *Biotech. Prog.* 7, 427-433

Weiss, R.L. (1973): "Attachment of bacteria to sulphur in extreme environments", *J. Gen. Microbiology*, 77, 501-507

Woese, C.R. (1981): "Archaebacteria". *Scientific American.* **244**, 94-106.

Woese, C.R. (1987): "Bacterial Evolution". *Microbiological Reviews.* **51**, 221-271.

Wood, A.P., D.P. Kelly and P.R. Norris (1987): "Autotrophic growth of four *Sulfolobus* strains on tetrathionate and the effect of organic nutrients". *Arch. Mikrobiol.* **146**, 382-389.

Part III

the Effect of Temperature on the Bioleaching Environment

5. Gas-Liquid Mass Transfer

In the biooxidation of sulphide minerals, the important mass transfer steps include the transfer of nutrients, including oxygen, carbon dioxide and ferrous iron to the cells as well as the transfer of ferric iron to the mineral. When considering thermophilic bio-oxidation, it is necessary to understand the effect of temperature on mass transfer. This also applies to other process such as, *e.g.* chemical bioleaching and thermophilic biodesulphurisation of coal. This chapter deals with the theory behind gas-liquid mass transfer.

Gas-liquid mass transfer itself is comprised of three contributions, the transfer coefficient (k_L), the effective area (a) and the concentration gradient, (C^*-C), also referred to as the driving force. The transfer coefficient is strongly dependent on the diffusion coefficient, which is in turn dependent upon the temperature and viscosity of the solvent. Moreover, the viscosity is also a function of temperature. The effective interfacial area is the ratio of surface area to volume of gas. It is dependent on the bubble size distribution. The effect of temperature on the interfacial area is mediated primarily through its effect on surface tension. The driving force is strongly dependent on the saturation concentration (C^*) which is a function of the solubility of the gas in the solvent. It is well known that temperature and the presence of solutes affect this solubility.

5.1 RESISTANCE TO MASS TRANSFER

An overview of gas-liquid mass transfer in a biochemical process is illustrated in Figure 5-1. There are several steps involved in the transfer of oxygen from a gas bubble to the reaction site inside the individual cells (Nielsen and Villadsen, 1994). These include:

1. Diffusion of oxygen from the bulk to the gas-liquid interface.
2. Transport across the gas-liquid interface.
3. Diffusion of oxygen from the gas-liquid interface to the well-mixed bulk liquid, across the relatively stagnant film.
4. Transport of oxygen through the well-mixed liquid to the liquid film surrounding the cells, or the reaction site.
5. Diffusion through the stagnant liquid film.
6. Transport from the film to the cell, or reaction site.

For most processes, there is only one rate-controlling step (Moo-Young and Blanch, 1981). As oxygen solubility is low, transport is controlled by transfer from the interfacial layer to the bulk liquid, Step 4, in unicellular systems (Winkler, 1981).

5.2 DIFFUSION AND TRANSFER COEFFICIENTS

Rates of transport processes are generally defined in terms of a resistance and a driving force, as demonstrated by (Welty *et al.*, 1984):

$$\text{rate of mass transfer} = \frac{\text{driving force}}{\text{resistance}} \qquad \textit{Equation 5-1}$$

Fick's law of diffusion defines the diffusion coefficient, D_{AB}, where A and B are the solute and solvent respectively. Equation 5-2 shows the relationship between J_A, the mass flux of component A for unidirectional, steady state diffusion of solute A through a stagnant medium where D_{AB} is a function of both solute and solvent properties (Oolman, 1984). Considering the solution of A into water, the liquid phase diffusion coefficient D_L represents D_{AB},.

$$J_A = D_{AB}\frac{dC_A}{dx} \qquad \textit{Equation 5-2}$$

where

J_A	mass flux of component A in B
D_{AB}	diffusion coefficient of solute A in solvent B
C_A	concentration of A in the liquid
x	a measure of displacement at the interface
$\frac{dC_A}{dx}$	concentration gradient

When conditions are not stagnant, Fick's law cannot be used as a general equation for predicting mass transfer. The effect of fluid motion on mass transfer is incorporated in the mass transfer coefficient, k_L. The relationship between the mass flux of solute into the liquid phase, J_A, and the concentration of the solute is shown in Equation 5-3. Assuming that the concentration gradient can be represented by the difference in concentration, $(C^*\text{-}C)$ is the

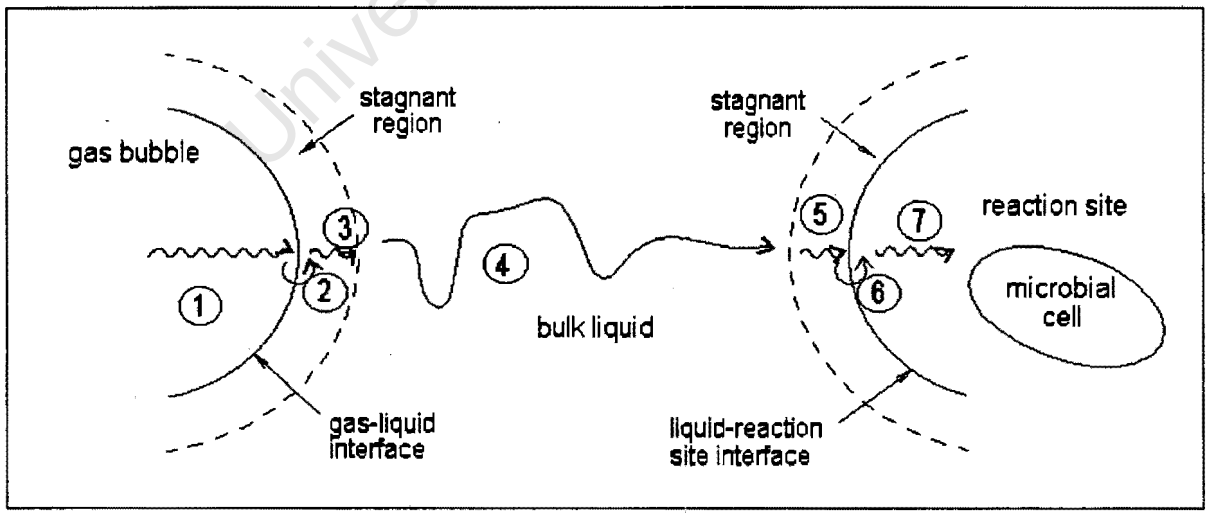


Figure 5-1: Schematic diagram of steps in transport of a gaseous component to a reaction site or microbial cells (Bailey and Ollis, 1986).

transfer driving force, where C^* and C are the concentration of solute at the interface and in the bulk liquid. The solute concentration at the interface is determined by the solubility of the diffusing gas in the liquid media (Nielsen and Villadsen, 1994).

$$J_A = k_L \cdot (C^* - C) \qquad \textit{Equation 5-3}$$

The mass transfer rate r_A of compound A is determined from the product of the flux and the area over which it operates (Equation 5-4). The gas-liquid interfacial area per unit liquid volume, a, is used as the characteristic area (Winkler, 1981).

$$r_A = k_L a \cdot (C^* - C) \qquad \textit{Equation 5-4}$$

Thus the rate of mass transfer is influenced by three domains: the transfer coefficient, interfacial area and the driving force. The effect of temperature on all three domains will be examined in order to elucidate the effect of temperature on mass transfer as a whole.

Based on the two-film approach (Nielsen and Villadsen, 1994), resistance to mass transfer is represented by:

$$\frac{1}{K} = \frac{1}{H \cdot k_G} + \frac{1}{k_L} \qquad \textit{Equation 5-5}$$

where

K	overall transfer coefficient
H	Henry's constant
k_G	gas side transfer coefficient
k_L	liquid side transfer coefficient

For sparingly soluble gases that are characterised by a high Henry's law constant (Section 5.7.), the controlling resistance to mass transfer occurs in the liquid phase. Thus, $(\frac{1}{H \cdot k_G})$ is insignificant and can be neglected and K can be estimated accurately using k_L (Nielsen and Villadsen, 1994).

5.3 SURFACE MOBILITY

Before discussing the analytical models used to describe the mass transfer coefficient, it is necessary to understand the concept of surface mobility. A fluid particle (gas or liquid), moving relative to a continuous liquid phase has two possible extremes of interfacial movement (Moo-Young and Blanch, 1981):

a) No internal circulation within the particle, *i.e.* the particle behaves essentially as if they are solid with rigid surfaces.
b) Circulation within the particle is fully developed due to an interfacial velocity. The surface is mobile and therefore renewable.

The mobility of the interface around a gas bubble varies with the composition and hydrodynamics of the liquid phase. For a bubble rising through a single component liquid, the interface will flow freely and internal circulation within the bubble will develop. Such bubbles are said to have mobile surfaces and are referred to as 'clean' (Oolman. 1984, Moo-Young and Blanch, 1981). When a second surface active component accumulates at the interface, the mobility of the bubble is restricted. A sufficiently high concentration of this surfactant can totally immobilise the surface and stagnate the gas inside the bubble. Bubbles with immobile surfaces are referred to as 'dirty' bubbles (Oolman, 1984). In bioleaching the bubbles tend to be small immobile bubbles. Examples of velocity profiles for mobile and immobile surfaces are illustrated in Figure 5-2 (Moo-Young and Blanch, 1981).

5.4 THEORETICAL MODELS FOR THE MASS TRANSFER COEFFICIENT (K_L)

Overall mass transfer is not only a strong function of the solute and solvent properties, but also of the prevailing hydrodynamic conditions. Hence, mass transfer coefficient models are generally restricted to the system from which they have been derived. These models illustrate how the physical properties of the liquid influence the liquid mass transfer coefficient (Nielsen and Villadsen, 1994). This is valuable as the influence of temperature on mass transfer can be considered in terms of the effect of temperature on these physical properties. The models and correlations presented illustrate the relative values of mass transfer coefficients as a function of diffusivities of the compounds which are in turn a function of temperature (Nielsen and Villadsen, 1994).

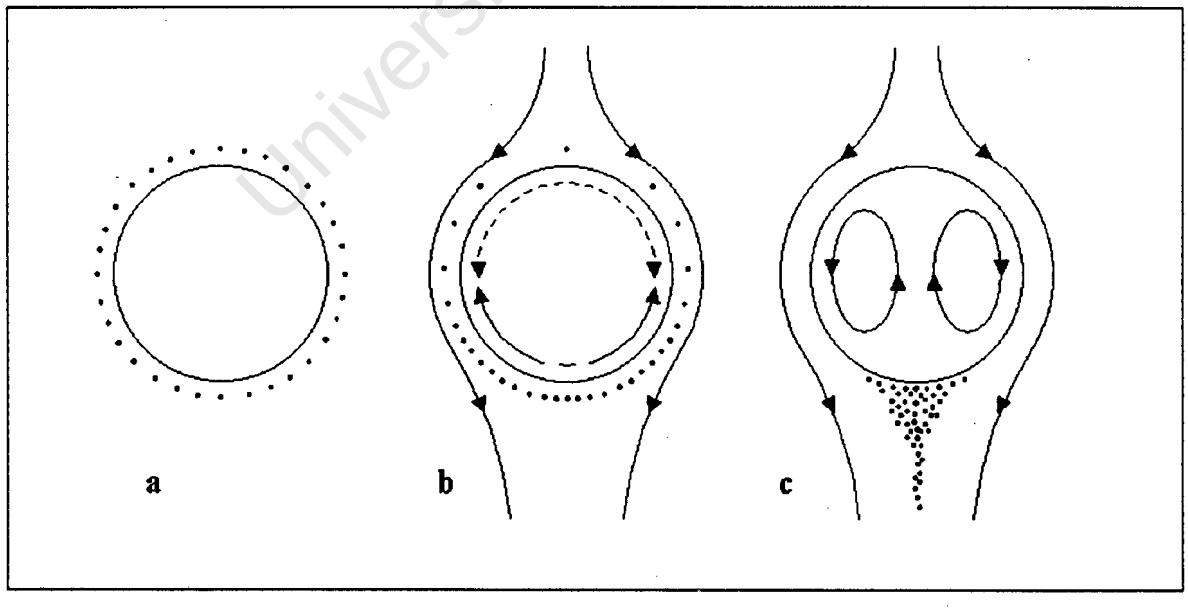


Figure 5-2: Surfactant effects on bubble/drop surface-flow at (a) zero (b) low (c) high, relative particle velocities (Moo-Young and Blanch, 1981).

5.4.1 k_L FROM FILM THEORY

The film theory of Whitman is based on diffusion through a stagnant film, or laminar region, which surrounds the interface. The major weakness of the film theory is the assumption of a film of uniform thickness (dependent on hydrodynamic conditions) and of a constant concentration gradient.

From Fick's first law (Equation 5-2) the mass transfer coefficient in a stagnant film of thickness δ with a constant concentration gradient is given by:

$$k_L = \frac{D_L}{\delta} \qquad \textit{Equation 5-6}$$

Boundary layer theory can be applied to rigid non-slip or immobile surfaces. Hence the film thickness is a function of the viscosity, density and the diffusivity of the solute A:

$$\delta \propto \left(\frac{\mu}{\rho \cdot D_L}\right)^{-\frac{1}{3}} \qquad \textit{Equation 5-7}$$

where

δ stagnant film thickness
μ fluid viscosity
ρ fluid density

Combination of Equations 5-6 and 5-7 suggests that k_L is proportional to the two third power of D_L (Nielsen and Villadsen, 1994):

$$k_L \propto (D_L)^{\frac{2}{3}} \qquad \textit{Equation 5-8}$$

5.4.2 SURFACE RENEWAL THEORIES

An alternative approach is to consider the liquid at the gas-liquid interface to flow freely over the surface, being constantly renewed. This surface renewal approach is applicable to mobile bubble surfaces (Oolman, 1984). It considers the replacement of discrete liquid elements at the gas-liquid interface by interaction with the bulk liquid. Each element is regarded as stagnant and being infinitely deep, and the absorption of compound A from the gas phase is determined by the exposure time of the liquid element (Nielsen and Villadsen, 1994).

According to **Higbie's theory**, the specific rate of mass transfer is given by:

$$r_A = 2\left(\frac{D_L}{\pi \cdot t_E}\right)^{\frac{1}{2}} \cdot (C^* - C) \qquad \textit{Equation 5-9}$$

where t_E is the contact time of the fluid as it flows around the bubble. This exposure time is defined as the quotient of the bubble diameter d_b and the bubble rise velocity u_b *i.e.* the time required for the fluid particle to travel one equivalent diameter (Moo-Young and Blanch, 1981). By comparison of Equation 5-9 and Equation 5-3:

$$k_L a = 2\left(\frac{D_L}{\pi \cdot t_E}\right)^{\frac{1}{2}} \qquad \textit{Equation 5-10}$$

Which can be interpreted as (Atkinson and Mavituna, 1992):

$$k_L \propto (D)^{\frac{1}{2}} \qquad \textit{Equation 5-11}$$

Danckwerts proposed a random rate of surface renewal, shown in Equation 5-12 where s is a statistical parameter representing the surface renewal rate (Oolman, 1984). Again dependence of k_L to the square root of D is observed.

$$k_L \propto (D_L s)^{\frac{1}{2}} \qquad \textit{Equation 5-12}$$

5.5 CORRELATIONS FOR DETERMINING $k_L a$

When considering empirical relationships, it is beneficial to correlate the dependence of $k_L a$ with the properties of the fluids, geometry and operating conditions using dimensionless groups.

The **Reynolds number** signifies the ratio between inertial and viscous forces:

$$\mathrm{Re} = \frac{\text{intertial forces}}{\text{viscous forces}} = \frac{du\rho}{\mu} \qquad \textit{Equation 5-13}$$

where

d	characteristic linear dimension	m
u	appropriate velocity	$m.s^{-1}$
ρ	fluid density	$kg.m^{-3}$
μ	fluid viscosity	Pa.s

The **Sherwood number** signifies the ratio between the overall mass transfer and diffusional mass transfer:

$$Sh = \frac{\text{total mass transfer}}{\text{diffusive mass transfer}} = \frac{k_L d}{D_L} \qquad \textit{Equation 5-14}$$

The **Schmidt number** signifies the ratio between momentum diffusivity and mass diffusivity. For a particular set of conditions, this is a constant:

$$Sc = \frac{\text{momentum diffusivit y}}{\text{mass diffusivit y}} = \frac{\mu}{\rho D_L} \qquad \textit{Equation 5-15}$$

The **Grashof number** is the ratio between the buouyancy and viscous forces acting on bubbles:

$$Gr = \frac{\text{gravitaional forces}}{\text{viscous forces}} = \frac{d^3 \rho g(\rho_L - \rho_G)}{\mu^2} \qquad \textit{Equation 5-16}$$

The **Peclet number** is given by:

$$Pe = \frac{du}{D_L} \qquad \textit{Equation 5-17}$$

Table 5-1 contains correlations for determining the mass transfer coefficient, k_L, in various systems (Atkinson and Mavituna, 1992). Where these are empirically based, their applications are limited to the range of the experimental data. In keeping with the approach of the analytical models (Section 5.4), the functionality in terms of the diffusivity is also given.

Table 5-1: Mass transfer coefficient correlations (Moo-Young and Blanch, 1981).

Correlation	k = fn(D)	Comments and References
Analytical Equations		
$k_L = \frac{D}{z}$	$k_L \propto D$	For stagnant medium or a film around a bubble – two film theory (Lewis and Whitman, 1924)
$k_L = 2(\frac{D}{\pi t})^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	Penetration theory (Higbie, 1935)
$k_L \propto (Ds)^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	Surface renewal theory (Danckwerts, 1951)
Bubbles in Stagnant Environment		
$Sh = 2$	$k_L \propto D$	Bubbles with rigid or mobile surfaces, Re = Gr = 0 (Frossling, 1938)
$Sh = 1.01 \cdot (Gr \cdot Sc)^{\frac{1}{3}}$	$k_L \propto (D)^{\frac{2}{3}}$	Re = 0, Gr > 0, Gr is based on the boundary layer thickness (Levich, 1962)
Moving bubbles		with rigid surfaces, free rise or free fall, packed bed, trickle bed
$Sh = 0.99 \cdot (\text{Re} \cdot Sc)^{\frac{1}{3}}$	$k_L \propto (D)^{\frac{2}{3}}$	Creeping flow, Re < 1 (Levich, 1962)
$Sh = 1.01 \cdot (Pe)^{\frac{1}{3}}$	$k_L \propto (D)^{\frac{2}{3}}$	Re < 1, Pe >> 1 (Levich, 1962)
$Sh = 2 + 0.73 \cdot (\text{Re})^{\frac{1}{2}} (Sc)^{\frac{1}{3}}$	$k_L \propto D, (D)^{\frac{2}{3}}$	Re >> 1 (Griffin, 1960)
$Sh = 2 + 0.31 \cdot (Sc \cdot Gr)^{\frac{1}{3}}$	$k_L \propto D, (D)^{\frac{2}{3}}$	Re is expressed in terms of bulk Gr, free suspension (Calderbank and Moo-Young, 1961)
$Sh = 0.13 \cdot (\text{Re})^{\frac{3}{4}} (Sc)^{\frac{1}{3}}$	$k_L \propto (D)^{\frac{2}{3}}$	Re is for isotropic turbulence (Calderbank and Moo-Young, 1961)
Moving bubbles with Mobile Surfaces		
$Sh = 0.65 \cdot (Pe)^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	Re < 1, Pe >> 1 (Levich, 1962)
$Sh = 1.13 \cdot (Pe)^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	Potential flow (Bossinesq, 1905)
$Sh = 0.13 \cdot \left(1 - 2.9\,\text{Re}^{-\frac{1}{2}}\right)^{\frac{1}{2}} (Pe)^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	Re >> 1, Pe << 1(Lochiel and Calderbank, 1964)
$Sh = 0.65 \cdot (Pe)^{\frac{1}{2}} (1 + \frac{\text{Re}}{2})^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	10 < Re < 100 (Handamard, 1911)
$Sh = 1.13 \cdot (1 - \frac{2.9}{\text{Re}^{\frac{1}{2}}})(Pe)^{\frac{1}{2}}$	$k_L \propto (D)^{\frac{1}{2}}$	100 < Re < 1000 (Handamard, 1911)

5.6 EFFECTIVE INTERFACIAL AREA

Oolman (1984) claims that interfacial area is the factor which has the greatest effect on total oxygen transfer in an air-sparged reactor. The interfacial area varies with hydrodynamic conditions and liquid phase composition. It is a direct function of the bubble size distribution and gas hold-up within the reactor (Oolman, 1984).

The specific interfacial area can be defined based either on the liquid volume V_L or on the volume of the gas-liquid dispersion V_d. These definitions are related by the gas holdup, ε (Nielsen and Villadsen, 1994):

$$a = \frac{A}{V_L}$$

$$a_d = \frac{A}{V_d} = \frac{A}{V_L + V_G} \qquad \textit{Equation 5-18}$$

$$= (1-\varepsilon)a$$

$$\varepsilon = \frac{V_G}{V_L}$$

where

A	total interfacial area in the gas-liquid dispersion
a	specific interfacial area
a_d	specific interfacial area based on dispersion volume
V_L	liquid volume
V_G	gas volume
ε	gas hold up
V_d	dispersion volume

The specific interfacial area, a, is a function of the bubble size distribution in the gas-liquid dispersion, and may be expressed in terms of the gas holdup, ε, and the average bubble size, d_{mean}, as shown in Equation 5-19. The mean Sauter diameter, d_{Sauter}, a surface averaged diameter, may also be used as shown below where d_p is the primary or initial bubble diameter (Nielsen and Villadsen, 1994).

$$a = \frac{6\varepsilon}{(1-\varepsilon)d_{mean}} = \frac{6\varepsilon}{(1-\varepsilon)d_{Sauter}} \qquad \textit{Equation 5-19}$$

$$d_{Sauter} = \frac{\sum n_i d_p^3}{\sum n_i d_p^2}$$

5.6.1 BUBBLE SIZE DISTRIBUTION

In reactors, three main factors interact to determine the bubble size distribution (Nielsen and Villadsen, 1994):

1. **Bubble formation**, determined by the breakup of the gas stream into discrete bubbles as it is sparged into the liquid phase.
2. **Bubble breakup**, determined by the competition between the stabilising effect of the surface tension and the destabilising effect of inertial forces.
3. **Bubble coalescence**, *i.e.* fusion of bubbles, determined by properties of the gas-liquid interface.

A bubble is formed at the orifices of a sparger when the buoyancy force on the bubble exceeds the surface tension acting on the periphery of the orifice. The initial bubble diameter d_p can be determined from a force balance and is dependent on the surface tension, σ, liquid density, ρ_L, gas density, ρ_G, and orifice diameter, d_o, (Equation 5-20). This equation only holds for low flow rates where the bubble size is independent of the gas flow rates (Nielsen and Villadsen, 1994). At higher gas flowrates, the size of the bubble is not dependent on the size of the orifice, but rather on the degree of turbulence in the continuous phase (Moo-Young and Blanch, 1981).

$$d = \left(\frac{6\sigma \cdot d_0}{g(\rho_L - \rho_G)} \right)^{\frac{1}{3}} \qquad \textit{Equation 5-20}$$

5.7 DRIVING FORCE

As mentioned previously, the driving force for the mass transfer is determined by the concentration gradient. Across a stagnant film, this can be represented by the change in solute concentration, (C^* - C). In the context of microbial reactions, the overall rate of biochemical reaction is high (Neilsen and Villadsen, 1994). As a result, the bulk concentration, C, is frequently low. The critical oxygen concentration in bioleaching suspensions is around 0.5-1 mg/l. The maximum driving force is determined by the saturation concentration, C^*, quantified by Henry's law (Appendix B) (Schumpe *et al.*, 1982).

5.8 DISCUSSION

The effect of temperature on mass transfer can be considered in terms of its effect on the sub-processes of mass transfer *i.e.* mass transfer coefficient, interfacial area and the transfer driving force. As seen by the analytical models of mass transfer (Section 5.4), the transfer coefficient is affected by the diffusivity of the solute in the medium (Table 5-1) as shown by:

$$k_L \propto (D_L)^n \qquad \textit{Equation 5-21}$$

It is well known that temperature affects the diffusivities of compounds (Reid *et al.*, 1977). Hence, the effect of temperature on the mass transfer is governed mainly by the effect of temperature on diffusivity. The surface mobility of the bubbles is directly dependent on the chemical environment. This expresses the state of the interface, which in turn effects the value of n, the power of D_L in Equation 5-21 (Atkinson and Mavituna, 1992).

Interfacial area is directly related to the bubble size distribution and gas holdup, affected in turn by the hydrodynamics. Hence any effect of temperature on these phenomena can be considered as the effect of temperature on interfacial area. The driving force is directly related to the equilibrium solubility of the gas in the liquid. The effect of temperature on solubility has been correlated (Wilhelm *et al.*, 1977), hence this renders the effect of temperature on the driving force.

By considering the relative effect of temperature on these mass transfer sub-processes, the main contributing factor can be elucidated. This is valuable as it enables one to approximate the effect of temperature on mass transfer simply, as covered in the next chapter.

5.9 NOMENCLATURE

Symbol	*Description*	*Units*
A	total interfacial area in the gas-liquid dispersion	
C_A, C	solute concentration in the bulk liquid	
C^*_A, C^*	equilibrium solute concentration	
D_{AB}	diffusion coefficient of solute A in solvent B	
D_L	diffusivity of the solute in the liquid	
H	Henry's constant	atm
J_A	mass flux of component A in B	
K	overall transfer coefficient	
V_d	dispersion volume	
V_G	gas volume	
V_L	liquid volume	
x	measure of displacement across the gas-liquid interface	
$\frac{dC_A}{dx}$	concentration gradient	
dimensionless groups		
Gr	Grashof number	
Pe	Peclet number	
Re	Reynolds number	
Sc	Schmidt number	
Sh	Sherwood number	
We	Weber number	
a	specific interfacial area in the gas-liquid dispersion	
a_d	specific interfacial area based on dispersion volume	
d	characteristic linear dimension	m
d_b	appropriate bubble size parameter	m
d_o	orifice diameter	m

Symbol	*Description*	*Units*
d_p	primary or initial bubble diameter	m
d_{Sauter}	Sauter bubble diameter	m
d_{mean}	mean bubble size	m
g	acceleration due to gravity	$m.s^2$
k_G	gas side transfer coefficient	
k_L	liquid side transfer coefficient	
n	diffusivity order	
n_i	number of bubbles of a particular size	
p	partial pressure of the solute phase	
r_A	rate of mass transfer of A	
r_o	solute A sphere radius	
s	surface renewal factor	
t_E	exposure time	s
u	appropriate velocity	$m.s^{-1}$
x	mole fraction of the solute in the liquid	
δ	stagnant film thickness	
ε	gas hold up	
μ	fluid viscosity	Pa.s
ρ	fluid density	$kg.m^{-3}$
ρ_L, ρ_G	liquid and gas density	$kg.m^3$
σ	surface tension	$N.m^{-1}$

5.10 REFERENCES

Atkinson, B. and F. Mavituna (1992): "Biochemical Engineering and Biotechnology Handbook", Stockton Pr, New York, Chapter 12

Moo-Young, M. and H.W. Blanch (1981): "Design of Biochemical Reactors, Mass transfer criteria for simple and complex systems", *Advances in Biochemical Engineering*, **19**, 1-70, Springer Verlag, Berlin

Nielsen, J. and J. Villadsen (1994): "Bioreaction Engineering Principles", Plenum Press, New York, p295-303

Oolman, T. (1984): "Bubble coalescence and mass transfer in air sparged bioreactors", Ph.D. Thesis, University of California, Berkeley

Reid, R.C., J.M. Prausnitz and T.K. Sherwood (1977): "The properties of gases and liquids, 3rd Edition", McGraw-Hill Book Company, United States of America

Schumpe, A., G. Quicker and W.D. Deckwer (1982): "Gas Solubilities in Microbial Culture Media", *Advances in Biochemical Engineering*, **24**, 1-38, Springer Verlag, Berlin

Welty, J.R., C.E. Wicks and R.E. Wilson (1984): "Fundamentals of Momentum, Heat and Mass Transfer, 3rd Edition", John Wiley & Sons, Canada

Winkler, M. (1981): "Biological Treatment of Waste Water", Ellis Horwood Ltd. Publishers, Chicester, p 42-76

Wilhelm, E., R. Battino and R.J. Wilcock (1977): "Solubility of Gases in Water", *Chemical Reviews*, 77, 219-262

6. Effect of Temperature on Mass Transfer - A Theoretical Approach

The previous chapter reviewed the three domains of mass transfer, *i.e.* the transfer coefficient, interfacial area and the transfer driving force. In this section, the effect of temperature on each of these domains is investigated from a theoretical point of view by studying the parameters affecting the domain and incorporating our knowledge of the effect of temperature on each of these parameters.

6.1 TRANSFER COEFFICIENT

The magnitude of k_L is influenced by the diffusivity of dissolved gas in the liquid, surfactants which affect the interfacial properties (Section 5.3), rheology of the liquid, bubble size and flow regimes in the liquid (Atkinson and Mavituna, 1992). According to analytical equations for mass transfer (Section 5.4), the transfer coefficient is a function of the diffusion coefficient to the power of half, two thirds or unity. The diffusion coefficient is in turn modelled according to the $\left(\frac{T}{\mu_B}\right)$ group (Liley *et al.*, 1988), hence the effect of temperature on both the diffusion coefficient and solvent viscosity needs to be evaluated in order to understand its effect on the mass transfer coefficient.

6.1.1 MOLECULAR DIFFUSIVITY

Diffusion refers to the net transport of material within a single phase in the absence of mechanical or convectional mixing. The diffusion coefficient for a binary mixture relates the net rate of transport to the transfer driving force by the following expressions:

$$J_A^M = -cD_{AB}\nabla x_A \qquad \textit{Equation 6-1}$$

$$J_B^M = -cD_{BA}\nabla x_B \qquad \textit{Equation 6-2}$$

where

c	total molar concentration
D_{AB}	the diffusion coefficient of one constituent in a binary system
J	mass flux

The forces around the molecules also affect diffusivity. These forces are a function of temperature, pressure and composition. Since molecules are more densely packed in liquids, the forces existing around the molecules are greater. This results in a lower diffusivity than for that of the low-pressure gases (Reid *et al.*, 1977).

The diffusivity constant can be estimated by the Stokes-Einstein equation (Equation 6-3) (Moo-Young and Blanch, 1981), which is based on the hydrodynamic theory of large spherical molecules diffusing in a dilute solution (Reid *et al.*, 1977). This can be simplified to render the contributions from viscosity and temperature as shown below (Liley *et al.*, 1988) and will be used as a starting point in developing correlations for D_L as a function of temperature.

$$D_L = \frac{RT}{6\pi \cdot r_0 \mu_B} = f\frac{T}{\mu_B} \qquad \textit{Equation 6-3}$$

where

D_L	liquid phase diffusivity
R	universal gas constant
T	temperature
r_o	solute A sphere radius
μ_B	solvent B viscosity
f	constant representing solute size

6.1.1.i Estimation of Diffusivity at Infinite Dilution

Diffusion of A into an infinitely dilute solution of A in B implies that each molecule of A is in an environment of essentially pure B. In engineering work, this is considered to apply even for concentrations of A up to 5 mole percent (Reid *et al.*, 1977).

The **Wilke-Chang Estimation Method** (Equation 6-4) is an empirical model of the Stokes-Einstein relation and is well accepted for the estimation of diffusivity at infinite dilution. This method was tested using 251 solute-solvent systems with an average error of 10% (Reid *et al.*, 1977). Several authors have tested the Wilke-Chang correlation with sparingly soluble gases in water with varying results. Recently it was reported that this correlation gave an error of 6.9% using recommended value for the association factor (ϕ) for water as solvent of 2.6. If the association factor for water was decreased to 2.26, then the error was reduced to 0.4% (Liley *et al.*, 1988).

$$D^{\circ}_{AB} = 7.4\times10^{-8}\frac{(\phi \cdot M_B)^{\frac{1}{2}}T}{\mu_B V_{Ab}^{0.6}} \qquad \textit{Equation 6-4}$$

where

$D°_{AB}$	liquid phase diffusivity at infinite dilution	$cm^2.s^{-1}$
M_B	molecular weight of solvent B	$g.mol^{-1}$
T	absolute temperature	K
V_{Ab}	molal volume of solute A at its normal boiling temperature	$cm^3.mol^{-1}$
μ_B	viscosity of solvent B	cP
ϕ	association factor of solvent B	
ρ_A	solute density	$g.dm^3$

The molal volume at the normal boiling point V_{Ab} can be estimated from the saturated liquid density (V^{sat}) using the Bhirud correlation for non-polar compounds (Liley *et al.*, 1988) given in Equation 6-5. This correlation was tested with a wide range of hydrocarbons with an average error of less than 1%. The values for carbon dioxide and oxygen are 34.8778 and 27.7145 $cm^3.mol^{-1}$ respectively.

$$V^{sat} = \exp\left[\frac{RT_c}{P_c}\cdot\left((\ln U)^0 + \omega(\ln U)^1\right)\right]$$

$$(\ln U)^0 = \ln T_r + 1.39644 - 24.076T_r + 102.615T_r^2 - 255.719T_r^3 + 355.805T_r^4 - 256.671T_r^5 + 75.1088T_r^6$$

$$(\ln U)^1 = 13.4412 - 135.743T_r + 533.380T_r^2 - 1091.453T_r^3 + 1231.43T_r^4 - 728.22T_r^5 + 176.737T_r^6$$

Equation 6-5

where

V^{sat}	saturated liquid molar volume at bubble point	$cm^3.mol^{-1}$
P_c	critical pressure	atm
R	universal gas constant	$atm.cm^3.mol.K$
T_c	critical temperature	K
$(ln\ U)^0, (ln\ U)^1$	functions of the reduced temperature	
ω	accentric factor	
T_r	reduced temperature	

6.1.1.ii Estimation Method for Diffusivity of Carbon Dioxide in Water

Danckwerts (1966) proposed the empirical expression for estimating the diffusivity of dissolved carbon dioxide in solvents for which data is lacking:

$$D_{AB} = 5.4\times10^{-8}\left(\frac{M^{0.5}L_S^{0.5}T}{\mu_L V_{AT}^{0.5}\lambda^{0.3}}\right)^{0.93}$$

Equation 6-6

where

M	molecular weight of the solvent	
L_S	latent heat of vaporisation of the solvent at normal boiling point	$cal.g^{-1}$
μ_L	solvent viscosity	cP
V_{AT}	molecular volume of the solute at temperature T	$cm^3.mol^{-1}$
λ	latent heat of vaporisation of the solute	$cal.g^{-1}$

Most of the infinite dilution techniques account for temperature by assuming that the group $\left(\frac{D^{\circ}_{AB}\cdot\mu_B}{T}\right)$ remains constant (Equation 6-3). This assumption has been shown to be accurate over small temperature ranges, *i.e.* from 10° to 90° C (Liley *et al.*, 1988). Hence it is necessary to have an understanding of the solvent viscosity.

6.1.2 VISCOSITY

Viscosity is a measure of the internal fluid friction which tends to oppose any dynamic change in the fluid motion. This frictional drag between the layers originates owing to individual collisions between the molecules and interacting force fields between the molecules. It is a dynamic, non-equilibrium property and is a function of the state of the material, including temperature, pressure and volume (Reid *et al.*, 1977). The frictional drag between the layers governing the viscosity is affected primarily by the interacting force fields between the close packed liquid molecules. The density of liquids is such that the average molecular separation distance falls within the effective range in which the force fields between the molecules are effective (Reid *et al.*, 1977). On increasing molecular separation with increasing temperature, the viscosity of a liquid decreases.

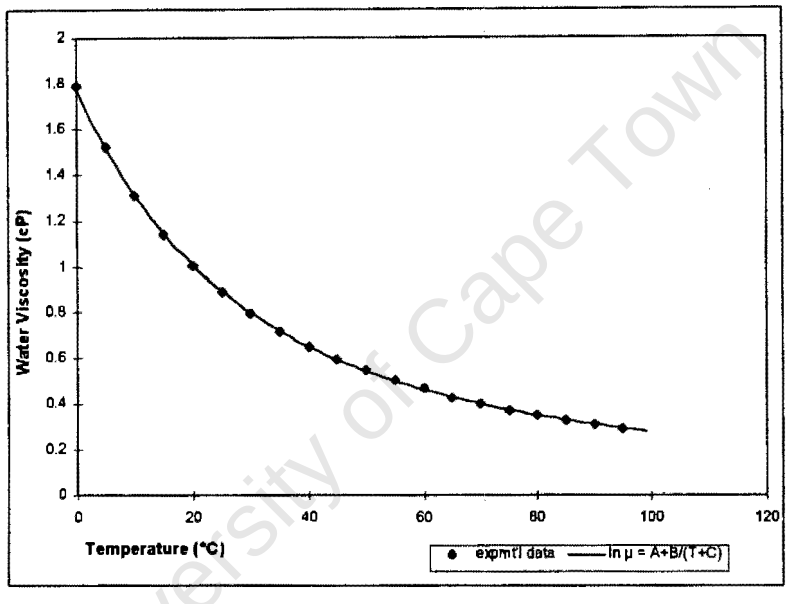


Figure 6-1: Variation of the viscosity of water with temperature, modelled using a three-parameter correlation. Experimental data is taken from Weast and Astle (1982).

At a reduced temperature below 0.75 (Liley *et al.*, 1988), liquid viscosity can be modelled using either a three or two parameter model, Equation 6-7 or Equation 6-9 respectively. The regression constants, a, b and c are obtained from experimental data. Constant *c* can be estimated using Equation 6-8. The *a* and *b* constants are dependent on the molecular structure of the liquid. The simpler two-parameter model is known as the Andrade correlation and is obtained by setting *c* to zero. Water viscosity data taken from Weast and Astle (1982) was modelled using the three-parameter correlation as shown in Figure 6-1. Values for *a*, *b* and *c*, determined using mathematical optimisation, are as follows: -3.679, 551.941 and -143.372.

$$\ln \mu_L = a + \frac{b}{T + c} \qquad \textit{Equation 6-7}$$

$$c = 17.1 - 0.19 \cdot T_b \qquad \textit{Equation 6-8}$$

$$\ln \mu_L = a_1 + \frac{b_1}{T} \qquad \textit{Equation 6-9}$$

where

μ_L	liquid viscosity	cP
T	absolute temperature	K
T_c	critical temperature	K
T_b	normal boiling point	K
a,b,c	normal regression constants	

6.1.3 EFFECT OF TEMPERATURE ON DIFFUSION COEFFICIENT

According to Equation 6-3 the term $\left(\frac{T}{\mu_B}\right)$ represents the influence of temperature on the diffusion coefficient. Combination of Equations 6-3 and 6-7 illustrates the dependence of diffusivity on temperature. Hence the mass transfer coefficient is influenced by temperature and viscosity:

$$k_L \propto (D_L)^n \propto \left(\frac{T}{\mu_B}\right)^n \qquad \textit{Equation 6-10}$$

where $0.5 \leq n \leq 1$ (Table 5-1). Based on Equation 6-10, the effect of temperature on k_L can be seen from Figure 6-2 in which Equation 6-10 is plotted with n=$^1/_2$, $^2/_3$ and 1. The values used for the regression constants are those calculated previously for water.

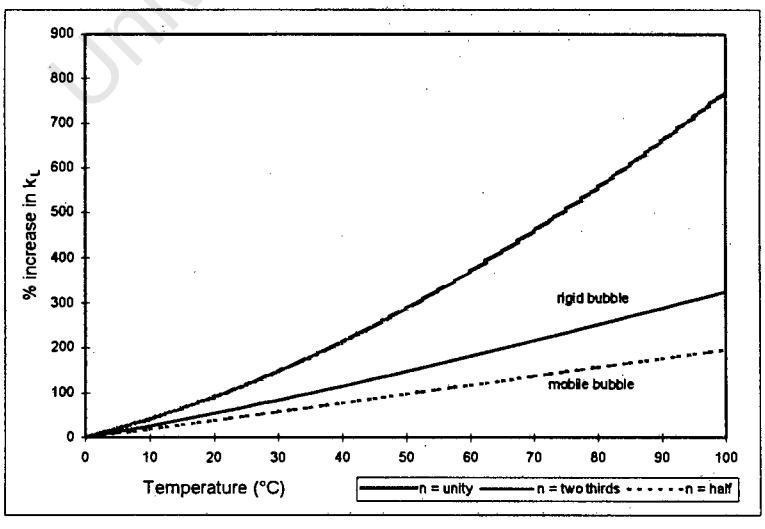


Figure 6-2: Increase in the mass transfer coefficient with temperature.

Another important consideration is the retardation effect of electrolytes and adsorbed materials at the bubble surface, which leads to an increase in the liquid phase resistance near and at the gas liquid interface. Hence, it is necessary to consider the effect of electrolytes on gas diffusivity (Ratcliff and Holdcroft, 1963). This is not included in this study owing to the availability of data.

6.2 INTERFACIAL AREA

The interfacial area is strongly dependent on the bubble size distribution. The bubble size is governed by initial bubble size, density, surface tension, power input and gas holdup and the extent to which coalescence occurs. In media containing salts, the tendency for bubbles to coalesce decreases with increasing ionic strength (Winkler, 1981).

6.2.1 BUBBLE COALESCENCE AND BREAKUP

Coalescence of bubbles rising in a line takes place in several stages (Moo-Young and Blanch, 1981):

1. the approach of the following bubbles to the vortex region of the leading bubble
2. the trailing bubble moves in the vortex of the leading bubble until the bubbles are separated only by a thin interface
3. final thinning and rupture of the film between the bubbles

During the final stage, the initial film thickness decreases due to a pressure difference between the liquid of the film and the liquid outside the border of the film. The concentration of the surface-active material within the film differs from the outer concentration resulting in diffusion across the film. The difference in surface tension between the film and the bulk is eliminated, hence the film is stretched to keep the balance of forces satisfied. The predictions of the theory agree well with coalescence times reported for various electrolyte systems (Moo-Young and Blanch, 1981). Bubble breakup occurs when the hydrodynamic force exerted on the surface of the bubble exceeds the tension force (Oolman, 1984). Hence, surface tension affects the rate of bubble coalescence and therefore bubble size distribution.

The **Weber number** is the ratio of the dynamic pressure forces to surface forces:

$$\mathrm{We} = \frac{\text{dynamic pressure forces}}{\text{surface forces}} = \tau \frac{d_b}{\sigma} \qquad \textit{Equation 6-11}$$

where

We	Weber number
τ	shear stress
d_b	bubble diameter
σ	surface tension

Using isotropic turbulence theory, the dynamic pressure forces can be expressed by:

$$\tau \propto \rho \left[\left(\frac{P}{V}\right)\left(\frac{d_b}{\rho}\right)\right]^{\frac{2}{3}} \qquad \textit{Equation 6-12}$$

where

d_b	bubble diameter
d_m	maximum stable bubble diameter
P	power input
V	volume
ρ	density of the gas phase

At equilibrium, the Weber number is constant and a maximum stable bubble size can be predicted by Equation 6-13 (Moo-Young and Blanch, 1981). Thus the bubble size distribution is affected by surface tension, density and volumetric power input. The effect of temperature will be mediated through its effect on surface tension and vapour density.

$$d_m \propto \left[\frac{\sigma^{0.6}}{\left(\frac{P}{V}\right)^{0.4}\rho^{0.2}}\right] \qquad \textit{Equation 6-13}$$

where

d_m	maximum stable bubble diameter

6.2.2 SURFACE TENSION

The boundary layer between a liquid phase and a gas phase may be considered to be a third phase with properties intermediate between those of a liquid and a gas. At low gas densities surface molecules are attracted sideways and toward the bulk liquid. They experience less attraction in the direction of the bulk gas. Hence the surface layer is in tension and tends to be attracted to the smallest area compatible with the mass of material, container restraints and external forces. This tension is represented by σ, the surface tension. It is defined as the force exerted in the plane of the surface per unit length (Reid *et al.*, 1977).

As the temperature is raised, the surface tension of a liquid in equilibrium with its own vapour decreases and becomes zero at the critical point. Essentially all useful estimation techniques for the surface tension of a liquid are empirical (Reid *et al.*, 1977).

6.2.2.i *Macleod-Sugden Correlation*

The Macleod-Sugden correlation (Equation 6-14) is applicable to hydrogen bonded compounds. It provides a relationship between surface tension and the liquid and vapour densities.

$$\sigma^{\frac{1}{4}} = [\mathrm{P}](\rho_L - \rho_G) \qquad \textit{Equation 6-14}$$

where

σ	surface tension	dyne.cm^{-1}
[P]	parachor, temperature independent parameter	
ρ	density	g.cm^{-3}

The parachor is a temperature independent parameter which is based on the structure of the molecule, where the contributions for various groups of the molecule are additive (Reid *et al.*, 1977). At low pressures, the liquid density is far greater than the vapour density, and the latter term can be neglected. Then, according to Equation 6-14, the surface tension varies with $([P]\cdot\rho_L)^4$, and is very sensitive to the values of the parachor and the liquid density chosen (Reid *et al.*, 1977).

Instead of employing experimental densities, correlations relating density to temperature may be used (Reid *et al.*, 1977):

$$\rho_L - \rho_G = \rho_{Lb}\left(\frac{1-T_r}{1-T_{br}}\right)^n$$

$$\sigma = ([P]\rho_{LB})^4\left(\frac{1-T_r}{1-T_{br}}\right)^{4n} \qquad \textit{Equation 6-15}$$

where

ρ_{LB}	molal liquid density at the normal boiling point	g.cm^{-3}
n	index which ranges from 0.25 to 0.31	
T_{br}	reduced boiling temperature	

6.2.2.ii *Correlations of Corresponding-State*

The **Brock and Bird correlation** applies to non-polar liquids (Liley *et al.*, 1988):

$$\sigma = P_c^{\frac{2}{3}} T_c^{\frac{1}{3}} Q' (1-T_r)^{\frac{11}{9}} \qquad \textit{Equation 6-16}$$

$$Q_b = 0.1207\left(1+\frac{T_{br}\ln P_c}{1-T_{br}}\right)-0.281 \qquad \textit{Equation 6-17}$$

where

P_c	critical pressure	atm
T_c	critical temperature	K
Q_b	as defined in Equation 6-17	
T_{br}	reduced boiling point	

The **Riedel correlation** is an extension of the Brock and Bird correlation, to include polar liquids. The accuracy of these methods have not been reported (Reid *et al.*, 1977).

$$\sigma = P_c^{\frac{2}{3}}T_c^{\frac{1}{3}}Q_r\left(\frac{1-T_r}{1-T_{br}}\right)-0.281 \qquad \text{Equation 6-18}$$

$$Q_r = 0.1574+0.359\varpi-1.769X-13.69X^2-0.510\varpi^2+22.03\varpi X \qquad \textit{Equation 6-19}$$

where

P_c	critical pressure	atm
T_c	critical temperature	K
Q_r	as defined in Equation 6-19	
X	Stiel polar factor (0.023 for water)	
P_{vp_r}	reduced vapour pressure	
ϖ	accentric factor (0.344 for water)	

6.2.2.iii *Effect of Temperature on Surface Tension*

In the three correlations considered, surface tension depends on reduced temperature according to:

$$\sigma \propto (1-T_r)^y \qquad \textit{Equation 6-20}$$

where y for water is 1.24 according to the Macleod-Sugden, 1.22 according to Brock and Bird, and 1.02 according to Riedel. This provides agreement for the effect of temperature on surface tension. For values of T_r between 0.4 and 0.7, *i.e.* over a small range of temperature, the change of surface tension with temperature is almost linear and can be represented by (Reid *et al.*, 1977):

$$\sigma = a+bT \qquad \textit{Equation 6-21}$$

Figure 6-3 displays the values obtained for the surface tension of pure water using the Riedel correlation. The results of the linear regression concur that the change of surface tension with temperature is linear in the range of interest.

6.2.3 BUBBLE SIZE DISTRIBUTION

The break up of submerged gas jets into bubbles in stagnant liquid can be modelled by (Joshi and Sharma, 1976):

$$d_p = 1.41\left(\frac{Q^2}{g}\right)^{\frac{1}{5}} \qquad \textit{Equation 6-22}$$

where

d_p	diameter of primary bubbles	m
Q	volumetric gas flowrate	$m^3.s^{-1}$
g	acceleration due to gravity	$m.s^{-2}$

The bubbles formed by the break up of gas jets may be classified as primary bubbles. It has been observed that the shape and form of these primary bubbles is random and they are unstable. As a result they disintegrate into smaller bubbles. It was determined that a wide bubble size distribution forms under turbulent conditions, according to (Joshi and Sharma, 1976):

$$\overline{d_s} = 7.13 \cdot 10^{-3}\,\mathrm{Re}^{-0.05} \qquad \mathrm{Re} > 10{,}000$$

$$\mathrm{Re} = \frac{d_o u_n \rho_G}{\mu_G} \qquad \textit{Equation 6-23}$$

where

$\overline{d_s}$	average diameter, representing the whole distribution
d_o	nozzle or orifice diameter
u_n	nozzle gas velocity

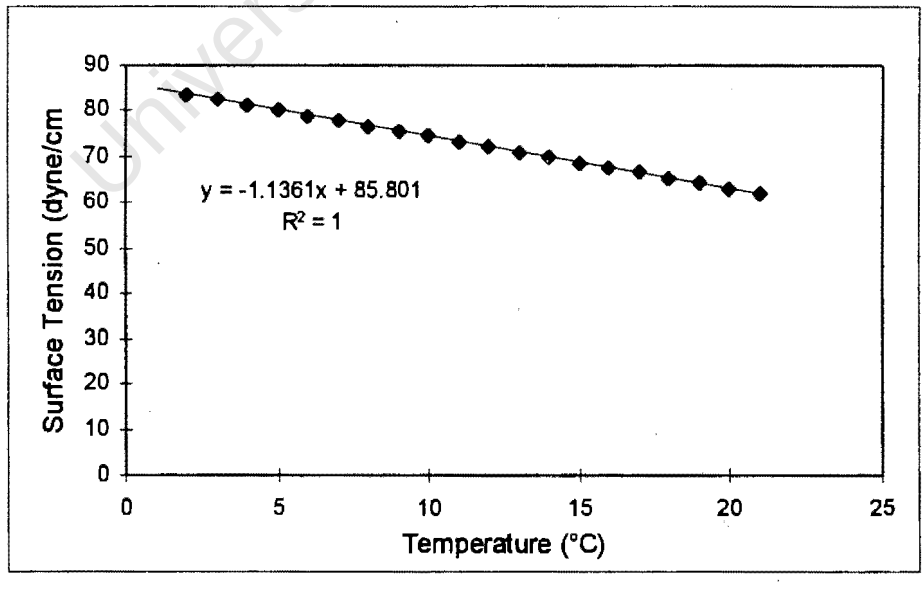


Figure 6-3: Effect of temperature on the surface tension of water as predicted by the Riedel correlation (Equation 6-18).

ρ_G gas density
μ_G gas viscosity

A bubble of diameter $\overline{d_s}$ has a ratio of volume to surface area equal to that of the entire size distribution. There is no noticeable effect of nozzle diameter on $\overline{d_s}$ under the turbulent flow conditions. The Reynolds number is defined according to Equation 6-23 (Joshi and Sharma, 1976). At this point it is pertinent to note that temperature will have an effect on the density and viscosity of the gas, and as a result on the bubble size distribution.

6.2.4 EFFECT OF TEMPERATURE ON VAPOUR DENSITY

Assuming ideal gas behaviour for the vapour, the effect of temperature on vapour density can be determined using a modified Charles law as shown in Equations 6-24. Figure 6-4 shows vapour density data for carbon dioxide and oxygen from Holman (1989) correlated using this approach.

$$\frac{V_1}{T_1} = \frac{V_2}{T_2}$$

$$\rho = \frac{m}{V}$$

Equation 6-24

where

V	volume	dm^3
T	temperature	K
ρ	fluid density	$g \cdot dm^{-3}$
m	mass of gas	g

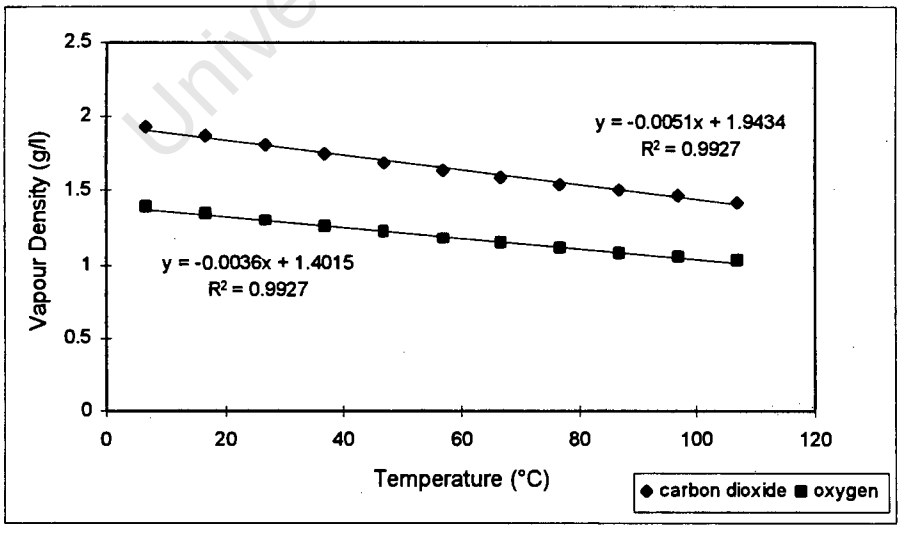


Figure 6-4: Linear variation of the vapour density of pure carbon dioxide and oxygen , with temperature (Holman, 1989).

6.2.5 EFFECT OF TEMPERATURE ON INTERFACIAL AREA

As mentioned previously, Equation 5-19, interfacial area is inversely proportional to the bubble size distribution. As changes in the maximum bubble size distribution will be echoed throughout the whole bubble size distribution, Equation 6-13 can be used to determine the effect of temperature on interfacial area.

$$a \propto \frac{\rho^{0.2}}{\sigma^{0.6}} \qquad \text{Equation 6-25}$$

Figure 6-5 shows the percentage increase in interfacial area as a function of temperature estimated using Equation 6-25. Based on this approach an increase in interfacial area of some 2.5 fold is found on increasing temperature from 40° to 80°C.

6.3 DRIVING FORCE

The driving force for gas-liquid mass transfer is the concentration gradient (C^*-C). As the rate of reaction or nutrient uptake is generally fast, the bulk liquid concentration, C, is maintained close to zero. The critical oxygen concentration in bioleaching suspensions is approximately 0.5-1 mg/l. The rate-controlling step is the transfer of the nutrient from the interface to the bulk medium. Hence the effect of temperature on the driving force is governed by the effect of temperature on the saturation concentration of the gas.

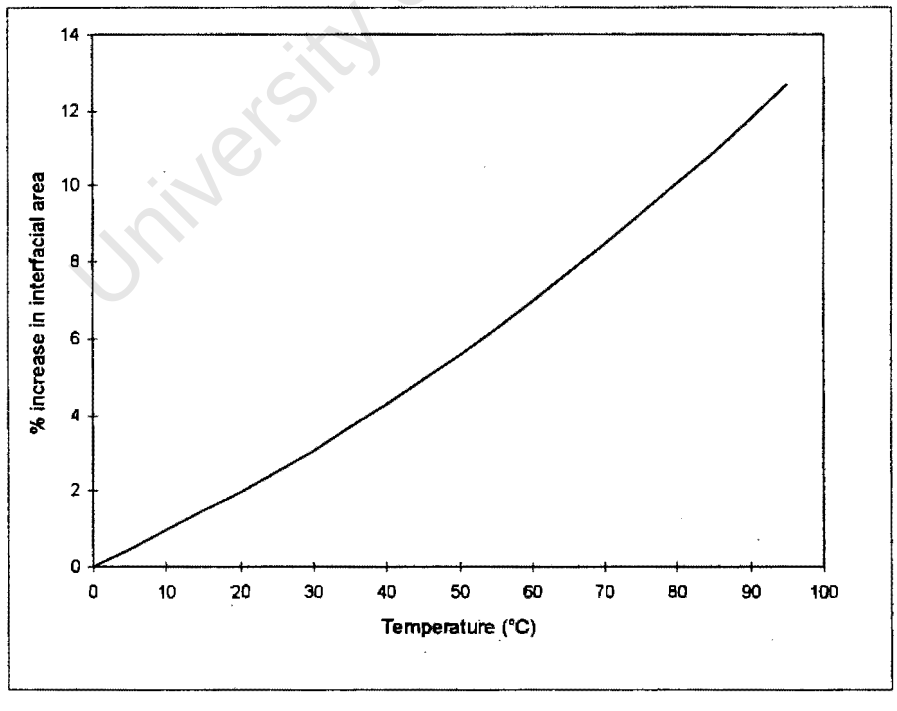


Figure 6-5: Increase in interfacial area estimated using the maximum stable bubble size at equilibrium, at constant Weber number.

6.3.1 OXYGEN AND CARBON DIOXIDE SOLUBILITY

Oxygen is very sparingly soluble and carbon dioxide is moderately soluble in aqueous solutions (Atkinson and Mavituna, 1992). The solubility of a gas is generally determined by the temperature and equilibrium partial pressure of the solute gas in the gas phase (Schumpe *et al.*, 1982). Various correlations for depicting the solubilities of these gases in water are detailed in Appendix B and illustrated for the case of oxygen in Figure 6-6. At pressures below 5 atmospheres, gas solubility can be expressed in terms of Henry's law, Equation 6-26 (Schumpe *et al.*, 1982). Table 6-1 contains Henry's Law constants for oxygen, nitrogen and carbon dioxide.

$$p = H \cdot x \qquad \textit{Equation 6-26}$$

where

p	partial pressure of the solute in the gas phase	atm
H	Henry's law constant	atm
x	mole fraction of the solute in the liquid phase	

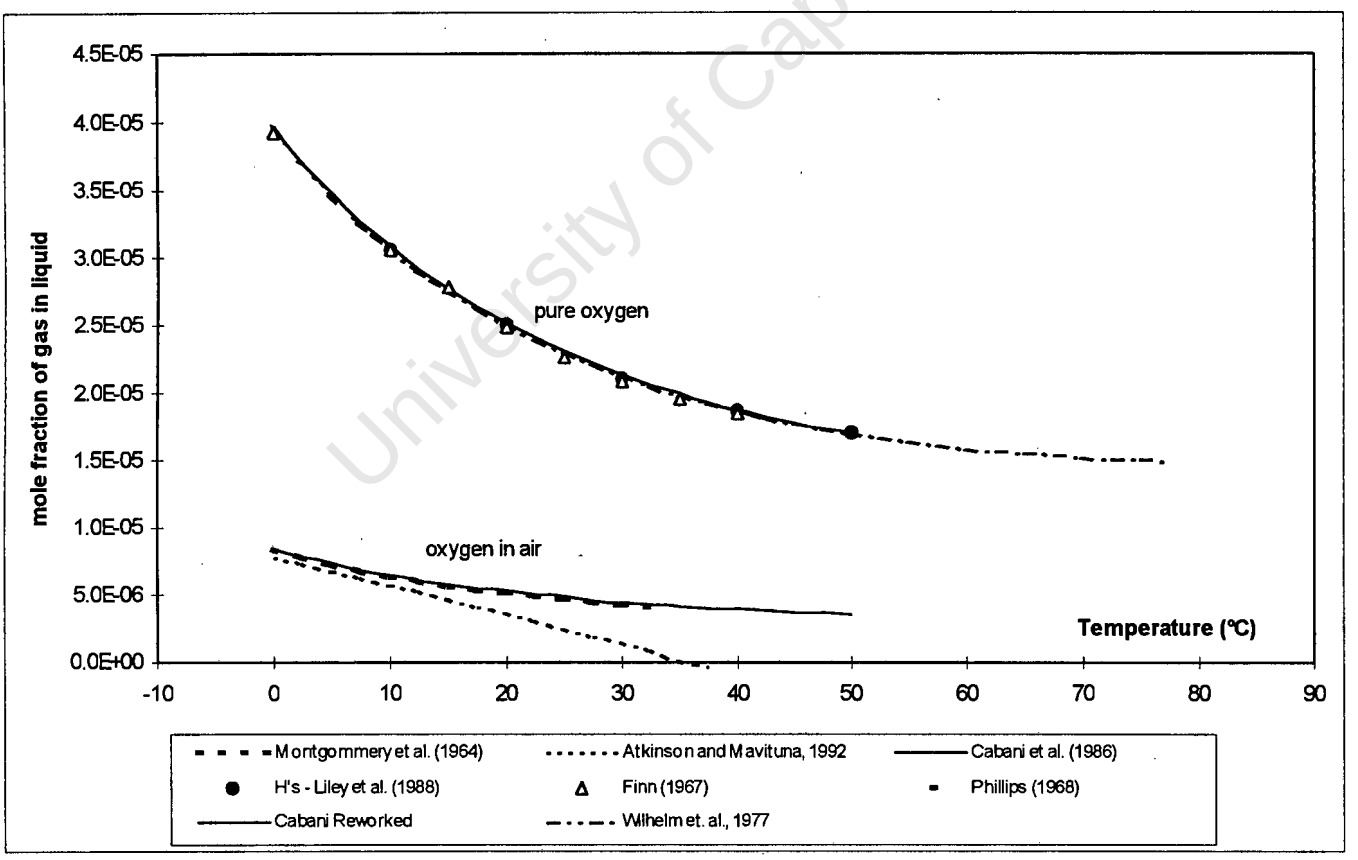


Figure 6-6: Solubility of oxygen in water predicted by various methods.

Table 6-1: Solubility of various gases in water at different temperatures, (Liley et al., 1988).

Temperature °C	Henry's Law constant, H (atm x 10^{-4})		
	O_2	N_2	CO_2
10	3.27	6.68	0.104
20	4.01	8.04	0.142
30	4.75	9.24	0.186
40	5.35	10.4	0.233
50	5.88	11.3	0.283

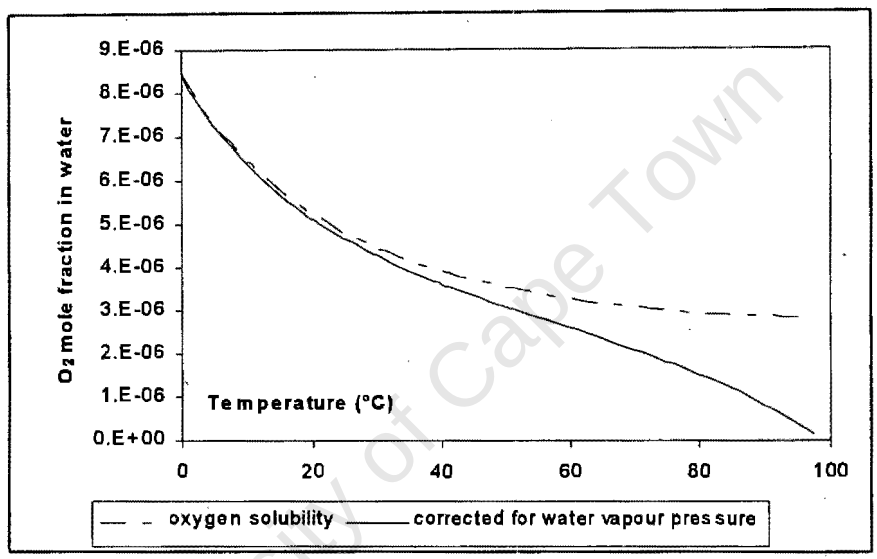


Figure 6-7: Oxygen solubility with and without water vapour pressure correction.

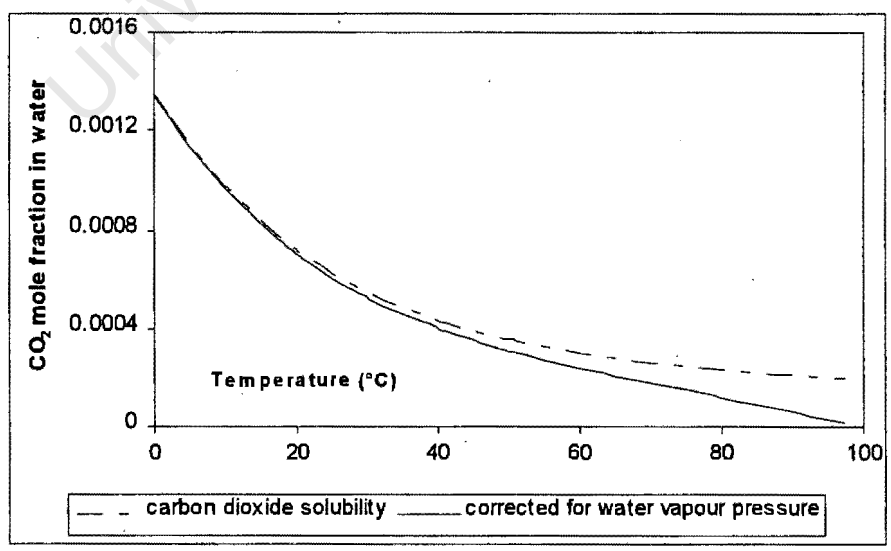


Figure 6-8: Carbon dioxide solubility with and without water vapour pressure correction.

Henry's law constants were correlated according to:

$$H = a + b.T + c.T^2 + d.T^3 \qquad \text{Equation 6-27}$$

where

H	Henry's constant	atm x 10^{-4}
T	temperature	°C

Solubility data and correlations from Cabani *et al.*, (1986), Wilhelm *et al.* (1977) and Liley *et al.* (1988) were combined over the specified temperature ranges as detailed in Appendix B and Figure 6-6. Figures 6-7 and 6-8 depict the variation of oxygen and carbon dioxide solubility between 0 and 100°C. The correlation constants are detailed in Table 6-2.

Table 6-2: Henry's Law correlation coefficients.

	CO_2	O_2	Order
	0-100°C	0-80°C	
a	0.0744	2.4788	0
b	0.0024	0.079	1
c	$5x10^{-5}$	$-1x10^{-4}$	2
d	$-3x10^{-7}$	$-2x10^{-6}$	3

6.3.2 GAS SOLUBILITIES CORRECTED FOR WATER VAPOUR PRESSURE

Owing to potential problems of evaporation when operating at higher temperatures, it is necessary to pass humidified gas to the reactors. Hence, equilibrium water vapour pressure is taken into account when calculating gas solubilities (Boogerd *et al.*, 1992). Charle's Law indicates that this will become increasingly important as temperature increases. The correction is incorporated into the Henry's law equation as follows:

$$x = \frac{p\left(1 - p^*_{WG}\right)}{p_T} \cdot \frac{p_T}{H} \qquad \text{Equation 6-28}$$

where

H	Henry's Law constant	atm
p	partial pressure of the gas in the mixture	atm
p_T	total pressure	1 atm
p^*_{WG}	vapour pressure of water	atm

This becomes increasingly important when considering operation at higher temperatures, as illustrated in Figures 6-7 and 6-8.

6.3.3 EFFECT OF ELECTROLYTES

The solubility of gases in aqueous electrolyte solutions has been the subject of numerous investigations. A reduction of the gas solubility in the presence of salts is observed. In other words, the activity coefficients of dissolved gases are usually increased by electrolytes (Schumpe *et al.*, 1982).

The Sechenov equation:

$$\log \frac{C_{jH_2O}}{C_j} = K_s C_{el} \qquad \textit{Equation 6-29}$$

where

C_{el}	electrolyte concentration	$mol \cdot l^{-1}$
C_{jH_2O}	concentration of the gas in water	$mol \cdot l^{-1}$
C_j	concentration of the gas	$mol \cdot l^{-1}$
K_s	Sechenov constant	$l \cdot mol^{-1}$

is widely used when considering the salting out effect on solubility (Schumpe *et al.*, 1982; Atkinson and Mavituna, 1992; Danckwerts, 1970; Long and McDevit 1952). Appendix C details the theoretical development of this approach.

The Sechenov constants, K_s, are specific with respect to gas, temperature and salt (Schumpe *et al.*, 1982; Danckwerts, 1970). Schumpe *et al.* (1978) propose a model to predict the salting out effect in terms of a change in Henry's constant for both single and mixed electrolytes. This model takes into account the ionic strength attributable to individual ions (I_i):

$$\log \frac{C_{jH_2O}}{C_j} = \log \frac{H_j}{H_{jH_2O}} = \sum_i H_i I_i \qquad \text{Equation 6-30}$$

$$I_i = \frac{1}{2} C_i z_i^2$$

where

Ci	concentration of ion i
z_i	ionic charge of ion i
I_i	ionic strength attributable to ion i
H_i	adjusted constant

The constant H_i is specific to the gas, the ion and the temperature (Tables C-3 and C-4). For single salt solutions, the Sechenov constants are then given by:

$$K_s = \frac{1}{2} \sum_i H_i x_i z_i^2 \qquad \text{Equation 6-31}$$

A sample calculation was conducted to elucidate the effect of an acidic (pH 1.5) ferric sulphate solution on the solubility of oxygen and carbon dioxide. The H_i data of Schumpe *et al.* (1982) is not amenable to extrapolation to higher temperatures. Hence, a temperature of 37°C was chosen to ensure sufficient data was available. The ionic components taken into consideration were H^+, SO_4^{2-} and Fe^{3+}. It was assumed that Fe^{2+} would be present in negligible quantities in solution. The SO_4^{2-} concentration was estimated by a charge balance. Figure 6-9 illustrates

the decrease in solubility of oxygen and carbon dioxide with increasing ferric iron concentration. In the bioleaching system the electrolyte concentrations would be at least ten times higher than these concentrations, including the presence of arsenic, copper and other metal ions. Calculation of the effect of these electrolytes is subject to the availability of the H_i parameters.

6.3.4 EFFECT OF TEMPERATURE ON DRIVING FORCE

Figure 6-10 shows the decrease in oxygen and carbon dioxide solubility, and therefore the mass transfer driving force, with temperature. No allowance is made for the presence of electrolytes. The solubilities were calculated assuming a molal concentration of oxygen and carbon dioxide in air of 20.96% and 1.003% respectively. A decrease in oxygen solubility of some 25%, and carbon dioxide solubility of 21% is seen on increasing the temperature from 40°C to 80°C, following correction for water vapour pressure.

6.4 COMPARISON OF THEORETICAL PREDICTION AND EXPERIMENTAL STUDIES

Based on the extent of the effect of temperature on each mass transfer sub-process (Figure 6-11) it is evident that the greatest contribution is that of the transfer coefficient. It must be stressed that Figure 6-11 indicates the functionality of the domain, *i.e.* k_L, a or C^*-C, with temperature and cannot be used to contrast absolute values of these parameters.

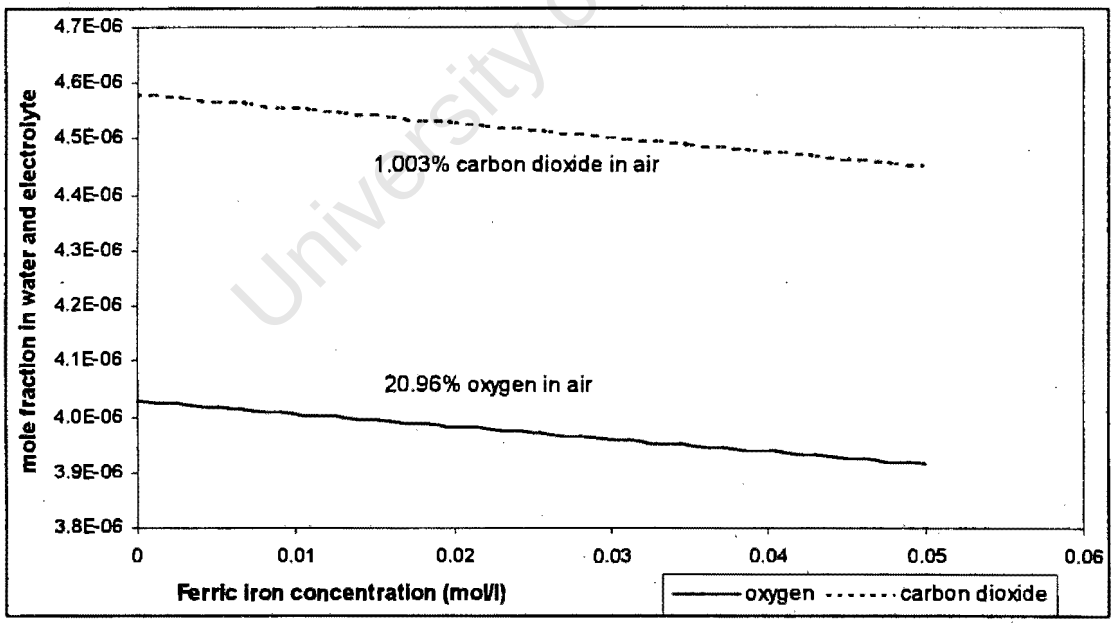


Figure 6-9: Effect of an acidic ferric sulphate solution on the solubility of oxygen and carbon dioxide in water at 37°C.

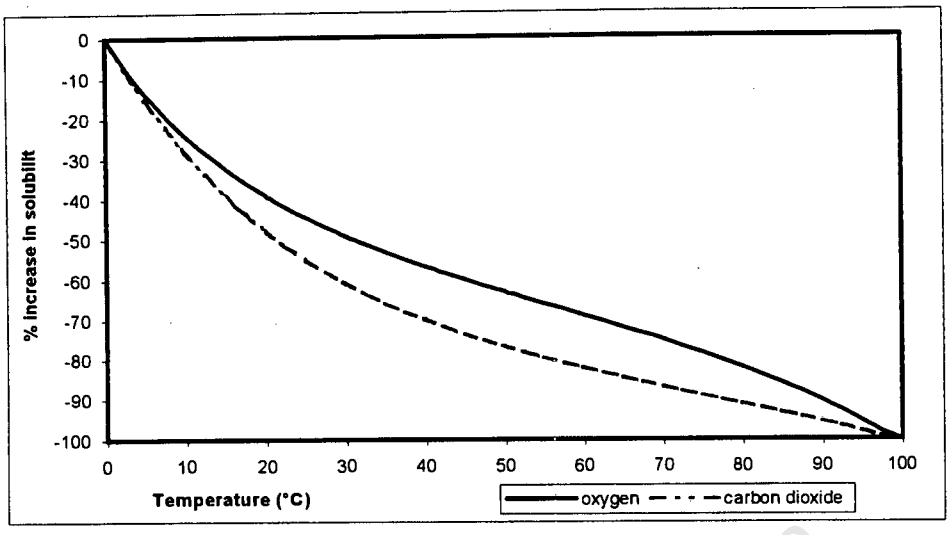


Figure 6-10: Percentage increases in oxygen and carbon dioxide solubility with water vapour correction.

Barnhart (1969) investigated the effect of temperature on k_L, and k_La in a rising bubble, air-water system, up to 50°C. He proposed a logarithmic correction between mass transfer and temperature given by:

$$\frac{k_{LT}}{k_{L20}} = (1.028)^{(T-20)} \qquad \text{Equation 6-32}$$

where

T	Temperature	°C
k_{LT}	mass transfer coefficient at T	
k_{L20}	mass transfer coefficient at 20°C	

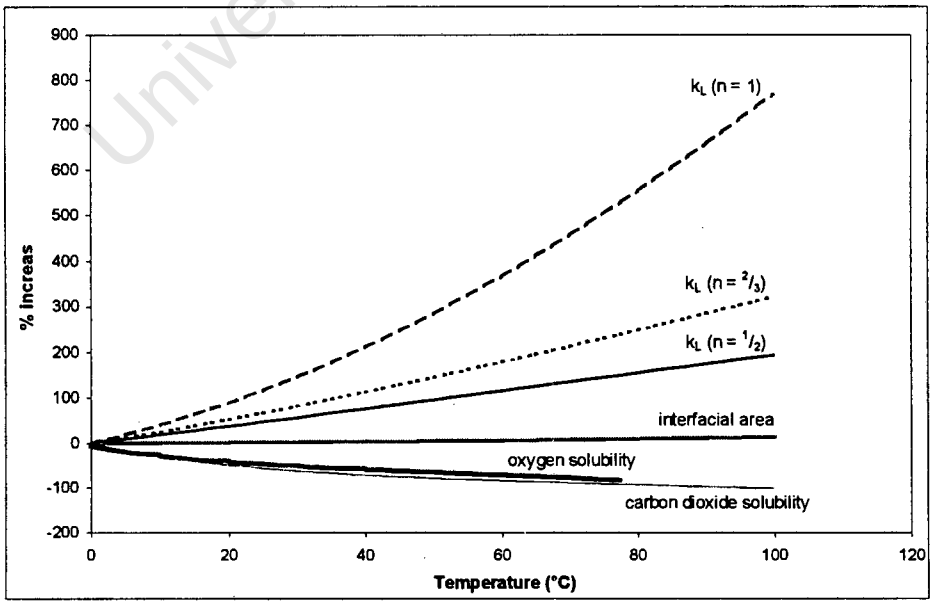


Figure 6-11: Effect of temperature on mass transfer domains; k_L, a and C.*

Further he found that the effect of temperature on interfacial area was insignificant when compared to that of the mass transfer coefficient, in accordance with Figure 6-11. Hence a similar correlation was proposed for the effect of temperature on the volumetric transfer coefficient (k_La) (Equation 6-33) where z is a constant between 1.020 and 1.024. Neither the surface mobility nor the accuracy of these correlations at high temperatures was reported. Jackson and Shen (1978) took a similar approach in the study of an impeller-sparger combination in a temperature range of 10-30°C. They correlated the effect of temperature on k_La with constant z between 1.016 and 1.037. Figure 6-12 illustrates this model for various values of z.

$$\frac{(k_La)_T}{(k_La)_{20}} = (z)^{(T-20)} \qquad \text{Equation 6-33}$$

where

T	Temperature	°C
$(k_La)_T$	volumetric mass transfer coefficient at T	
$(k_La)_{20}$	volumetric mass transfer coefficient at 20°C	

Work conducted by Boogerd *et al.* (1992) in the context of coal desulfurisation showed that the logarithm of k_La increased linearly with temperature in a very dilute sulphuric acid solution, 5mM H_2SO_4, pH 2. This was investigated under 3 different mixing conditions:

- mechanically well mixed, 700 rpm stirring speed, 7.5l/h air flow rate, 900 W/m^3 energy input
- gas-mixed, 0 rpm stirring speed, 75l/h air flow rate, 30 W/m^3 energy input
- mechanically, moderately mixed 200 rpm stirring speed, 7.5l/h air flow rate, 20 W/m^3 energy input

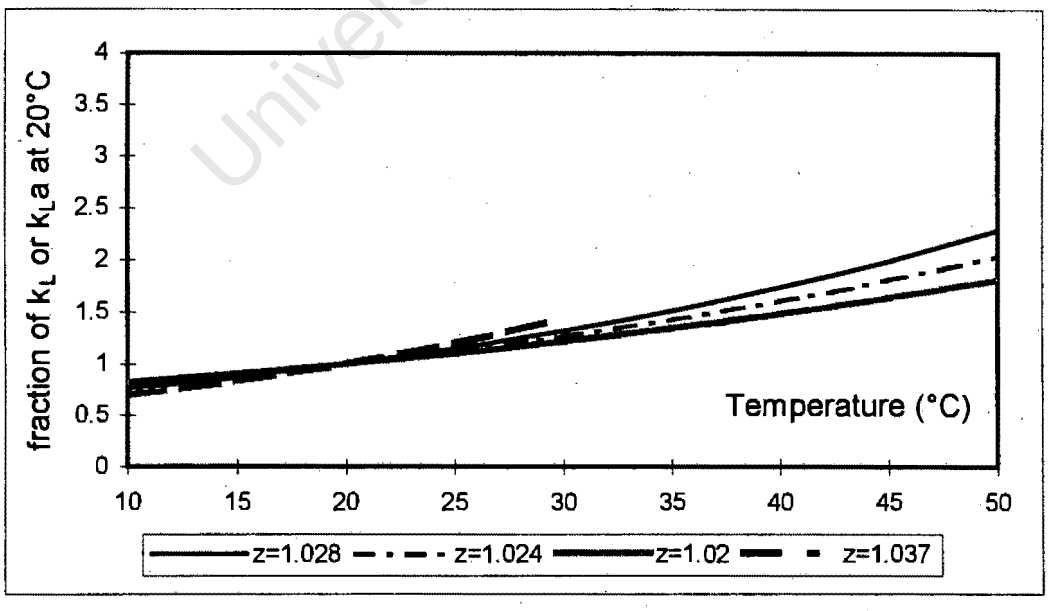


Figure 6-12: Barnhart (1969) and Jackson and Shen (1978) mass transfer coefficient correlations.

Table 6-3: Log(k_La) as a function of temperature (°C) for three mixing conditions (Boogerd et al., 1990).

	stirring speed	air flow rate	k_La correlation (T in °C)	z[†]
Condition 1	700 rpm	7.5 l/h	$\log k_L a = 0.0081 \cdot T - 0.144$	1.019
Condition 2	0 rpm	75 l/h	$\log k_L a = 0.0055 \cdot T - 0.809$	1.013
Condition 3	200 rpm	7.5 l/h	$\log k_L a = 0.0068 \cdot T - 1.290$	1.016

[†] assuming that Equation 6-35 is valid up to 70°C.

Table 6-3 contains correlations for the volumetric mass transfer coefficient (k_La) and z-factors for Equation 6-33. The values given for z in Table 6-3 fall within the range given by Jackson and Shen (1978) and fit the form of the k_La correlation originally given by Boogerd *et al.*, (1992) well, validating the assumption that Equation 6-33 is valid up to higher temperatures (>70°C). They concur that the negative effect of temperature on gas solubility is compensated for by the increase in volumetric mass transfer coefficient with temperature.

Clean bubbles are rarely achieved in industrial systems, hence it is safe to base a process design or understanding on contaminated interface behaviour (Moo-Young and Blanch, 1981; Oolman, 1984). Moving particles with rigid surfaces have k_L proportional to $(D_L)^{2/3}$ and to $(D_L)^{1/2}$ for moving spheres with mobile surfaces. This implies that the transfer coefficient for rigid bubbles is a stronger function of temperature than for mobile bubbles (Figure 6-11). The value of the transfer coefficient decreases during the transition from a mobile to rigid interface (Moo-Young and Blanch, 1981). However, the overall k_La in non-coalescing systems (rigid bubbles) is higher by a constant factor of two than that in coalescing systems (mobile bubbles) under the same aeration-agitation conditions. This implies that the rate of increase in the interfacial area is higher than the rate of decreases in k_L during the transition from a mobile to a rigid interface (Kargi and Moo-Young, 1985).

Figure 6-13 illustrates the importance of the prevailing hydrodynamic conditions. The three sets of conditions employed by Boogerd *et al.* (1992) (z=1.019, 1.016 and 1.103) display a different dependence on temperature. The mixing conditions affect the surface mobility (section 5.3) and the interfacial area. At high agitation intensities, turbulent flow, eddy diffusivity needs to be taken into account (Moo-Young and Blanch, 1981). Thus, k_L is a function of the sum of the contribution from molecular and eddy diffusivity (Brodkey and Hershey, 1988), including energy transfer from turbulent eddies. Without including the effect of hydrodynamics, it is evident that the general trend as predicted by the variation of molecular diffustivity with temperature (the grey lines of Figure 6-13) gives a fair indication of the variation of mass transfer in a non-coalescing, rigid system with temperature.

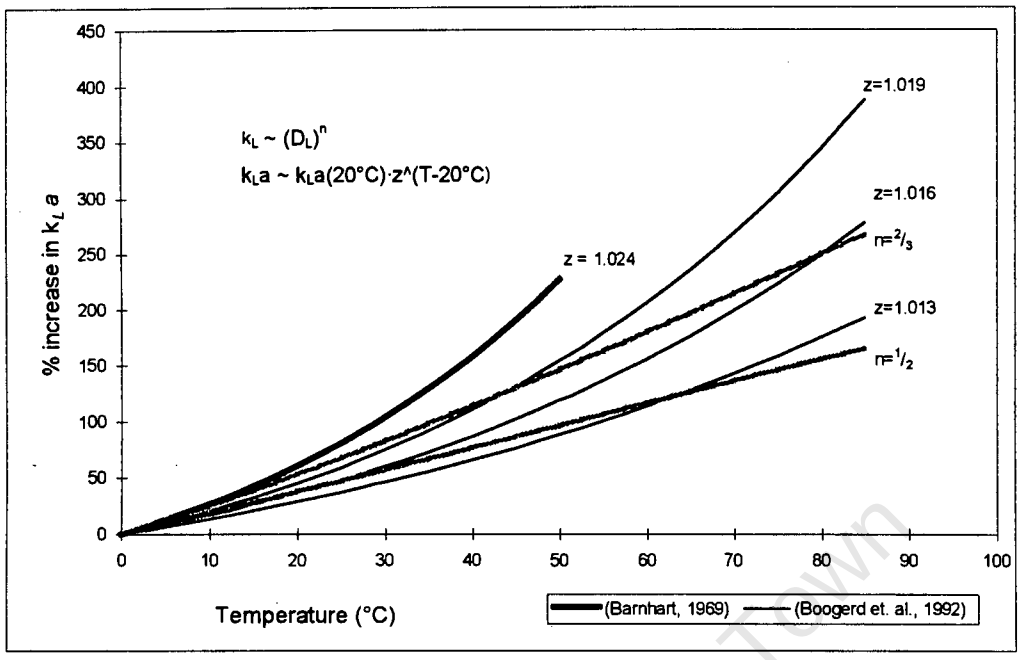


Figure 6-13: Mass transfer prediction at with temperature.

6.5 NOMENCLATURE

Symbol	Description	Units
A	total interfacial area in the gas-liquid dispersion	
C^*_A, C^*	equilibrium solute concentration	
C_A, C	solute concentration in the bulk liquid	
C_{el}	electrolyte concentration	$mol \cdot l^{-1}$
C_j, C_{jH_2O}	concentration of the gas, and concentration of the gas in water	$mol \cdot l^{-1}$
$D°_{AB}$	liquid phase diffusivity at infinite dilution	$cm^2.s^{-1}$
D_{AB}	diffusion coefficient of solute A in solvent B	
D_L	liquid phase diffusivity	cm^2s^{-1}
f	constant representing solute size	
H	Henry's law constant	atm
J_A	mass flux of component A in B	
K	overall transfer coefficient	
K_s	Sechenov constant	$l \cdot mol^{-1}$
L_S	latent heat of vaporisation of the solvent at normal boiling point	
M	molecular weight of the solvent	
M_A	molecular weight of solute A	$g.mol^{-1}$
M_B	molecular weight of solvent B	$g.mol^{-1}$
P	power input	
P_c	critical pressure	atm
Q	volumetric gas flowrate	$m^3.s^{-1}$
Q'	as defined in Equation 6-19	
Q_p	as defined in Equation 6-21	
R	universal gas constant	$atm.cm^3.mol.K$

Symbol	Description	Units
T	temperature	K or °C
T_b	normal boiling point	K
T_{br}	reduced boiling point	
T_c	critical temperature	K
T_r	reduced temperature	
V	molecular volume of the solute	$cm^3.mol^{-1}$
V_A	molal volume of solute A at its normal boiling temperature	$cm^3.mol^{-1}$
V_d	dispersion volume	
V_G	gas volume	
V_L	liquid volume	
V^{sat}	saturated liquid molar volume	$cm^3.mol^{-1}$
X	Stiel polar factor (Equation 6-23)	
$\frac{dC_A}{dx}$	concentration gradient	
$[P]$	parachor, temperature independent parameter	
$(ln\ U)^0, (ln\ U)^1$	functions of the reduced temperature	
We	Weber number	
a,b,c	normal regression constants	
d_b	bubble diameter	
d_m	maximum stable bubble diameter	
d_o	nozzle or orifice diameter	
d_p	diameter of primary bubbles	m
d_s	diameter of bubble which represents the whole distribution	
g	acceleration due to gravity	$m.s^{-2}$
k_L	mass transfer coefficient	
k_{LT}	mass transfer coefficient at temperature T	
k_{L20}	mass transfer coefficient at 20°C	
$(k_La)_T$	volumetric mass transfer coefficient at temperature T	
$(k_La)_T$	volumetric mass transfer coefficient at 20°C	
m	as defined in Equation 6-22	
m	mass of gas	g
n	index which ranges from 0.25 to 0.31	
p	partial pressure of the solute phase	atm
p_T	total pressure	1 atm
p_{vp_r}	reduced vapour pressure	
p^*_{WG}	vapour pressure of water	atm
u_n	nozzle gas velocity	
x	mole fraction of the solute in the liquid phase	
z	Barnhart (1969) and Jackson and Shen (1978) constant	
ϕ	association factor of solvent B	
λ	latent heat of vaporisation of the solute	$cal.g^{-1}$
μ_B	solvent B viscosity	
μ_L	liquid viscosity	cP
μ_G	gas viscosity	
τ	shear stress	

Symbol	Description	Units
ρ	density	$g.cm^{-3}$
ρ_A	solute density	$g.dm^3$
ρ_G	gas density	
ρ_{Lb}	molal liquid density at the normal boiling point	$g.cm^{-3}$
σ	surface tension	$dyne.cm^{-1}$
ϖ	accentric factor (Equation 6-24)	

6.6 REFERENCES

Atkinson, B. and F. Mavituna (1992): "Biochemical Engineering and Biotechnology Handbook", Chapter 12, Stockton Pr

Barnhart, E.L. (1969); "Transfer of oxygen in aqueous solutions", *J. San. Eng. Div. ASCE*, **95**(SA3), 645-661

Boogerd, F.C., P.Bos, J.G. Kuenen, J.J. Heijnen and R.G.J.M. van der Lans (1990): "Oxygen and Carbon Dioxide Mass Transfer and the Aerobic, Autotrophic Cultivation of Moderate and Extreme Thermophiles: A case study related to the microbial desulfurisation of coal", *Biotechnology and Bioengineering*, **35**, 111-1119

Brodkey, R.S. and H.C. Hershey (1988)): "Transport phenomena – A unified approach", McGraw-Hill Book Company, New York, p227-257

Cabani, S. and P.Gianni (1986): "in *Thermodynamic data for biochemistry and biotechnology*, Ed. H.J. Hintz, pp 261-262, Springer Verlag, Berlin; cited in Sandler (1989) p 439-440

Danckwerts, P.V. (1970): "Gas-Liquid Reactions", pp 6-20, McGraw-Hill Inc., United States of America

Danckwerts, P.V. and M.M. Sharma (1966): "The adsorption of carbon dioxide into solutions of alkalis and amines", *The Chemical Engineer*, October, CE244-CE280

Holman, J.P. (1989): "Heat Transfer, SI Edition", McGraw-Hill Book Company, Singapore

Jackson, M.L. and C. Shen (1978): "Aeration and Mixing in Deep Tank Fermentation Systems", *A.I.Ch.E. Journal*, **24**(1), 63-71

Joshi, J.B. and M.M. Sharma (1976): "Mass transfer characteristics of horizontal sparged contactors", *Trans. Instn Chem. Engrs*, **54**, 42-53

Kargi, F. and M. Moo-Young (1985): "Transport phenomena in bioprocesses", in *Comprehensive Biotechnology Volume 2*, Eds M. Moo-Young, C.L. Cooney and A.E. Humphrey, Pergamon Press, England, p5-56

Liley, P.E., R.C. Reid and E.Buck (1988): "Physical and chemical data", in *Perry's Chemical Engineers' Handbook, 6th Edition*, Eds R.H. Perry, D.W. Green and J.O. Maloney", McGraw-Hill Book Company, United States of America, p 3-273-274, 3-278-282, 3-285-289

Long, F.A. and W.F. McDevit (1952): "Activity Coefficients of non-electrolyte solutes in aqueous salt solutions", *Chem. Rev.*, 51, 119-150

Moo-Young, M. and H.W. Blanch (1981): "Design of Biochemical Reactors, Mass transfer criteria for simple and complex systems", *Advances in Biochemical Engineering*, **19**, 1-70, Springer Verlag, Berlin

Oolman, T. (1984): "Bubble coalescence and mass transfer in air sparged bioreactors", Ph.D. Thesis, University of California, Berkeley

Ratcliff, G.A. and J.G. Holdcroft (1963): "Diffusivities of gases in aqueous electrolyte solutions", *Trans. Instn Chem. Engrs*, **41**, 315-319

Reid, R.C., J.M. Prausnitz and T.K. Sherwood (1977): "The Properties of Gases and Liquids, 3rd Edition", McGraw-Hill Book Company, United States of America

Sandler, S.I. (1989): "Chemical and Engineering Thermodynamics, 2nd Edition", John Wiley & Sons, Canada

Schumpe, A., G. Quicker and W.D. Deckwer (1982): "Gas Solubilities in Microbial Culture Media", *Advances in Biochemical Engineering*, **24**, 1-38, Springer Verlag, Berlin

Schumpe, A., I. Adler and W. –D. Deckwer (1978): "Solubility of oxygen in electrolyte solutions", *Biotechnology and Bioengineering*, **20**, 145

Weast, R.C. and M.J. Astle (1982): "Handbook of Chemistry and Physics 63rd Edition", CRC Press Inc.. Florida

Wilhelm, E., R. Battino and R.J. Wilcock (1977): "Solubility of Gases in Water", *Chemical Reviews*, 77, 219-262

Winkler, M. (1981): "Biological Treatment of Waste Water", pp 42-76, Ellis Horwood Ltd. Publishers, Chicester

7. Thermodynamics of Aqueous Solutions

The bioleaching process is considered to be a two step process (2.5). In the second step ferric iron acts as a chemical leaching agent. In the iron sulphate system found in the bioleaching chemical environment, a large portion of free ferric iron forms complexes (Nagpal and Dahlstrom, 1994). As equilibrium constants generally increase with temperature, one would expect the extent of speciation to increase with temperature. A corresponding decrease in the rate of chemical leaching is expected to result owing to the reduced free ferric iron concentration. For this reason it is necessary to understand the thermodynamics of aqueous bioleaching solutions.

Barret *et al.* (1993), discusses the importance of sulphur (VI), arsenic(III), arsenic(V), iron(II), iron (III) and ions of other solubilised metals in bioleaching. Further ions are produced by hydrolysis and complex formation. Dry (1984) and Crundwell (1988) published equilibrium data for several pertinent components as a function of temperature. Using this information Dry and Crundwell included the formation of several ionic species in their calculations at conventional bioleaching temperatures. As the availability of thermodynamic data for aqueous species is limited, the prediction of speciation at elevated temperatures is hampered. The Criss Cobble correlation predicts heat capacity data for ionic species (Criss and Cobble, 1964a). As a result Gibbs free energy data can be estimated at elevated temperatures, enabling the estimation of equilibrium constants for reactions with ionic components at various temperatures (Barner and Scheuerman, 1978). This chapter details the thermodynamics behind these concepts.

7.1 METAL IONS IN SOLUTION

Water molecules surround metal cations in accordance with their co-ordination chemistry. Co-ordination bonds attach these water molecules to the metal cation in what is known as the inner co-ordination sphere. Outside the sphere, water molecules are held loosely. They become less and less organised further towards the bulk solvent (Bryson and Nicol, 1996). Interactions between the cation and any ligands can be in either the outer or inner sphere of the complex. This determines the mechanism of electron transfer in redox reactions (Burgess, 1988). Furthermore, the stability of aqueous species varies according to whether it is an inner or outer sphere complex. Outer sphere complexes are generally bonded by weaker forces than inner sphere complexes (Bryson and Nicol, 1996; Dry, 1996). As a result, inner sphere complexes are more stable.

Certain ligands are able to form complexes with metal ions in which they penetrate only as far as the outer co-ordination sphere, leaving the inner sphere untouched *e.g.* sulphate (SO_4^{2-}). The forces that hold outer sphere complexes together are mainly electrostatic, sometimes with the aid of hydrogen bonding (as is the case with sulphate). Other groups or ligands, for example NH_3 or CN^-, may replace the water molecules in the inner sphere. In the ligand, the atoms that are directly attached to the metal atom are known as the donor atoms (Bryson and Nicol, 1996).

7.2 STANDARD STATES

Standard states are important in thermodynamics as several quantities, including enthalpy and Gibbs free energy are given relative to a reference state (Sandler, 1989). The standard states for a pure liquid or solid is the substance at the specified temperature under a pressure of 1 atm. For a gas, the standard state is the hypothetical ideal gas at the specified temperature and a fugacity of 1 atm. Since the behavior of most gases is nearly ideal at low pressure, this state approximates that of the real gas at the specified temperature and 1 atm pressure (Barner and Scheuerman, 1978). The reference temperature is often taken to be 298K.

The standard state for a dissolved species in aqueous solution is defined as the hypothetical ideal solution at a molality of 1. This state is arrived at by extrapolation from the behaviour in infinitely dilute solutions where behaviour is ideal (Barner and Scheuerman, 1978). Enthalpy and Gibbs free energy of formation for individual ions are based on the convention that the enthalpy and free energy of formation of the hydrogen ion are zero at all temperatures (Bryson and Nicol, 1996; Barner and Scheuerman, 1978).

7.3 ENTHALPY

Under given conditions of temperature, pressure and specific volume, the internal energy of a substance is a unique property of that substance (Thompson and Ceckler, 1978). Enthalpy is a defined energy measure related to internal energy by Equation 7-1. Enthalpy is conventionally represented at constant volume, and hence a function of temperature and pressure (Sandler, 1989).

$$H = U + P \cdot V \qquad \textit{Equation 7-1}$$

where

H	enthalpy
U	internal energy
P	pressure
V	volume

The enthalpy of formation of a substance represents the change in enthalpy when 1 mole of the substance in its standard state is formed at the specified temperature from the elements, each in its standard reference state (Barner and Scheuerman, 1978). The molar enthalpy of a species (denoted by subscript *i*) at temperature *T* may be expressed as in Equation 7-2. As pressure changes in bioleaching are minimal, ΔH_i^1 can be neglected (Bryson and Nicol, 1996).

$$H_i(T) = H_i^o(T^o) + \int_{T^o}^{T} C_{Pi}^o(T)dT + \Delta H_{Vi} + \Delta H_i^1 \qquad \textit{Equation 7-2}$$

where

$H_i^o(T^o)$	is the molar enthalpy of substance i at some specified base condition
$C_{Pi}^o(T)$	is the molar specific heat of substance i at the reference pressure (P°)
ΔH_{Vi}	enthalpy associated with a change in phase
ΔH_i^1	effect of a change in pressure on the enthalpy (over a specific pressure interval)

7.3.1 ENTHALPY OF MIXTURES

Th effect of mixtures on enthalpy could be significant for concentrated solutions (Bryson and Nicol, 1996). For a mixture of k components, the total enthalpy for the mixture will be

$$H_{mix}(T) = \sum_{i=0}^{k} n_i H_i(T) + \Delta H_m \qquad \textit{Equation 7-3}$$

where

n_i	number of moles of substance I
ΔH_m	effect of various components on the heat of mixing total enthalpy at temperature T and pressure P
k	number of components in the mixture

7.4 HEAT CAPACITY

Heat capacity (C_P) is defined as the amount of heat required to raise the temperature of unit mass of any substance by one degree Kelvin, at constant pressure (Thompson and Ceckler, 1978). Heat capacity data is used when converting standard state (298K) thermodynamic data to higher temperatures.

$$\begin{aligned} Q &= n_i C_{Pi} \Delta T \\ dQ &= n_i C_{Pi}(T) dT \\ Q &= n_i \int_{T}^{T_f} C_{Pi}(T) dT \end{aligned} \qquad \textit{Equation 7-4}$$

where

Q	heat
n_i	number of moles of substance i
C_{Pi}	heat capacity of substance i
T_f *and* T	final and initial temperatures

7.4.1 HEAT CAPACITY DATA

Heat capacity data is experimentally determined. It is a function of temperature and is given at constant pressure. It is often predicted using a power series (Laidler and Meiser, 1995). The format implemented in the simulation is given below. The coefficients (a, b, c, d and e) are stored such that the units of heat capacity are J/mol·K.

$$C_P^o(T) = a + b \cdot T + \frac{c}{T^2} + d \cdot T^2 + e \cdot T^3 \qquad \textit{Equation 7-5}$$

A mean heat capacity ($\overline{C}_P^o(T)$) is useful as it simplifies several thermodynamic calculations without forfeiting accuracy presuming that the calculations are carried out over a narrow temperature range (< 80°C) (Bryson and Nicol, 1996).

The mean heat capacity is determined by:

$$\overline{C}_P^o(T) = \frac{1}{T_f - T}\int_T^{T_f} C_P^o dT$$

$$= \frac{1}{T_f - T}\left\{a(T_f - T) + \frac{b}{2}\left(T_f^{\,2} - T^2\right) - \frac{c}{(T_f - T)} + \frac{d}{3}\left(T_f^{\,3} - T^3\right) + \frac{e}{4}\left(T_f^{\,4} - T^4\right)\right\}$$

Equation 7-6

7.5 EQUILIBRIUM

With aqueous solutions of electrolytes we have two types of equilibrium to consider: phase equilibrium and chemical or ionic reaction equilibrium. Phase equilibria of interest are primarily vapour-liquid or solid-liquid (Zemaitis *et al.*, 1986). The necessary condition of phase equilibrium is that the chemical potential of any species i in phase a ($\mu_{i,a}$) is equal to the chemical potential of that same species i in phase b ($\mu_{i,b}$) or

$$\mu_{i,a} = \mu_{i,b} \qquad \textit{Equation 7-7}$$

For chemical or ionic equilibrium in a particular phase, the condition of equilibrium is of the same form as the chemical equation. Thus if the reaction at equilibrium is represented by

$$aA + bB = cC + dD \qquad \textit{Equation 7-8}$$

the condition of chemical equilibrium in a particular phase would be denoted by

$$\sum n_i \mu_i = 0 \qquad \textit{Equation 7-9}$$

where n_i is the stoichiometric coefficient of species i in the reaction of interest (Sandler, 1989).

The conditions of equilibrium for phase and chemical equilibrium can be combined to represent the heterogenous liquid-solid equilibrium of an aqueous solution of a salt B in equilbrium with the solid of salt B

$$\mu_{B,s} = \mu_{B,aq} \qquad \textit{Equation 7-10}$$

The chemical potential of the solid crystal salt B is in phase equilibrium with the dissolved salt B in the liquid. Strong electrolytes in water dissociate completely to the constituent cations and anions of the salt. Thus the condition of equilibrium for a strong electrolyte dissolved in water and in equilibrium with its crystalline phase becomes (Zemaitis *et al.*, 1986):

$$\mu_{B,aq} = \upsilon_c \mu_c + \upsilon_a \mu_a \qquad \textit{Equation 7-11}$$

where

υ_c	chemical potential of c
υ_a	chemical potential of a

7.6 ACTIVITIES OF CHEMICAL SPECIES

The activity of a species, a_i, is dimensionless. It describes the chemical potential of that species (Zemaitis *et al.*, 1986) and is defined by

$$a_i = \frac{f_i}{f_i^o} \qquad \textit{Equation 7-12}$$

where

f_i fugacity of component *i*
f_i° fugacity of component *i* at standard state

Further

$$a_i = \gamma_i \frac{m_i}{f_i^o} \qquad \textit{Equation 7-13}$$

where

m_i molality of component *i*
γ_i activity coefficient of component *i*

An error of approximately 1% is introduced when the molality is replaced with the concentration of the species (C_i). Therefore

$$a_i \approx \gamma_i C_i \qquad \textit{Equation 7-14}$$

is an accurate estimation of the activity of chemical species *i* (Bryson and Nicol, 1996), where C_i is the concentration of i.

7.6.1 ACTIVITY COEFFICIENTS

Models have been proposed for predicting activity coefficients of ions in solution. These models can be either theoretical or empirical. Such estimation of activity coefficients allows the estimation of the non-ideality of a solution by calculating the activities (Atkins, 1983). Activity coefficients of ions are determined by the interactions of that ion with each oppositely charged ion and molecule present in the solution (Zemaitis *et al.*, 1986). In hydrometallurgical or biohydrometallurgical processes, this needs to take into account a large number of components. In order to simplify the calculation of activity coefficients, it has been assumed that the dissolved electrolytes in water completely dissociate into ions (Zemaitis *et al.*, 1986).

Debye-Hückel Model

Electrolyte solutions are particularly non-ideal as there are substantial electrostatic interactions between the ions. One of the first theoretical models developed, the Debye-Hückel model, takes these coloumbic interactions into account and provides a platform upon which the effect of ionic changes can be investigated (Atkins, 1983). Their expression for γ_i is given by:

$$-\log_{10}\gamma_i = \frac{Az_i^2\sqrt{I}}{1+d_iB\sqrt{I}} \qquad \textit{Equation 7-15}$$

where

A and B	Debye-Hückel constants at the specified temperature and pressure	
I	ionic strength of the solution	mol of ion/l
z_i	charge on the species	
d_i	effective ionic diameter	Å

The ionic strength is given by

$$I = \frac{1}{2}\sum m_i z_i^2$$
$$\approx \frac{1}{2}\sum C_i z_i^2 \qquad \textit{Equation 7-16}$$

where

m_i	molality of species i
C_i	concentration of species i

Although the Debye-Hückel model was originally derived for dilute solutions ($I < 0.1$), some success has been reported in applying it at higher concentrations, particularly when ion-pair equilibria are taken into account (Dry and Bryson, 1988). Details of the Debye-Hückel model and coefficients used by Dry and Crundwell are contained in Appendix D.

Researchers working with Pitzer published a series of papers expanding the Debye-Hückel model, whereby terms were added to account for the ionic strength dependence of the short-range forces effected in binary interactions (Zemaitis *et al.*, 1986). Study of the bisulphate ion is of interest in bioleaching (Pitzer *et al.*, 1977). Details of this calculation are contained in Appendix E.

7.7 REDOX POTENTIAL AND THE NERNST EQUATION

Redox (Eh) measurements are taken in both research and industrial bioleaching applications. These measurements pertain to the electrochemical potential of a solution. Using the Nernst equation (Equation 7-17) they can be interpreted to deduce the activities or concentrations of the redox couple ions, *e.g.* ferric and ferrous iron. The ratio of the electrolyte couple is in terms of the activities of the free components (Atkins, 1983). This constitutes another reason why it is important to understand the speciation of soluble iron in electrolyte systems.

$$E = E_o + \frac{RT}{nF}\cdot\ln\left(\frac{a_i}{a_j}\right)$$
$$E_o = -\frac{\Delta G^\circ}{nF} \qquad \textit{Equation 7-17}$$

where

E	electrochemical or redox potential of the solution (Eh)	V
E_o	Eh at standard state	V
n	stoichiometric coefficient	
F	Faradays constant	96 483.5 J/V·mol

R	Universal Gas constant	8.314 39 J/mol·K
$\Delta G°$	Gibbs free energy	J/mol
i and j	denotes ionic species of the redox couple, for example Fe^{3+} and Fe^{2+} respectively	

7.8 GIBBS FREE ENERGY CHANGE FOR A REACTION

Gibbs free energy (G) has been defined by

$$G = H - TS \qquad \textit{Equation 7-18}$$

where S is entropy, measured in J/mol·K. At constant temperature

$$dG = dH - T \cdot dS \qquad \textit{Equation 7-19}$$

Gibbs free energy minimisation can be employed in solving for equilibrium conditions in reaction systems (I and Nancollas, 1977). For example take reaction:

$$aA + bB \leftrightarrow cC \qquad \textit{Equation 7-20}$$

At chemical equilibrium, the change in Gibbs free energy for the reaction is zero:

$$\Delta G_R(T) = 0 \qquad \textit{Equation 7-21}$$

Hence the equilibrium constant can be determined by calculating the change in Gibbs free energy for the reaction (Sandler, 1989).

$$\Delta G_R(T) = \Delta G_R^o(T) + RT \ln\left[\frac{(a_c)^c}{(a_a)^a (a_b)^b}\right]$$

$$\frac{(a_c)^c}{(a_a)^a (a_b)^b} = \exp\frac{-\Delta G_R^o(T)}{RT} = K \qquad \textit{Equation 7-22}$$

where

R	universal gas constant
K	equilibrium constant for the reaction
(a_i)	activity of species i
a, b and c	stoichiometric coefficients of component A, B and C respectively

7.8.1 GIBBS-HELMHOLTZ EQUATION

Rewriting the definition of Gibbs free energy (Equation 7-19),

$$G = H - TS$$

$$= H + T\left(\frac{\partial G}{\partial T}\right)_P \qquad \textit{Equation 7-23}$$

where P denotes constant pressure. Further,

$$\left(\frac{\partial G}{\partial T}\right)_P = \frac{-H}{T^2} \qquad \textit{Equation 7-24}$$

The Gibbs-Helmholtz expression is useful in calculating the effect of temperature on equilibrium constants (Zemaitis *et al.*, 1986).

7.9 THERMODYNAMIC PROPERTIES AT ELEVATED TEMPERATURES

Thermodynamic properties are related to temperature using heat capacities or mean heat capacities (Sandler, 1989).

$$\Delta G^o(T) = \Delta H^o(T) - T\Delta S^o(T)$$
$$\Delta H^o(T) = \Delta H^o(T^o) + \int_{T^o}^{T} \Delta C_P^o(T)dT \qquad \textit{Equation 7-25}$$
$$\Delta S^o(T) = \Delta S^o(T^o) + \int_{T^o}^{T} \Delta C_P^o(T)dT$$

Replacing $\Delta C_P^o(T)$ by a mean heat capacity and combining the above equations for the Gibbs free energy of reaction (Equation 7-19) (Bryson and Nicol 1996):

$$\Delta G_{rxn}^o(T) = \Delta G_{rxn}^o(T^o) - (T - T^o)\Delta S_{rxn}^o(T^o) + (T - T^o)\Delta \overline{C}_P^o(T) - T\left(\ln\frac{T}{T^o}\right)\Delta \overline{C}_P^o(T)$$

Equation 7-26

7.10 CRISS COBBLE CORRELATION

As the mean heat capacity is an experimentally determined quantity, it cannot be measured for aqueous species. The Criss Cobble correlation is a method for predicting the mean heat capacity of an ionic quantity and subsequently various other thermodynamic data for the substance (Bryson and Nicol, 1996). It relies on the 'correspondence principle' summarised as follows:

> "A standard state can be chosen at every temperature such that the partial molal entropies of one class of ions at that temperature are linearly related to the corresponding entropies at some reference temperature (Criss and Cobble, 1964a)."

The reference temperature is taken to be 298K. The correlation thus relies on the implementation and comparison of a true or conventional entropy ($S°$) and absolute entropies (S_a^o) as follows:

$$S_a^o(T^o) = S^o(T^o) - 20.9z \qquad \textit{Equation 7-27}$$

where z is the ionic charge. The choice of standard state for absolute entropy corresponds to an ionic entropy of 20.9 J/mol·K for the hydrogen ion (Criss and Cobble, 1964a; Criss and Cobble, 1964b).

$$S_a^o(T) = a(T) + b(T)S_a^o(T^o) \qquad \textit{Equation 7-28}$$

where

$S_a^o(T)$	standard absolute entropy at temperature T	J/mol·K
$a(T)$	Criss Cobble constant	J/mol
$b(T)$	Criss Cobble constant	
T^o	reference temperature (298 K)	K

The absolute entropy at any temperature is determined in accordance with fundamental thermodynamics as follows:

$$S_a^o(T) = S_a^o(T^o) + \int_{T^o}^{T} \frac{C_P^o}{T} dT$$

$$\approx S_a^o(T^o) + \overline{C}_P^o \int_{T^o}^{T} \frac{1}{T} dT \qquad \textit{Equation 7-29}$$

where

C_P^o	substance heat capacity	J/mol·K
$\overline{C}_P^o$	substance mean heat capacity	J/mol·K

Hence:

$$\overline{C}_p^{\,o} = \frac{S_a^o(T) - S_a^o(T^o)}{\ln \frac{T}{T^o}} \qquad \textit{Equation 7-30}$$

which on rearrangement becomes

$$\overline{C}_P^o(T) = \alpha(T) + \beta(T) S_a^o(T^o) \qquad \textit{Equation 7-31}$$

where

$$\alpha(T) = \frac{a(T)}{\ln\left(\frac{T}{T^o}\right)} \qquad \textit{Equation 7-32}$$

and

$$\beta(T) = \frac{-(1 - b(T))}{\ln\left(\frac{T}{T^o}\right)} \qquad \textit{Equation 7-33}$$

7.10.1 EQUILIBRIUM CONSTANTS FROM THE CRISS AND COBBLE CORRELATION

Figure 7-1 presents equilibrium constants and experimental data taken from the JESS database (May and Murray, 1991) for the formation of the bisulphate ion.

$$H^+ + SO4^{2-} \leftrightarrow HSO4^-$$

$$K = \exp\left[\frac{-\Delta G_R^o}{R \cdot T}\right] \qquad \textit{Equation 7-34}$$

The trend of the experimental data is indicated by a 'line of best fit'. The Criss and Cobble correlation accurately predicts the real data in the region of interest (0-100°C).

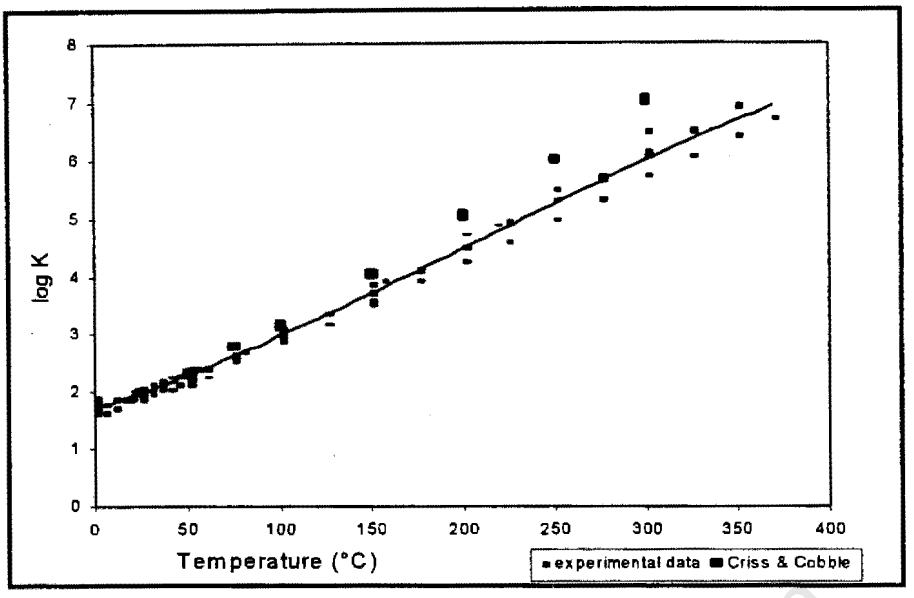


Figure 7-1: Criss Cobble equilibrium constant prediction.

7.10.2 CRISS COBBLE CONSTANTS

Original data from Criss and Cobble (1964a, 1964b) is presented in Appendix F. It was experimentally obtained at 200°C and below, and extrapolated, based on the assumption that the trend will remain linear, above 200°C. In order to use this empirical correlation at different temperatures it was assumed that the whole range was linear, and a linear equation was fitted to the data as depicted in Figures 7-2 and 7-3. These coefficients are tabulated in Table 7-1. For ease of use, these a(T) and b(T) constants are converted to $\alpha(T)$ and $\beta(T)$ coefficients according to Equation 7-32. Zemaitis *et al.* (1986) state that data for the $\alpha(T)$ and $\beta(T)$ coefficients above 200°C have recently become available. This data has shown that extrapolation above 200°C is often quite inaccurate.

The mean heat capacity of the hydrogen ion (H^+) can be estimated using the standard state absolute entropy of the ion at that temperature according to Equation 7-21. Criss and Cobble (1964a, 1964b) presented data for this. Good agreement is found in the region of 0-200°C when using this approach to predict the mean heat capacity, as illustrated in Figure 7-4. Appendix F contains a sample calculation and further details of the constants.

Table 7-1: Criss Cobble regression coefficients.

Description	slope (m)	y intercept (c) (Kelvin scale)
cations a(T)	0.5553	-165.4794
cations b(T)	-0.0017	1.5066
simple anions a(T)	-0.731	217.838
simple anions b(T)	-0.00007	1.02086
oxy-anions a(T)	-1.6084	479.3032
oxy-anions b(T)	0.0058	-0.7284
acid oxy-anions a(T)	-1.6736	498.7328
acid oxy-anions b(T)	0.0112	-2.3376
absolute entropy (J/mol·K) of H^+	0.3954	-140.439
heat capacity (J/mol·K) of H^+	0.1729	55.242

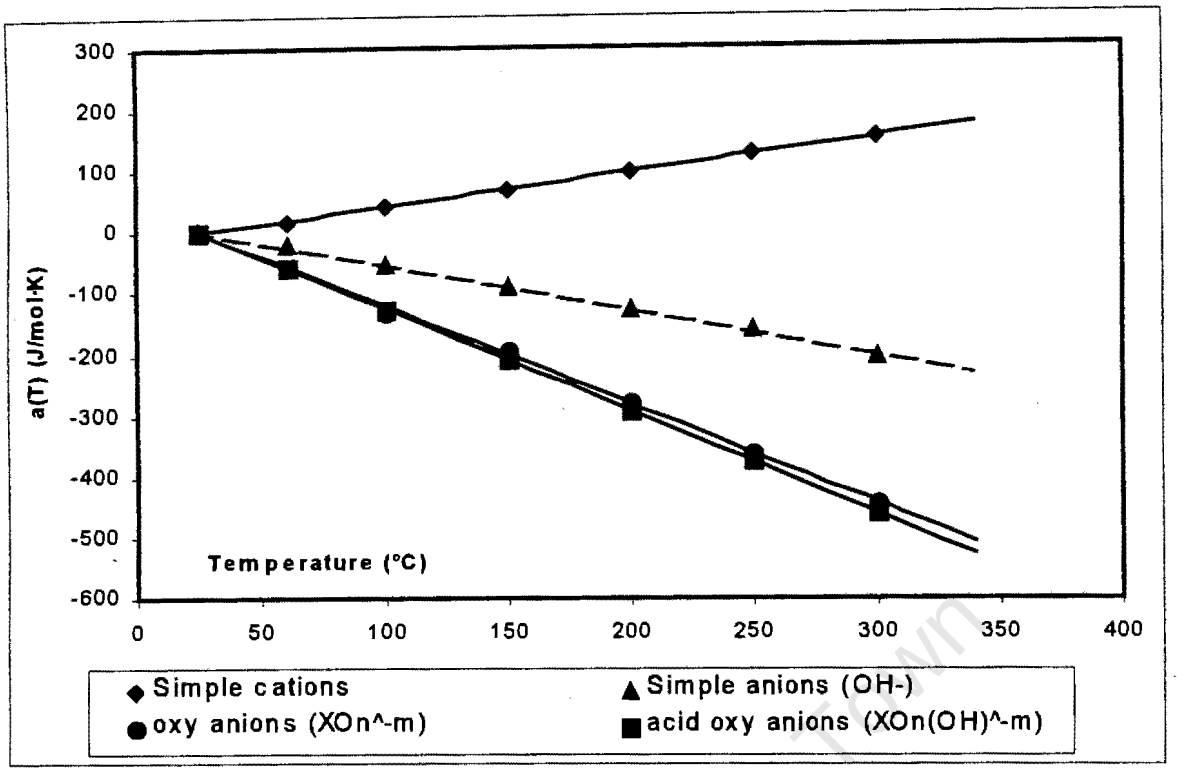


Figure 7-2: Criss Cobble (a(t) constants (Criss and Cobble, 1964a; 1964b).

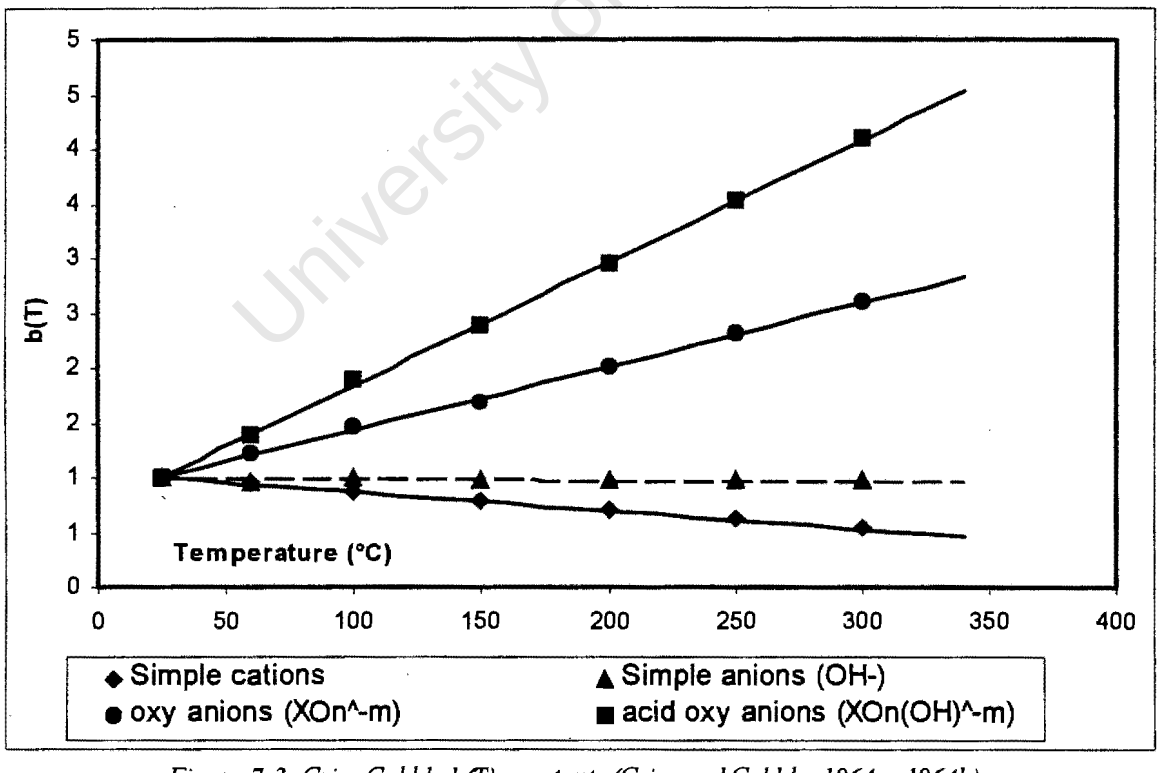


Figure 7-3: Criss Cobble b(T) constants (Criss and Cobble, 1964a; 1964b).

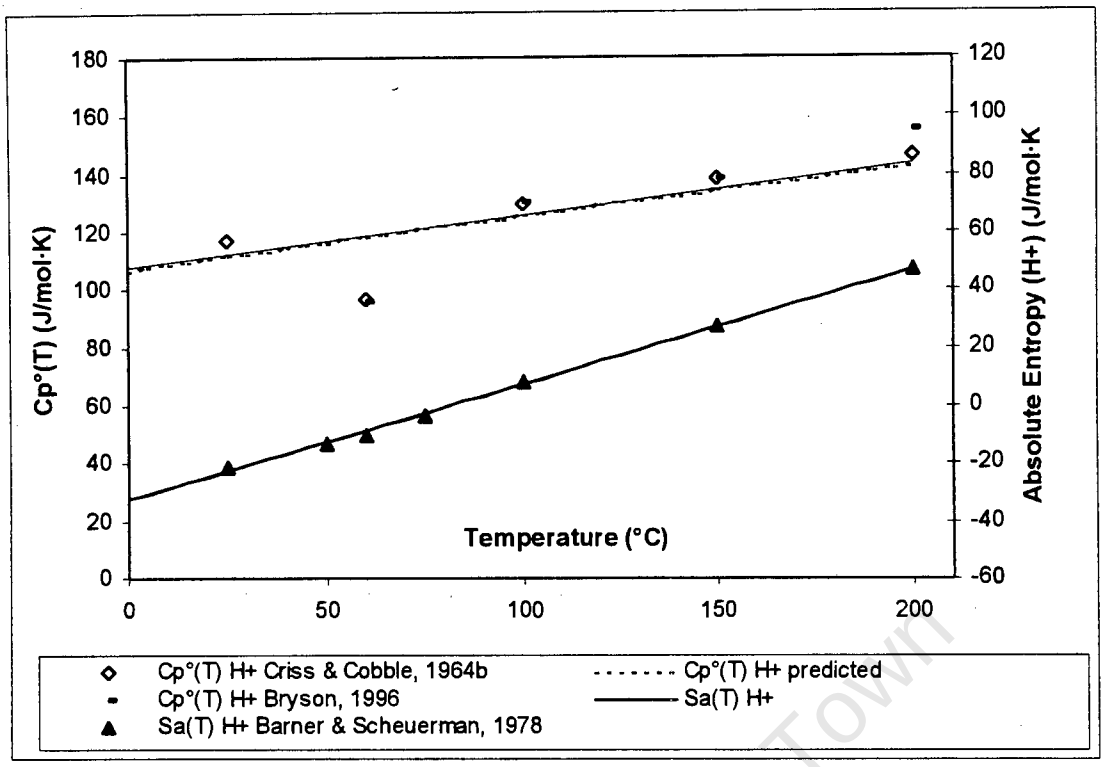


Figure 7-4: Heat capacity and absolute entropy of the hydrogen ion at different temperatures.

7.11 NOMENCLATURE

Symbol	*Description*	*Units*
A and B	Debye-Hückel constants at the specified temperature and pressure	
C_i	concentration of species i	
$C^o_{Pi}(T)$	molar specific heat of substance i at the reference pressure (P°) and temperature T	J/mol·K
E	electrochemical or redox potential of the solution (Eh)	V
E_o	Eh at standard state	V
F	Faradays constant - 96 483.5 J/V·mol	
$\Delta G°$	Gibbs free energy	J/mol
H	enthalpy	kJ/mol
$H^o_i(T^o)$	is the molar enthalpy of substance i at some specified base condition	
$H_{mix}(T)$	total enthalpy for a mixture	
ΔH_{Vi}	enthalpy associated with a change in phase	
ΔH^1_i	effect of a change in pressure on the enthalpy (over a specific pressure interval)	
ΔH_m	effect of various components on the total enthalpy at temperature T and pressure P	
I	ionic strength of the solution	mol of ion/l
K	equilibrium constant for the reaction	
P	pressure	
Q	heat	
R	Universal gas constant – 8.31439 J/mol·K	
$S^o_a(T)$	standard absolute entropy at temperature T	J/mol·K
T	Temperature	K

Symbol	*Description*	*Units*
T_f and T	final and initial temperatures	K
T°	reference temperature (298 K)	K
U	internal energy	
V	volume	
a, b, c, d and e	coefficients	
a_i	activity of species i	
a(T)	Criss Cobble constant	J/mol
b(T)	Criss Cobble constant	
d_i	measure of the effective ion diameter	Å
f_i	fugacity of i	
f_i°	fugacity of i at standard state	
i and j	denotes ionic species of the redox couple, for example Fe^{3+} and Fe^{2+} respectively	
k	number of components in the mixture	
m_i	molality of species i	
n	stoichiometric coefficient	
n_i	number of moles of substance i	
z_i	charge of species i	
$\alpha(T)$	Criss Cobble coefficient	
$\beta(T)$	Criss Cobble coefficient	
γ_i	activity coefficient of species i	
$\mu_{i,\theta}$	chemical potential of species i in phase a	
υ_i	chemical potential of *i*	

7.12 REFERENCES

Atkins, P.W. (1983): "Physical Chemistry", Oxford University Press, Oxford

Barner, H.E. and R.V Scheuerman (1978): "Handbook of thermochemical data for compounds and aqueous species", John Wiley & Sons, New York

Barret, J., M.N. Hughes, G.I. Karavaiko and P.A. Spencer (1993): "Metal extraction by bacterial oxidation of minerals", Ellis Horwood, England, p73-101

Boon, M., G.S. Hansford and J.J. Heijnen (1995): "The role of bacterial ferrous oxidation in the bio-oxidation of pyrite", *Biohydrometallurgical Processing, Volume I*, Eds T. Vargas, C.A. Jerez, J.V. Wiertz and H. Toledo, University of Chile

Bryson, A.W. and M. Nicol (1996): "Thermodynamics of Aqueous Solutions", Continuing Engineering Education Course, University of Witwatersrand, South Africa

Burgess, J. (1988): "Ions in Solution: basic principles of chemical interactions", Ellis Horwood Limited, England

Criss, C.M. and J.W. Cobble (1964a): "The thermodynamic properties of high temperature aqueous solutions. IV. Entropies of the Ions up to 200°C and the correspondence principle.", *J. Amer. Chem. Soc.* **86** 5390-5393

Criss, C.M. and J.W. Cobble (1964b): "The thermodynamic properties of high temperature aqueous solutions. V. The calculations of ionic heat capacities up to 200°C. Entropies and heat capacities above 200°C", *J. Amer. Chem. Soc.* **86** 5390-5393

Crundwell, F.K. (1988): "The role of charge-transfer in the oxidative and non-oxidative dissolution of sphalerite", PhD Thesis, University of Witwatersrand, South Africa

Dry, M.J. (1996): Personal Conversation, MINTEK, Randburg, South Africa

Dry, M.J. and A.W. Bryson (1988): "Prediction of Redox Potential in Concentrated Iron Sulphate Solutions", *Hydrometallurgy*, **21**, 59-72

Dry, M.J. (1984): "Kinetics of Leaching of a Low Grade Matte in Ferric Sulphate Solution", PhD Thesis, University of Witwatersrand, South Africa

I, T. and G.H. Nancollas (1972): "EQUIL - A General Computational Method for the Calculation of Solution Equilibria", *Analytical Chemistry*, **44** (12) 1940-1950

Laidler, K.J. and J.H. Meiser (1995): "Physical Chemistry, Second Edition", Houghton Mifflin Company, Boston

May, P.M. and K. Murray (1991): "JESS, a joint expert speciation system – II. The thermodynamic database", *Talanta*, **38**(12), 1419-1426

Nagpal, S. and D. Dahlstrom (1994): "A mathematical model for the bacterial oxidation of a sulfide ore concentrate.", *Biotech.Bioeng.* **43** 357-364

Pitzer, K.S., R.N. Roy and L.F. Silvester (1977): "Thermodynamics of Electrolytes. 7. Sulfuric Acid", *Journal of the American Chemical Society*, **99** (15) 4930-4936

Sandler, S.I. (1989): "Chemical and Engineering Thermodynamics, Second Edition", John Wiley & Sons, New York

Thompson, E.V. and W.H. Ceckler (1978): "Introduction to Chemical Engineering", McGraw-Hill Book Company, London, pp57-65,

Zemaitis, J.F. Jr; D.M. Clark, M. Rafal and N. C Scrivner (1986): "Handbook of Aqueous Electrolyte Thermodynamics", American Institute of Chemical Engineers, New York, pp3-43

8. Speciation Evaluation

The extent of formation of a particular complex is dependent on the equilibrium constant of the formation reaction (Section 7-8). As equilibrium constants vary with temperature, one can expect speciation to vary with temperature. Speciation is of importance to bioleaching as there is evidence that the rate of bioleaching can be related to the concentration of free ferric iron concentration (Lacey and Lawson, 1970) or redox potential (Breed *et al.*, 1997). Various authors have considered a selection of speciation reactions when modelling bioleaching at conventional temperatures (Dry, 1984; Crundwell, 1988, Barret *et al.*, 1993; Nagpal and Dahlstrom, 1994; Smith *et al.*, 1988).

8.1 EQUILIBRIUM CONSTANTS

True thermodynamic equilibrium constants are calculated at zero ionic strength (Section 7-6.1). Using activity coefficients these equilibrium constants are adjusted for solutions of differing ionic strengths (Dry and Bryson, 1988). Equilibrium constants are dimensionless. They are a re-expression of Gibbs free energy and are expressed in terms of a ratio of activity coefficients (Section 7-8). For ease of use, pseudo-equilibrium constants are used. The ratio of activity coefficients is calculated and, as a result, equilibrium constants can be expressed in terms of a ratio of concentrations (Dry, 1984).

$$K' = \prod_i \gamma_i^{n_i} \cdot \prod_i C_i^{n_i} \qquad \textit{Equation 8-1}$$

where

K'	pseudo-equilibrium constant
γ_i	activity coefficient of species i
n_i	stoichiometric coefficient for species i
C_i	concentration of species i

8.2 SPECIATION REACTIONS AND EQUILIBRIUM CONSTANTS

Nagpal and Dahlstrom (1994) considered ionic speciation when interpreting redox measurements. Equilibrium constants were calculated using Gibbs free energy data from Bard *et al.* (1985). Dry (1984) and Crundwell (1988) indicate that in the presence of hydrogen sulphates, ferrous and ferric iron, speciation includes reactions with the sulphate and bisulphate ions. Bryson and Nicol (1996) lists thermodynamic data for several compounds (Appendix G), including equilibrium constants for several reactions. The values of the equilibrium constants used by several authors are contained in Table 8-1. The discrepancies in these values are clear and indicate that accuracy to the same order of magnitude is all that can be expected. The Criss Cobble correlation (Section 7-10) enables one to estimate thermodynamic data for ionic compounds. This alleviates the problem of the inability to directly measure thermodynamic data for these compounds.

8.3 SPECIATION MODELS

8.3.1 EQUILIBRIUM CONSTANTS

Dry (1984) and Crundwell (1988) elucidated the equilibrium or stability constants (at zero ionic strength) of speciation reactions considered (see Table 8-1) as a function of temperature (Equations 8-2 to 8-6). The variation of these equilibrium constants with temperature is presented in Figure 8-1. By its effect on the equilibrium constants, it is clear that temperature affects speciation, particularly the formation of $FeSO_4^+$, $FeHSO_4^{2+}$ and $FeHSO_4^+$.

$K(FeSO_4^+)$: $$\exp\left[-900.421+\frac{39110.926}{T}+136.626\ln(T)\right]$$ (Dry, 1984)- *Equation 8-2*

$K(FeHSO_4^{2+})$: $$\exp\left[33.334-\frac{7629.9}{T}\right]$$ (Dry, 1984)- *Equation 8-3*

$K(FeSO_4)$: $$10^{\left(3.339-\frac{337.37}{T}\right)}$$ (Dry, 1984)- *Equation 8-4*

$K(FeHSO_4^+)$: $$\exp\left[30.69-\frac{7290.4}{T}\right]$$ (Dry, 1984)- *Equation 8-5*

$K(HSO_4^-)$: $$\exp\left[14.0321-\frac{2835.2}{T}\right]$$ (Pitzer, 1977)- *Equation 8-6*

Table 8-1: Equilibrium constants at 298K for several reactions.

Reaction	Bard *et al.*(1976)	Bryson (1996)	Nagpal *et al.*(1994)	Dry (1984)
$H^+ + SO_4^{2-} \leftrightarrow HSO_4^-$	98.81	81.283	98.81	91.652†
$Fe^{3+} + SO_4^{2-} \leftrightarrow FeSO_4^+$	13,539.22*	10,964.78	13,697.92	9,872.79
	18,178.40*			
$Fe^{2+} + SO_4^{2-} \leftrightarrow FeSO_4$		158.49	1.04	161.23
$Fe^{2+} + H^+ + SO_4^{2-} \leftrightarrow FeHSO_4^+$				512.80
$Fe^{2+} + SO_4^{2-} \leftrightarrow FeSO_4$		158.49		161.23
$Fe^{3+} + 2\ SO_4^{2-} \leftrightarrow Fe(SO_4)^-$		239,883.29		
$Fe\ SO_4^+ + SO_4^{2-} \leftrightarrow Fe(SO_4)^-$			19.21	
$Fe^{3+} + H^+ + SO_4^{2-} \leftrightarrow FeHSO_4^{2+}$				2,305.82

* This constant varies as the standard ΔG_f for Fe^{3+} is listed as either -4.7 or -16.8 J/mol (Bard *et al.*, 1985) (Appendix G)

† Taken from Pitzer *et al.*, 1997

8.3.2 EQUILIBRIUM CONCENTRATION SOLUTION

Dry (1984) and Crundwell (1988) implemented the approach of I and Nancollas (1972) to solve for the equilibrium concentration of several reactions (Appendix I). A set of reactions at equilibrium constitutes a set

of simultaneous linear equations. As such, the equilibrium concentration of the components can be achieved by an iterative numerical approach. The solution as dictated by these reactions is often hindered by overshooting during early iterations when the initial estimates required to start the solution differ considerably from the final convergent values.

The technique of l and Nancollas (1972) employs a modified Newton-Raphson method where various matrices are scaled and a convergence-forcer is employed. Appendix I contains a flowsheet representation of this technique as well as the 'Equil' program listing in Borland C. Appendix H describes the flowsheeting symbols used. Dry (1984) and Crundwell (1988) include activity coefficients predicted by the Debye-Hückel model (Section 7.6.1) the specific parameters used are detailed in Appendix D (Table D-1).

8.3.3 DOWNFALLS OF DRY AND CRUNDWELL 'EQUIL' PROGRAM

Ideally a program which can be used and adapted by several users is required. In this respect the 'EQUIL' program has several downfalls, owing to it being written for a particular system at a particular temperature. To consider a higher temperature or investigate the effect of a change in temperature, several other factors need to be taken into account. These are discussed in turn.

'hard-coded': Information for the reactions, *e.g.* equilibrium constants, activity coefficient parameters and other details are 'hard-coded' into the program. In order to change the information or add another reaction, the original source code needs to be altered. Because the calculations are based on an array structure, the code for the calculations needs to be amended. As a result the flexibility of the program is significantly reduced, thereby limiting its application.

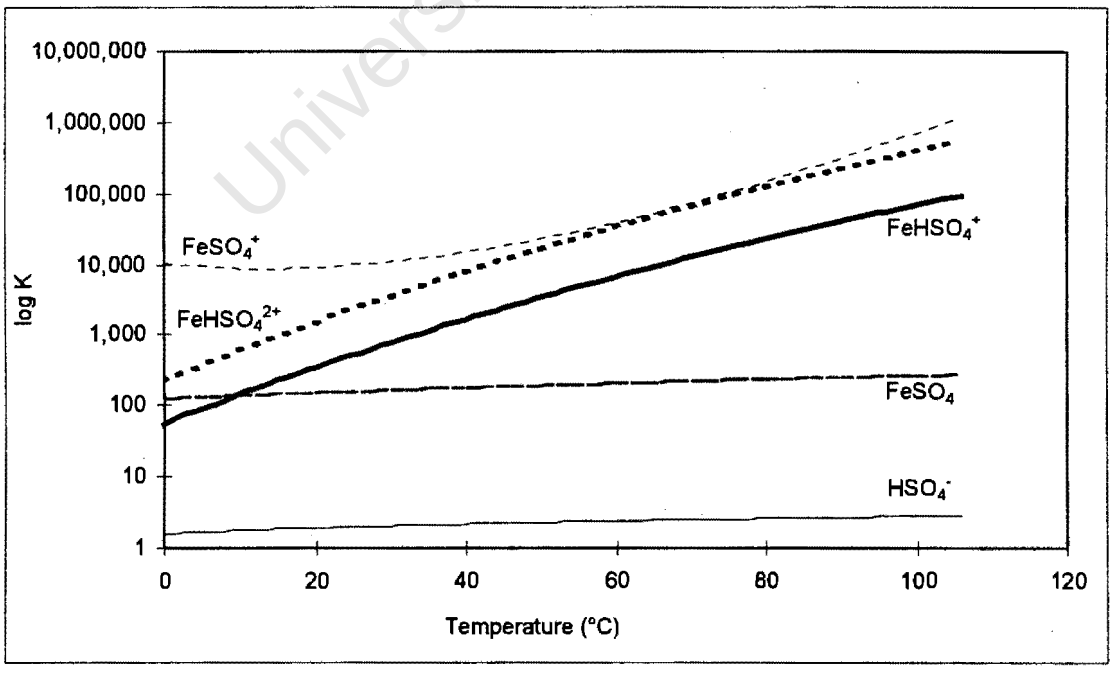


Figure 8-1: Variation of equilibrium constants with temperature (Dry, 1984).

$Fe^{2+} \leftrightarrow Fe^{3+} + é$: As temperature changes, so the equilibrium between ferrous and ferric iron changes. 'Equil' assumes a constant Fe^{2+} and Fe^{3+} concentration as opposed to a constant total iron concentration. Hence no migration between these ionic species is considered. This eliminates the possibility that, as temperature changes, ferrous iron is oxidised to ferric iron which impinges upon the distribution of ferric and ferrous iron species.

Formation of Precipitates: The bioleaching of mineral sulphides involves the gradual migration of ions from the solid state (within the mineral sulphide) to the aqueous state and possibly back to the solid phase if the ions are precipitated out in one form or another. Another form in which ferric or ferrous iron can be eliminated from the solution system is by the formation of precipitates. This essentially removes components from solution, hence the total iron concentration in solution does not remain constant. This will affect the equilibrium between ferric and ferrous iron as well as the distribution of ferric and ferrous iron complexes.

Pitzer Bisulphate Correlation: This correlation (Pitzer *et al.*, 1977) is used to predict the activity coefficient of the bisulphate ion, as detailed in Appendix E. The equation employed by 'Equil' for the Debye-Hückel limiting law coefficient (A_ϕ) is valid only between 0-55°C. The information required for higher temperatures is available (Zemaitis *et al.*, 1986).

Activity Coefficient Model: Zemaitis *et al.* (1986) state that more rigorous activity coefficient models are being developed. A combination of this and increased computing power enables more accurate activity coefficients to be predicted. A question which remains is whether the parameters required by these new models will be available for the range of ionic components present in the bioleaching environment. If one is satisfied with a first approximation then there is validity in sticking with the Debye-Hückel model.

I and Nancollas (1972) Equilibrium Solution Technique: A downfall of this approach is that compounds are defined as either components (*e.g.* the reactants Fe^{2+}, Fe^{3+}, H^+ and SO_4^{2-}) or derived species (*e.g.* products such as $FeSO_4^+$, $FeSO_4$, $FeHSO_4^{2+}$ $FeHSO_4^+$ and HSO_4^-). A substance cannot be considered as both a component and a derived species, hence the bisulphate ion formation reaction is noted in the reverse reaction. While this is possible for several reactants and products (or components and derived species), it implies that the addition of a reaction may require the 'flipping' of other reactions – hence a need to alter existing data.

Using this simulation (Appendix I) the effect of temperature on the ratio of free ferric to free ferrous iron was calculated, Figure 8-2. Despite the exclusion of the formation of precipitates, there is a significant decrease in this ratio. Redox potential is directly dependent on this ratio (Section 7.7), which in turn effects the behaviour of ore and leaching kinetics in bioleaching (Breed *et al.*, 1997; Nagpal and Dahlstrom, 1994; Lacey and Lawson, 1970). Hence, an increase in temperature has important repercussions on the speciation of iron in the bioleaching system and speciation needs. Further, the effect of temperature on ionic speciation needs to be investigated.

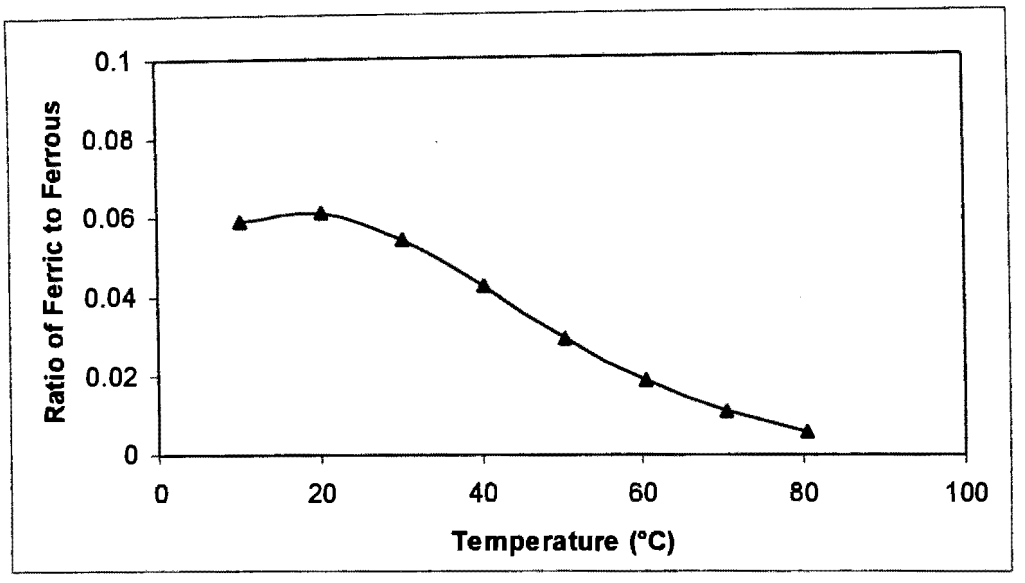


Figure 8-2: Variation in the ratio of ferric to ferrous iron with temperature, estimated using the 'Equil' simulation (Crundwell, 1988).

8.4 SPECIATION SIMULATION PACKAGES

Various speciation simulation packages are available including Minteqa2, JESS and Aspen Plus. These and other speciation packages often engender problems in the following areas:

- Technical obstacles are found in preparing input data owing to the availability of data. This varies with the complexity of the computational approach, *e.g.* the activity coefficient model.
- Chemical expertise is required particularly with the definition of components, the methods of expressing equilibria, the effects of background electrolytes and the choice of standard state. Alternately, the application attempts to remove the expertise from the user to the package. As a result the user may not correctly interpret the results from the application.
- Published thermodynamic data for various substances varies, *e.g.* the Gibbs free energy of formation for Fe^{3+} is recorded as –4.6 and –16.7 kJ/mol in Bard *et al.* (1985). A problem arises in choosing which values to use. Zemaitis *et al.* (1986) recommend using data from the same collection in order to minimise resulting inaccuracies.
- Existing thermodynamic systems have a set way of formulating and describing chemical reactions. This may or not model the natural chemical approach closely. It is more likely to be computationally convenient and not evident from the user interface. This complicates interfacing the thermodynamic module with other calculation sets, *e.g.* kinetic or microbial growth modules.
- Several packages, depending on their commercial application are not user-friendly or flexible.

8.4.1 MINTEQA2

MINTEQA2 is a geochemical equilibrium speciation programme developed by the United States Environmental Protection Agency. The whole package consists of two modules, MINTEQA2 and PRODEFA2. PRODEFA2 is designed to be executed first to create a MINTEQA2 input file. When it comes to high temperature

calculations, MINTEQA2 assumes a constant heat capacity and uses the enthalpy to predict changes in Gibbs free energy (Buckley, 1993). As the heat capacity does vary with temperature, it would be better to calculate the change in Gibbs free energy including variations in the heat capacity.

MINTEQA2 is mainly geared towards dilute aqueous systems. It makes use of either the Debye-Hückel or the Davies activity coefficient models. As MINTEQA2 generates set output files, it can be interfaced with other modules. However if one wishes to alter the system of reactions, then the MINTEQA2 input file will need to be altered. Ideally one requires a built-in speciation module.

8.4.2 JOINT EXPERT SPECIATION SYSTEM (JESS)

This package was designed to minimise the expertise needed to carry out various calculations concerned with chemical species in solution. One of the aims was to embody within the application comprehensive knowledge of solution chemistry (May and Murray, 1991a). This is accomplished by obtaining information about every species which may be peripheral to the chemical issue, but is required by the program to make decisions which will influence the program's chemical predictions (May and Murray, 1991a).

This large on-line database requires several properties for each species. These properties are often not readily available for those present in the minerals processing industry. Expertise rests both with the program and the user. This is an advantage if the user understands the computation. However, further advantage is obtained by implementing a stage-wise calculation to see the effect of various parameters or modules (May and Murray, 1991b).

Expansion of the package, or incorporation of the speciation model with other modules is possible. Once the system is defined, JESS builds a Fortran program, which can be interfaced with other modules. However on altering the speciation system, then the JESS program needs to be recreated.

A major advantage of JESS is that, because of its programming language (Fortran 77), it is relatively hardware and system independent. However, this is a downfall in terms of database modules as Fortran does not have an indexed file capability (May and Murray, 1991a). If one used a conventional, standardised database language for handling the data, then one would be able to augment the database functionality more readily.

8.4.3 ASPEN PLUS

Aspen is a commercial process simulation package. It has been said that Aspen is difficult to use as a package, owing to difficulty in understanding the procedures used by the program and the models implemented. Aspen provides the user with the option of doing more rigorous calculations. However, this is restricted by the availability of data for the substances being considered.

Extracting data from the Aspen database is not straightforward, if not impossible. Hence, one does not know what numbers are being used for the calculations. Considering the dispute surrounding several compounds, there is very little means for comparing the values received with those put forward by other systems or authors.

8.4.4 DISCUSSION

Both JESS and Aspen are very rigorous systems and are likely to produce accurate results in terms of current knowledge of chemical speciation. Interfacing other modules and sometimes understanding the chemical manipulations can be difficult. These systems require in depth knowledge of the chemical system proposed. Bryson (1997) believes that there are so many errors in assumptions made about non-dilute electrolytic solutions that one is better off taking a simplistic approach.

One is not able to ammend JESS to include specific kinetic information about the leaching process or mass transfer characteristics specific to a particular process. Where a speciation application unit is generated by the package, as with MINTEQA2 and JESS, alteration of the reaction system requires regeneration of the speciation unit. Ultimately a useful research tool would encapsulate the speciation module within the same framework as other process-related modules.

8.5 DISCUSSION AND FUTURE OBJECTIVES

Why propose another speciation model when several good ones are available? Modelling of bioleaching kinetics is often in terms of a ferric or ferrous iron species (Lacey and Lawson, 1972; Breed *et al.*, 1997; Nagpal and Dahlstrom, 1994). The benefit of a modularised ferric - ferrous based speciation application is its amenability to implementation in the bioleaching field. An advantage of this modularised approach is the ready implementation of improved data measurement or estimation. Further, as more areas of bioleaching are coded in a similar modular manner, then a simulation of bioleaching can be developed.

8.6 NOMENCLATURE

symbol	*description*	*units*
K'	pseudo-equilibrium constant	units of concentration
γ_i	activity coefficient of species i	
n_i	stoichiometric coefficient for species i	
T	Temperature	K
C_i	concentration of species i	mol/l

8.7 REFERENCES

Bard, A.J., R. Parsons and J. Jordan (1985): "Standard Potentials in Aqueous Solution", Marcel Dekker, New York

Barret, J., M.N. Hughes, G.I. Karavaiko and P.A. Spencer (1993): "Metal extraction by bacterial oxidation of minerals", Ellis Horwood, England, p73-101

Breed, A.W., S.T.L. Harrison and G.S. Hansford (1997): "Technical note: A preliminary investigation of the ferric leaching of a pyrite/arsenopyrite flotation concentrate", *Minerals Engineering*, in press

Bryson, A. (1997): Personal Communication, Chemical Engineering Department, University of Witwatersrand, South Africa

Bryson, A. and M. Nicol (1996): "Thermodynamics of Aqueous Solutions", *Continuing Engineering Education Course*, University of Witwatersrand, South Africa

Buckley, C. (1993): Course Notes, *Introduction to MINTEQA2*, Pollution Research Group, University of Natal Durban, South Africa

Crundwell, F. (1991): " The role of charge-transfer mechanisms in the oxidative and non-oxidative dissolutions of sphalerite", PhD thesis, University of Witwatersrand, Johannesburg

Dry, M.J. and A.W. Bryson (1988): "Prediction of Redox Potential in Concentrated Iron Sulphate Solutions", *Hydrometallurgy* **21** 59-72

Dry, M.J. (1984): "Kinetics of leaching of a low grade matte in ferric sulphate solution", PhD thesis, University of Witwatersrand, Johannesburg

I, T and G.H. Nancollas (1972): "EQUIL – A general computational method for the calculation of solution equilibria", *Analytical Chemistry*, **44**(12), 1940-1950

Lacey, D.T. and F. Lawson (1970): "Kinetics of the liquid-phase oxidation of acid ferrous sulfate by the bacterium *Thiobacillus ferrooxidans*", *Biotech. Bioeng.*, **12**, 29-50

May, P.M. and K. Murray (1991a): "JESS, a joint expert speciation system – I. Raison d'être", *Talanta*, **38**(12), 1409-1417

May, P.M. and K. Murray (1991b): "JESS, a joint expert speciation system – II. The thermodynamic database", *Talanta*, **38**(12), 1419-1426

Nagpal, S. and D. Dahlstrom (1994): "A mathematical model for the bacterial oxidation of a sulfide ore concentrate.", *Biotech.Bioeng.* **43** 357-364

Smith, J.R., R.G. Luthy and A.C. Middleton (1988): "Microbial ferrous iron oxidation in acidic solution", *Journal WPCF*, **60**(4), 518-530

Zemaitis, J.F. Jr; D.M. Clark, M. Rafal and N. C Scrivner (1986): "Handbook of Aqueous Electrolyte Thermodynamics", American Institute of Chemical Engineers, New York, p 47-66

9. Thermodynamic Data Model Implementation

9.1 OBJECTIVES AND INTENTIONS

The need for an adaptable, modular simulation of bioleaching processes has been outlined. Such a simulation must include the prevailing chemistry and thermodynamics of the system as a function of solution chemistry, temperature, *etc.* Owing to limitations of other simulation packages (Section 8.4) and the need for flexibility, development of a thermodynamic data model as a module of this simulation is addressed. This will be readily understood and adjustable by the end-user, as summarised in terms of the following objectives.

9.1.1 DATA MODEL

The thermodynamic data needs to be modelled such that the identity and relationships of the data (Section 7) is retained. This must not be at the expense of application flexibility. The end-user must be capable of extending the scope of the application. Hence, the modelling of the thermodynamic data plays an important role in the development of the application.

As mentioned in Section 8.4, most simulation packages have a set means of formulating and expressing chemical reactions. A common trend is to define some compounds as components (or reactants) and other compounds as derived species (or products) (May and Murray, 1991; I and Nancollas, 1972). This introduces system specific categorisations since a compound may act as both a product and a reactant in a system of reactions. Hence compounds must be handled consistently irrespective of whether they are reactants or products to provide unique interpretation and referencing of variables *e.g.* components, including the phase or state of the component.

9.1.2 FLEXIBILITY

To ensure that the simulation can be applied to a variety of mineral systems, it is necessary to be able to add reactions to the system effortlessly. As a result, the implementation of the data model in calculations must not be 'hard coded'. Hard coding eliminates the need to teach the application what the data represents, and therefore an inherent method of how to handle the data. Further, consistent fundamental operations are required as no allowance can be made for exceptional compounds. Ultimately the simulation needs to be able to handle a finite group of specific categories of data. The corollary of this is that the data needs to be appropriately classified. This in turn requires a good input structure, which is easy to follow. Wherever possible, low level error checking must be included, *e.g.* non-negative concentrations or activity coefficients. This will test for accidental errors, not errors in the actual values of the data. Hence, the expertise remains with the end user, unlike with JESS (Section 8.4.2; May and Murray, 1991). As an aside, it is essential that any functionality be implemented effortlessly. This requires that the theme of modularisation of data encapsulation must be strictly implemented. Hence this 'expert' functionality can be introduced at any stage.

9.1.3 DATA INTEGRITY AND REFERENCING EFFICIENCY

In keeping with the previous aspects and objectives of the data model, duplication of important information must be minimised. Any duplication must be confined within the different databases, *i.e.* no data duplication should occur within a database apart from referencing purposes. This ensures that if thermodynamic data for a particular compound is amended, there will be no copies of any previous values. The way in which tables reference each other represents the relationships between data. Hence the formulation and tracking of these referencing variables is crucial.

Error checking and handling, especially when retrieving data from the database is important. Before data is manipulated, checks must be performed to ensure that all the required data exists. The database should be constructed such that low-level error checking occurs whenever data is written to a table. One of the most important aspects of database design is understanding the safeguards which need to be implemented to protect data integrity.

9.1.4 USER FRIENDLINESS

As computer software technology migrates towards a graphical (windows) environment, end-users are becoming less tolerant of text-based systems. In order to maximise the user-friendliness of the application and ease of the future modifications, chemical reactions need to be recorded in a graphical (Windows) environment.

9.2 DATABASE DESIGN

A relational, normalised database is able to meet these objectives. A database is a collection of organised, inter-related data stored together without unnecessary redundancy. Data have an inherent structure. This structure needs to be modelled in order for the database to be coherent and flexible for further additions. This requires that the data be well indexed with unique keys (primary key). Incorporation of these principles ensures that the referential integrity of the data is not diminished through duplication. (Computer Associates, 1994).

9.2.1 DATABASE CONCEPTS

9.2.1.i Conceptual database design

Before building or implementing a database, the requirements of the database must be defined. This entails defining the objects to be modelled by the database, their characteristics, and their relationships. In the database design, the tables belonging in the database, columns belonging in each table, and the relationship between the tables is defined. A relational database affords flexibility because the logical structure of the database is independent of its physical storage and structure. Two concepts, relationship modelling and normalisation, are fundamental to designing a database (Borland, 1996b).

9.2.1.ii *Relationship modelling*

Relationship modelling includes:

- Identifying the major groups of information to store in the database
- Analysing the type of information and its properties
- Identifying relationships among sets of information

In a relational database, the data is organised into tables. Each row of a table contains information about a particular item or record. Each column contains one piece of the information that makes up a record, termed a field (Borland, 1996a). For example, think of the groups of information as tables, with each table describing one type of object, such as compounds or Criss Cobble constants. The type of information and its properties are columns in the table, for example, describing the compound formula and molecular weight or the regression constants to calculate Criss Cobble constants.

Relationships are modelled through linking tables with comparable fields, known as keys or indexes. The data in a relational database is the information stored in all the related tables. The advantage of a relational database is that data can be extracted or combined from several tables to get exactly the information you need, without changing the structure of the database. Moreover, a few small and discrete tables are more convenient to use and maintain than one large table (Borland, 1996a).

9.2.1.iii *Keying*

Indexes and primary keys govern the relationships between tables and therefore subsets of data, *i.e.* they ensure that the normalised tables are linked in an appropriate manner. A table can have many indexes defined, but one must be identified as the primary index, known as the primary key. The key is a field (or group of fields) that contains data that uniquely identifies each record of a table. A key requires a unique value for each record (row) of a table, ensuring no duplicate records in the table (Borland, 1996b). For example when describing the stoichiometry of reactions, the unique index is comprised of the reaction number and the component identification. Thus the unique key will ensure no duplicate entries for any component for a given reaction.

Relational theory requires that any item of information in a database be uniquely identified by three pieces of information:

1. the table in which that item is stored
2. the field in that table which is being accessed
3. the value of the primary key for that record

These criteria ensure that all information in the table can be appropriately and efficiently referenced (Ehrmann, 1995).

- Running Database: running information, *i.e.* calculation input and output (Appendix L)

The integrity of the actual thermodynamic data is ensured as it is kept apart from the working variables and can only be altered by certain procedures. Figure 9-1 details the table structure of the Species database. Note that the database is both relational and normalised. The structure of the running database is less crucial at this stage as it serves as temporary storage for calculation input and output.

The application using the databases was coded with object oriented Pascal, using the Delphi® 2.0 compiler. Connections to the databases are controlled by the Interbase™ server associated with the Delphi® Developer's pack, using the SQL command set.

9.3.1 SPECIES DATABASE TABLES

In keeping with the rules of normalisation (Appendix J), the Species database is discretised into several small tables. Tables storing thermodynamic and compound-specific data are described in Table 9-1. The tables of this database are depicted in Figure 9-1, represented by boxes. Reference keys (denoted by arrows) for the compound number, reaction number, reference code, Criss Cobble code and state code are used to demonstrate actual relationships between the data.

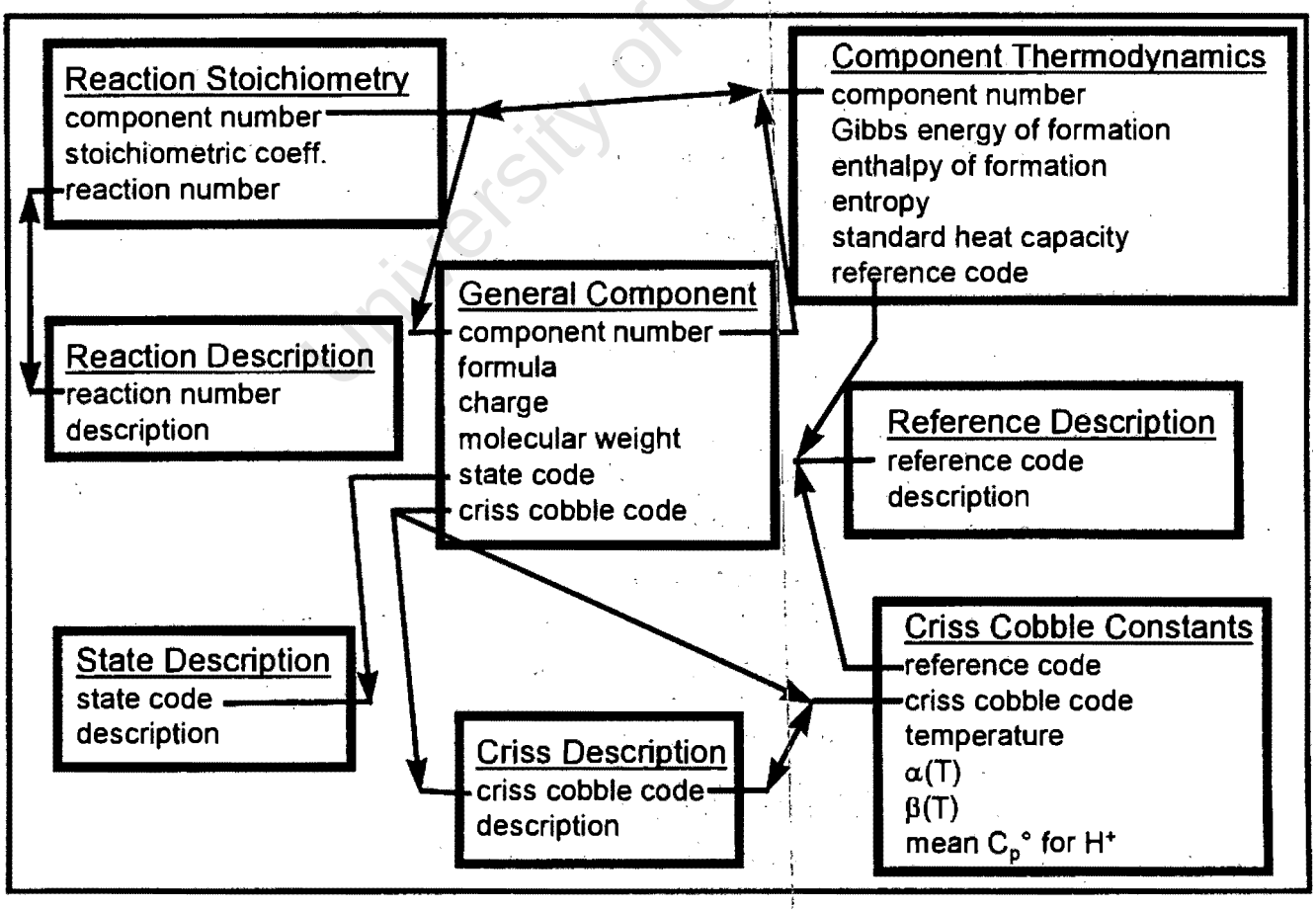


Figure 9-1: Schematic representation of the Species database

Table 9-1: Description of several Species database tables, sample data given.

Variable	*Description*	*Example*
CompGen Table - Contains all general component information		
CompNo	Component number, which is automatically assigned when a new record is entered. As a result the user does not need to keep track of component numbers.	1004
CForm	Unique chemical formula where a conventional and consistent notation is assumed, where appropriate, this should include information about the phase of the component.	SO4(-2)
CName	The full name of the component.	sulphate ion
Charge	Component charge	-2
MWeight	Component molecular weight (g/mol)	96.06
IonSize	Debye-Hückel ionic size parameter (Å)	4
St_Code	State code to describe the state (*e.g.* aqueous or liquid) of the component.	aqI
Criss_Code	A Criss Cobble correlation code (*e.g.* cations, simple anions or oxy-anions)	oxy-anion
Ref_Code	A unique reference code indicating the source of the data.	Dry1984
CompThermo Table – Thermodynamic data of the components		
CompNo	Component number instantiating the relationship between the general and thermodynamic data of the component.	1004
Gibbs_F	Gibbs free energy of formation (kJ/mol)	-744.4
Enthalpy_F	Enthalpy of formation (kJ/mol)	-909.9
Entropy	Entropy (J/mol·K)	20.1
HeatCap	Heat capacity at standard state (J/mol·K)	-293.1
Ref_Code	A unique reference code indicating the source of the data.	Bryson1996
Cp_Data Table – Heat capacity correlation coefficients for non-ionic chemicals		
The data is stored according to $C_P(T) = a + bT + \frac{c}{T^2} + dT^2 + eT^3$		*Equation 9-1*
where T is the temperature in Kelvin.		
CompNo	Component number instantiating the relationship between the component and the heat capacity information[†].	
Coeff_A	a	
Coeff_B	b	
Coeff_C	c	
Coeff_D	d	
Coeff_E	e	
Criss_Cobble Table – Criss Cobble correlation coefficients		
Criss_Code	Code to ensure that appropriate coefficients are used for each ionic, aqueous component.	oxy-anion
ATC	a(T) y-intercept (Table F-2)	479.3032
ATM	a(T) slope	-1.6084
BTC	b(T) y-intercept	-0.7284
BTM	b(T) slope	0.0058
HSATC	H^+ absolute entropy y-intercept[‡]	
HSATM	H^+ absolute entropy slope	

[†] Note there will be no entry for sulphate in the 'Cp_Data' table as it is an ionic species.

[‡] Note the HSATC and HSATM fields are only populated in the case of H^+.

9.4 CALCULATION CLASS DESIGN

Owing to the nature of equilibrium calculations and the direct relationship between them and reaction stoichiometry, careful planning needs to be implemented to ensure consistent and generic stoichiometry and variable referencing. Further it is necessary to ensure that consistent calculations can be maintained for set categories of data, enabling future functionality adaptation. This is achieved by taking an object-oriented approach.

Object-oriented modelling and design is a new way of thinking about problems, using models organised around real-world concepts. The fundamental construct is the object, which combines both data structure and behaviour in a single entity (Appendix M). Object-oriented models are useful for understanding problems, design programs and databases (van der Merwe, 1995; Rumbaugh *et al.*,1991).

The object model developed for these thermodynamic calculations is depicted in Figure 9-3. The boxes represent objects and the properties and methods of the objects are listed in two boxes below the object name. The object heirarchy within the Delphi 2.0® TObject framework is represented by arrows. Further details of the program are contained in Appendix N.

Figure 9-2 illustrates the form used when entering, editing or viewing compounds.

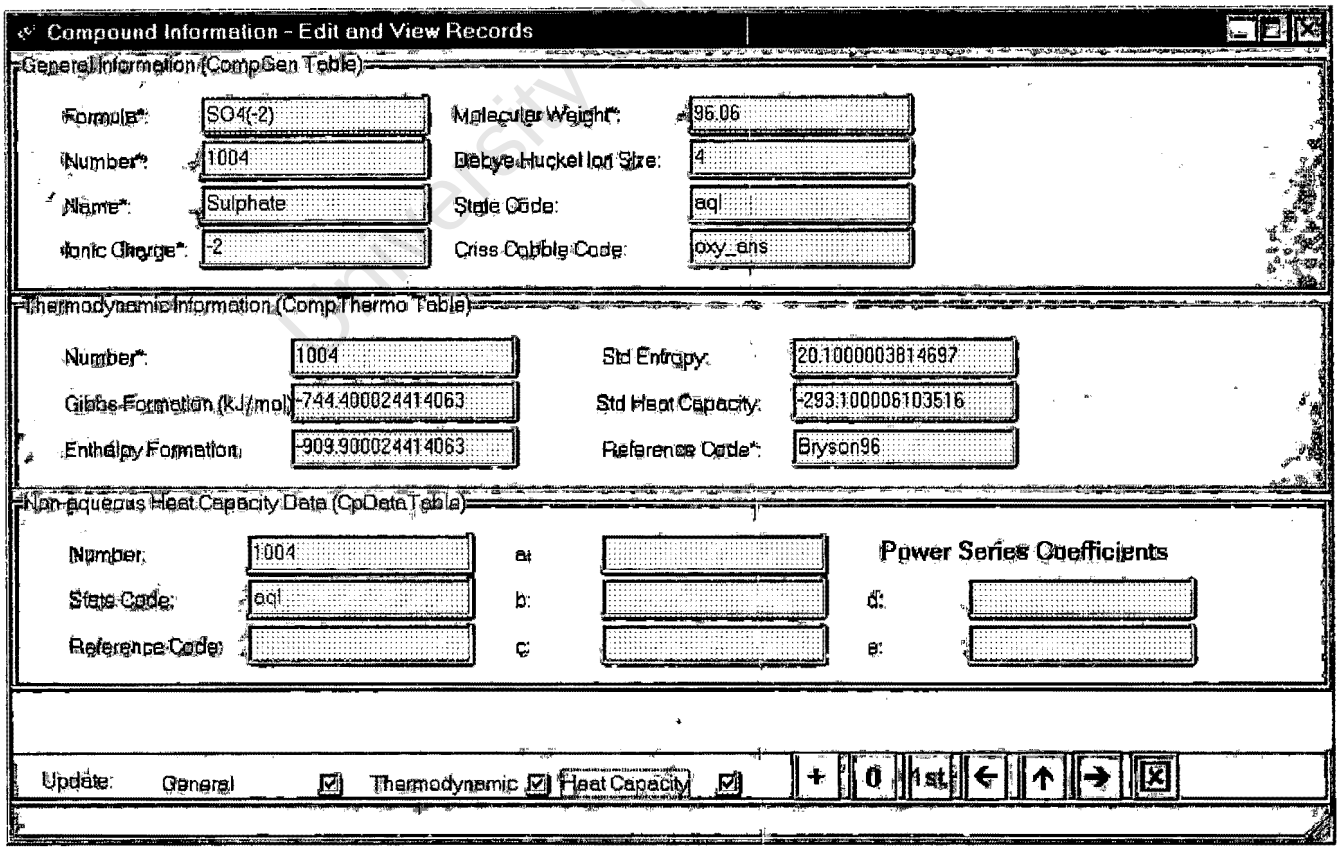


Figure 9-2: Screen shot depicting the compound data form.

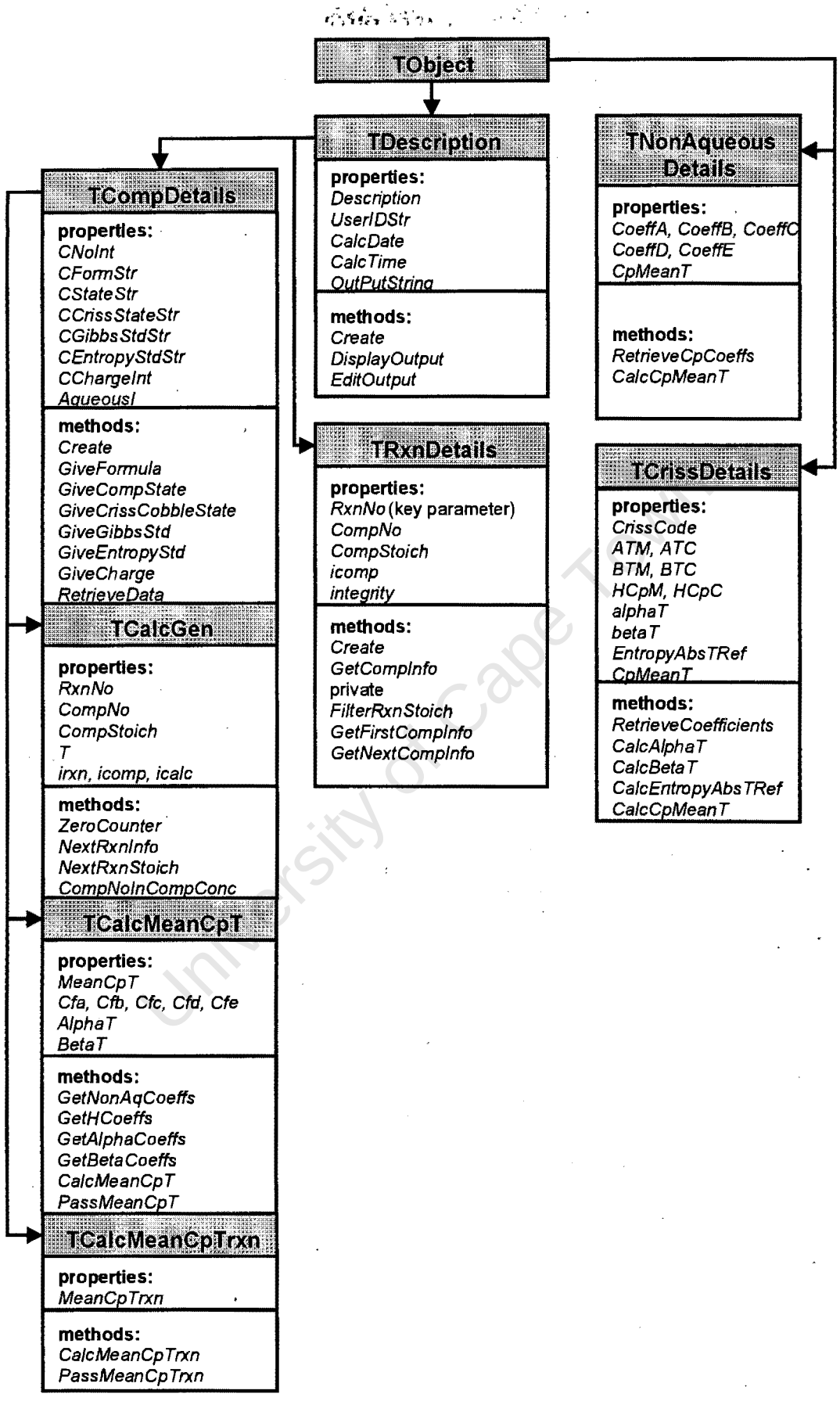


Figure 9-3: Schematic representation of the object oriented class design

9.5 SAMPLE OUTPUT

The simulation is able to calculate Gibbs free energy for compounds, including ionic species, and hence equilibrium constants for any given reactions. It generates output in the form of a text file, contained in the box below, and a graph (Figure 9-4).

```
User name:                    Karen
Date of calculation:          97/08/23
Time of calculation:          06:26:06 PM
Reaction number:              4
Reaction description:         H(+1) + SO4(-2) = HSO4(-1)
Std Entropy:                  111.8
Std Gibbs free energy :       -12.09998
Equilibrium const. (Std):     131.8081
log K (above):                2.1199
************************************************************
Calculation Output for reaction 4.

     T     |   Cp Rxn    |    dG Rxn   |        K Rxn  |  log K  |
     K     |   J/mol·K   |    kJ/mol   |               |         |
-------------------------------------------------------------------
   283.15  |    269.720  |   -10.541   |           88  |    1.945
   291.15  |    273.550  |   -11.356   |          109  |    2.037
   299.15  |    277.330  |   -12.229   |          137  |    2.135
   307.15  |    281.070  |   -13.162   |          173  |    2.238
   315.15  |    284.760  |   -14.155   |          222  |    2.346
   323.15  |    288.410  |   -15.210   |          287  |    2.459
   331.15  |    292.020  |   -16.326   |          376  |    2.575
   339.15  |    295.590  |   -17.504   |          497  |    2.696
   347.15  |    299.120  |   -18.746   |          662  |    2.821
************************************************************
```

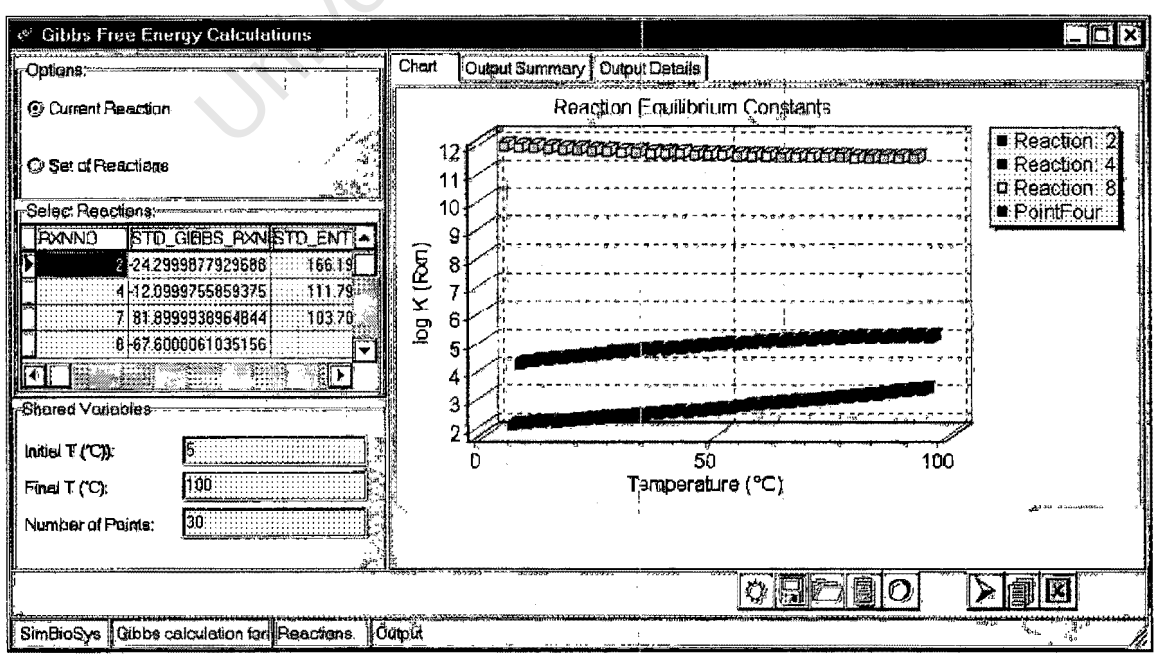


Figure 9-4: Screenshot depicting graphical output of equilibrium constants.

9.6 DISCUSSION

The infrastructure of the data model and object oriented approach ensure that any reaction and compound can be entered by the user for consideration. Further, both reactants and products are entered in the same database. At this stage, the data required by the simulation is limited to the essentials, without reducing the accuracy of the calculations. Future work on the simulation includes extending the calculation to solve for the equilibrium concentrations of the compounds present. Activity coefficients will need to be incorporated to take non-ideality into account. If the Debye-Hückel activity coefficient model is used, then the data model will not need to be extended to include these calculations.

9.7 REFERENCES

Borland (1996a): "Delphi 2.0 Manual, Database Desktop User's Guide", Borland International

Borland (1996b): "Delphi 2.0 SQL tutorial", Borland International

Computer Associates (1994): "CA-INGRES, Designing Ingres Databases (Student Guide release 6.4), Computer Associates International, Inc.

Ehrmann, D. (1995): "Paradox Queries - A Complete Reference", M&T Books, New York

I, T. and G.H. Nancollas (1972): "EQUIL - A General Computational Method for the Calculation of Solution Equilibria", Analytical Chemistry, 44 (12) 1940-1950

May, P.M. and K. Murray (1991): "JESS, A Joint Expert Speciation System - II The Thermodynamic Database", Talanta, 38 (12) 1419-1426

Pacheco, X and S. Teixera (1995): "Delphi™ Developer's Guide", p415-439, Sams Publishing, United States of America

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991): "Object-oriented modelling and design", Prentice Hall, United States of America

van der Merwe, J.J.N (1994): "Introduction to object-oriented programming", Course, Department of Electrical Engineering, University of Stellenbosch, South Africa

10 Conclusions and Recommendations

Bioleaching is an accepted and well-implemented process for the pretreatment of mineral sulphides. Alternative processes include pressure leaching and roasting. Of these three, bioleaching is an attractive option in terms of its lower impact on the environment. A drawback of bioleaching is its slow kinetics leading to long residence times and large stirred tank reactors. Chemical oxidation by ferric iron contributes to the overall rate of bioleaching in accordance with the two-step and indirect bioleaching mechanisms. The rate of chemical reactions generally increases with temperature. Hence, there is growing interest in increasing the temperature of bioleaching operations. Preliminary investigation has shown that at 65°C heat generation due to exothermic reaction balances the energy input required to maintain a constant temperature. Thus, overall increased leaching rates and reduced cooling costs make thermophilic leaching economically attractive.

Thermophilic bioleaching is a feasible process if the following criteria are demonstrated:

- oxidation of ferrous iron and /or reduced sulphur
- fast oxidation kinetics
- operation at high pulp densities
- tolerance to low pH and high metal concentration
- efficient gold or valuable metal recovery

The operating conditions of bioprocesses are determined by the tolerances of the microorganisms employed. Thus, it is necessary to utilise acidothermophiles similar to *Thiobacillus ferrooxidans* or *Leptospirilum ferrooxidans*, capable of oxidising reduced sulphur compounds and ferrous iron. Members of the *Acidianus*, *Sulfolobus* and *Metallosphaera* genera are able to withstand pH values below 2, temperatures ranging from 65 to 80°C and are capable of utilising reduced sulphur compounds or ferrous iron as an energy source. From the metabolic point of view they are good candidates for thermophilic bioleaching. These extreme thermophiles are widely distributed in a variety of hot acid habitats around the world, both terrestrial and aquatic, in environments from 55-92°C.

It is necessary to assess the effect of temperature on mass transfer and ionic speciation as these two phenomena affect the efficiency of the bioleaching process. Mass transfer is important, as extreme thermophiles require oxygen and carbon dioxide for respiration. The contribution from chemical leaching implies that the transfer of soluble iron is also important. Further, as the concentration of ferric iron and redox potential are important parameters, ionic speciation is also important.

Three major factors influencing mass transfer are the mass transfer coefficient (k_L), interfacial area (a) and the driving (C-C^*). By considering the effect of temperature on diffusivity, viscosity, density and surface tension, it was established that, excluding hydrodynamics, the mass transfer coefficient is the parameter most influenced by temperature. The mass transfer coefficient varies with temperature according to the liquid phase diffusivity to

the power of $^1/_2$ or $^2/_3$ for mobile and rigid bubbles respectively. The mobility of a bubble is dependent upon the surface regeneration and is affected by fluid velocities and surfactants or electrolytes present. Rigid bubbles do not coalesce, resulting in a smaller bubble size on average. Mobile bubbles form a coalescing system and a larger bubble size on average. These correlations do not enable the absolute values of transfer coefficients, in rigid and mobile bubble systems, to be contrasted. It merely indicates that the effect of temperature on the mass transfer coefficient in a rigid bubble is more significant.

Interfacial area is proportional to density to the power of 0.2 and inversely proportional to surface tension to the power of 0.6. This does not include the effect of hydrodynamic conditions. Of interest is the change in mass transfer coefficient on transition from a coalescing to a non-coalescing system. Both k_L and a are affected by this transition, but the rate of increase in the interfacial area is higher than the rate of decreases in the mass transfer coefficient during this transition resulting in improved mass transfer.

As the solubility of oxygen and carbon dioxide decreases with increasing temperature, the driving force decreases. As humidified air is used in thermophilic bioleaching, an important consideration is the inclusion of the effect of increasing vapour pressure. The negative effect of temperature on gas solubility is compensated for by the increase in volumetric mass transfer coefficient with temperature.

Various ionic speciation packages exist, including JESS, Minteqa2 and Aspen Plus. These packages all have their advantages and disadvantages. However, if one wishes to simulate and optimise the bioleaching system as a whole, then it is necessary to incorporate modules of the various contributing factors, including ionic speciation, oxidation kinetics, microbial growth kinetics and mass transfer. The benefit of this simulation is the ability to consider phenomena such as oxidation kinetics and ionic speciation in combination. As the first step towards this modular simulation a unique and consistent data model of chemical compounds was developed.

This data model is flexible and can be utilised by thermophilic calculations and future modules alike. Additional chemicals and reactions can be added effortlessly in a graphical, user-friendly environment. Chemicals are treated alike irrespective of whether they are reactants or products in the given reactions. The data model was implemented with a Delphi® 2.0 simulation with two relational normalised databases. To ensure the integrity of the data, the stoichiometric and thermodynamic data ('Species' database) is kept apart from the working calculations, which are stored in the 'Running' database. The simulation calculates the equilibrium constant for any given reaction, including ionic reactions. Heat capacity data for ionic species is predicted using the Criss Cobble correlation.

10.1 FUTURE WORK

Further work needs to be conducted to investigate the effect of temperature on mass transfer at various hydrodynamic conditions, both from an experimental and theoretical stand point. An understanding of the variation of bubble coalescence or surface mobility with temperature would be beneficial. The effect of eddy diffusivity and electrolytes on diffusivity needs to be incorporated into the liquid phase diffusivity term.

The simulation needs to be extended to calculate the equilibrium concentrations of the components selected. This will enable the quantification of ionic speciation at any temperature, including iron precipitation and changes in the total ferrous and ferric iron concentrations. The effect of this speciation on free ferric iron and redox potential needs to be investigated. As mentioned previously, the simulation must be extended to incorporate other modules, as they become available.

Appendices

A. *Sulfolobus* spp. Growth Environments

The following tables demonstrate the mineral nutrients and growth conditions used by various researchers when growing *Sulfolobus* species. Details of the references are contained in Chapter 3.

	Clark and Norris, 1996	Torres *et al.*, 1995	Escobar *et al.*, 1993	Vitaya and Toda, 1994	Vitaya and Toda, 1994	Furuya *et al.*, 1977
KCl	0.1				0.2	
$LiSO_4 \cdot H_2O$						0.005
NaCl				0.2		
$MgCl_2 \cdot 2H_2O$						
KH_2PO_4				0.3	0.2	0.28
K_2HPO_4	0.2	0.2				
$K_2HPO_4 \cdot 3H_2O$			0.2			
$(NH_4)_2SO_4$	0.4	0.4	0.4	1.3	0.4	1.3
$MgSO_4$		0.5				
$MgSO_4 \cdot 7H_2O$	0.5		0.5	0.25	0.4	0.09
Na_2MoO_4					3×10^{-4}	
$Na_2MoO_4 \cdot 2H_2O$						0.3
$Na_2SiO_3 \cdot 9H_2O$						0.52
$Na_2B_4O_7 \cdot 10H_2O$						0.005
$CaCl_2 \cdot 2H_2O$				0.05		
$CaSO_4 \cdot 2H_2O$						0.23
$Ca(NO_3)_2$						
$FeSO_4 \cdot 7H_2O$						0.023
$Fe_2(SO_4)_3 \cdot xH_2O$						0.05
$Al_2(SO_4)_3 \cdot 16\sim18H_2O$						0.03
$MnSO_4 \cdot 4\sim6H_2O$						0.008
trace elements						
yeast extract				1	0.2	1
casamino acids				1		
glucose				1		
pyrite/mineral sulphide	2%	1,3,5%	0.5-5%(w/v)	1,2,4%	1,2,4%	
ferrous iron			3000 ppm			
thiosulfate						
sulphur						
pH	2	2	1.9	2	2	2-3
temperature (°C)	68°C	70	70		70	70
water				distilled	distilled	distilled
air	400ml/min 1% (v/v) CO_2	1% CO_2 enriched air	3l/min	5% CO_2, 200ml/min	5% CO_2, 200 ml/min	
reactor		500 ml stirred, 400 ml air lift, 250ml flasks	flasks and stirred reactors (500 rpm)		500ml vessel, *ca* 300 rpm magnetic stirrer	1l (65% working volume) shake flask

	Le Roux and Wakerley, 1987	Karavaiko *et al.*, 1993	De Rosa *et al.*, 1975	Zillig *et al.*, 1980
sulphur				10
pH			1.4-2.6	2
temperature (°C)			75	
water			tap	
air	1% CO_2, 600ml/min			5% CO_2
reactor	baffled 700ml vessel, single bladed paddle at 500rpm			
solids				
comment		'Allen Brock' medium		
Ferric iron conc.				
inoculum				
culture			*S. solfataricus*	*A. brierleyi*

B. Solubility of Various Gases in Water

B.1 PREDICTING EQUILIBRIUM SOLUBILITY FOR OXYGEN AND OTHER GASES

Several correlations, which elucidate the effect of temperature on gas solubility, are reported in the literature. These are specific for particular gas solvent systems (Winkler, 1981).

B.1.1 HENRY'S LAW CONSTANTS

Below a pressure of five bar, proportionality of solubility and partial pressures (Henry's Law) can be assumed in the case of both O_2 and CO_2 without introducing appreciable errors (Schumpe *et al.*, 1982). Thus:

$$p = H \cdot x \qquad \textit{Equation B-1}$$

where

p	partial pressure of the solute in the gas phase	atm
H	Henry's law constant	atm
x	mole fraction of the solute in the liquid phase	

The proportionality constant H is detailed as a function of solute and temperature in Table B-1.

Table B-1: Solubility of various gases in water at different temperatures, (Perry et. al., 1988).

Temperature °C	Henry's Law constant, H (atm x 10^{-4})		
	O_2	N_2	CO_2
10	3.27	6.68	0.104
20	4.01	8.04	0.142
30	4.75	9.24	0.186
40	5.35	10.4	0.233
50	5.88	11.3	0.283

The Henry's law constant is an increasing function of temperature, which can be described by (Danckwerts, 1970):

$$\frac{d(\ln H)}{d(\frac{1}{T})} = \frac{\Delta H_a}{14.6 + T} \qquad \textit{Equation B-2}$$

where

T	absolute temperature
R	Universal Gas constant
ΔH_a	heat of absorption of the gas, at temperature T

Log H as a function of the inverse of temperature is a straight line over a range of temperature small enough for the heat of absorption to be constant. This can be used to predict solubilities over a range of temperatures.

B.1.2 MONTGOMMERY CORRELATION

Montgommery *et al.* (1964) developed an empirical equation for the solubility of oxygen in pure water in equilibrium with air at a pressure of 1atm over a range of 0 to 33°C.

$$C^* = \frac{468}{14.6 + T} \qquad \textit{Equation B-3}$$

where

T	temperature	°C
C*	saturation concentration of O_2	$mg \cdot l^{-1}$

B.1.3 ATKINSON AND MAVITUNA CORRELATION

The variation of solubility of oxygen with temperature has been correlated by (Atkinson and Mavituna, 1992):

$$C^* = 14.16 - 0.394 \cdot T + 0.007714 \cdot T^2 - 0.000646 \cdot T^3 \qquad \textit{Equation B-4}$$

where

T	temperature	°C
C^*	saturation concentration of O_2	$mg \cdot l^{-1}$

B.1.4 CABANI CORRELATION

Cabani and Gianni (1986) correlate the solubility of certain gases in known solvents at known fixed pressures across a range of temperatures in the form of:

$$\ln x = a + \frac{b}{T} + c \cdot \ln T + d \cdot T + e \cdot T^2 \qquad \textit{Equation B-5}$$

where

T	absolute temperature	K
a, b, c, d and e	coefficients (Table B-2)	
x	mole fraction of the solute gas in the solvent	

B.1.5 WILHELM CORRELATION

Wilhelm *et al.* (1977) considered experimental data for equilibrium solubility of several gases, including oxygen, carbon dioxide, air and nitrogen. The region of interest is temperatures below 100°C and at a pressure not exceeding 5 atmospheres. The data was fitted to a function of the form of Equation B-6. The constants *a*, *b*, *c* and *d* are detailed in Table B-2.

$$R \cdot \ln x = a + \frac{b}{T} + c \cdot \ln T + d \cdot T \qquad \textit{Equation B-6}$$

Table B-2: Solubility coefficients for various gases in water (Cabani and Gianni, 1986; Wilhelm et. al., *1977).*

Solute	Temperature Range (K)	a	b	c	d	e
		Cabani and Gianni (1986) correlation (@ 1.013 bar): $\ln x = a + \frac{b}{T} + c \cdot \ln T + d \cdot T + E \cdot T^2$				
CO_2	273-373	-4957.824	105288.4	933.1700	-2.854886	1.480857×10^{-3}
O_2	273-333	-1072.48902	27609.2617	191.886028	-0.40830901	2.244445×10^{-4}
Air	273-373	-388.760	14097.6	61.2018	-0.0617537	
N_2	273-348	-181.5870	8632.129	24.79808		
		Wilhelm *et. al. (1977)* correlation: $R \ln x = a + \frac{b}{T} + c \cdot \ln T + d \cdot T$				
		$cal \cdot K^{-1} \cdot mol^{-1}$	$cal \cdot mol^{-1}$	$cal \cdot K^{-1} \cdot mol^{-1}$	$cal \cdot K^{-2} \cdot mol^{-1}$	
CO_2	273-353	-317.658	17371.2	43.0607	-0.00219107	
O_2	274-348	-286.942	15450.6	36.5593	0.0187662	
Air	273-373	-319.323	15492.6	43.0259	0.0196194	
N_2	273-346	-327.850	16757.6	42.8400	0.0167645	

B.2 HENRY'S CONSTANT CORRELATION

Solubility data and correlations from Cabani and Gianni (1986), Wilhelm *et al.* (1977) and Liley *et al.* (1988) were used over the specified temperature ranges to fit Henry's constants were fitted to a power series:

$$H = a + b.T + c.T^2 + d.T^3$$

$$x = \frac{p}{H}$$

Equation B-7

where

H	Henry's constant	atm x 10^{-4}
x	gas solubility from Henry's Law	mole fraction
T	temperature	°C
p	partial pressure of the gas	atm

The Henry's constants are expressed in units of pressure per gas mole fraction (in water) x 10^{-4} (Liley *et al.*, 1988). The values for coefficients *a,b,c* and *d* are given in Table B-3. As a result, the same correlation can be used for the solubility of gases when mixed with a carrier gas, *i.e.* in air or nitrogen. By combining the correlations, the maximum temperature range is utilised.

Table B-3: Henry's Law correlation coefficients.

	CO_2	O_2	Order
temperature range (°C)	0-100	0-80	
a	0.0744	2.4788	0
b	0.0024	0.079	1
c	5×10^{-5}	-1×10^{-4}	2

	CO_2	O_2	Order
d	$-3x10^{-7}$	$-2x10^{-6}$	3

B.3 SOLUBILITY OF OXYGEN IN WATER

Experimental data of various authors, summarised by Atkinson and Mavituna (1992) is displayed in Table B-4.

Table B-4: Experimental data, mole fraction of oxygen in water at various temperatures, taken from Atkinson and Mavituna (1992).

Temperature °C	Finn (1967) ($1x10^5$)	Phillips (1968) ($1x10^5$)
0	3.9242	
10	3.0602	
15	2.7722	
20	2.4842	2.5201
25	2.2681	
30	2.0881	
35	1.9621	
40	1.8541	

In Figure B-1, experimental data (Table B-4) for the solubility of oxygen in water at partial pressure 1 atm and temperature range 0-50 °C is compared with the correlations discussed. Similar correlations for oxygen solubility in the presence of air are presented.

B.4 SOLUBILITY OF CARBON DIOXIDE IN WATER

Carbon dioxide provides the carbon source to microorganisms in the bioleaching system. Hence its solubility is

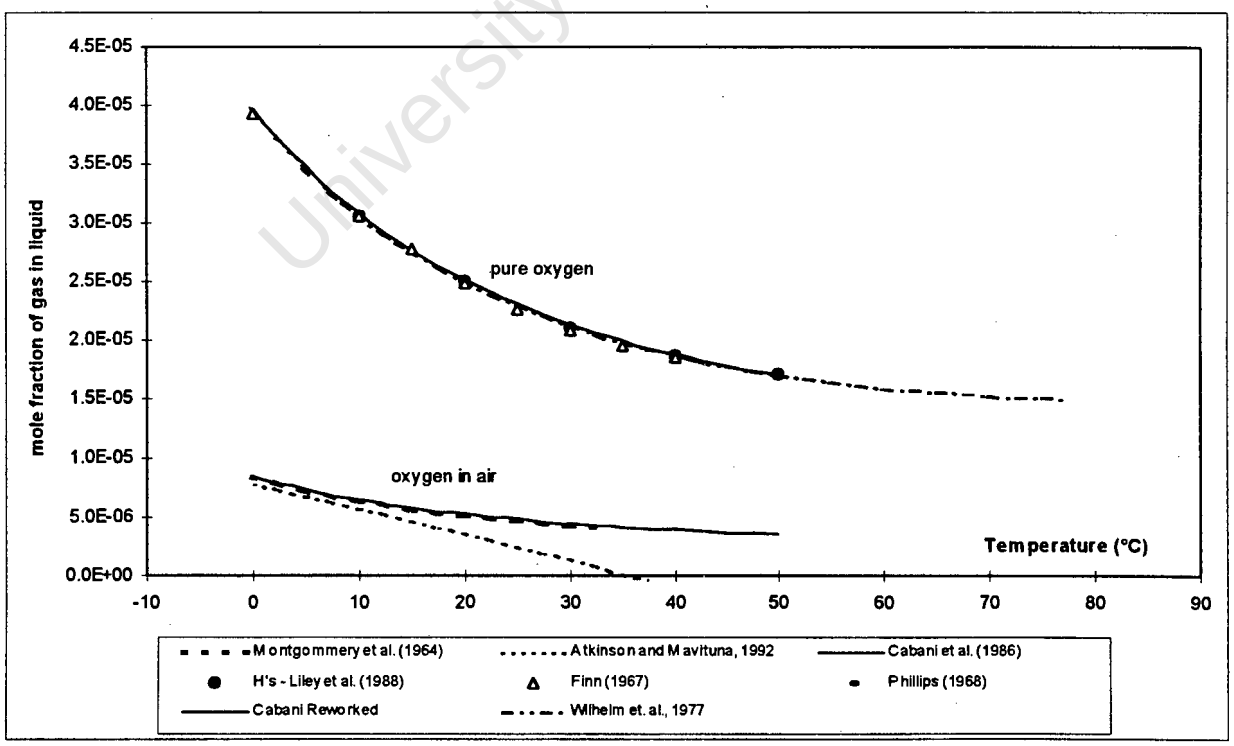


Figure B-1: Solubility of oxygen in water predicted by various methods.

of interest. Danckwerts and Sharma (1966) developed a correlation for the solubility of carbon dioxide in the temperature range between 0 and 25°C:

$$\log_{10}\frac{C^*}{P_T} = \frac{1140}{T} - 5.30 \qquad \textit{Equation B-8}$$

where

C^*	solubility of CO_2	$mol \cdot l^{-1}$
P_T	total pressure	atm
T	temperature (273-298K)	K

In Figure B-2, the correlations of Cabani and Gianni (1986), Wilhelm *et al.* (1977) and Danckwerts and Sharma (1966) and Henry's constants of Liley *et al.* (1988) are compared with solubilities predicted by the Henry's constant correlation (Section B-2).

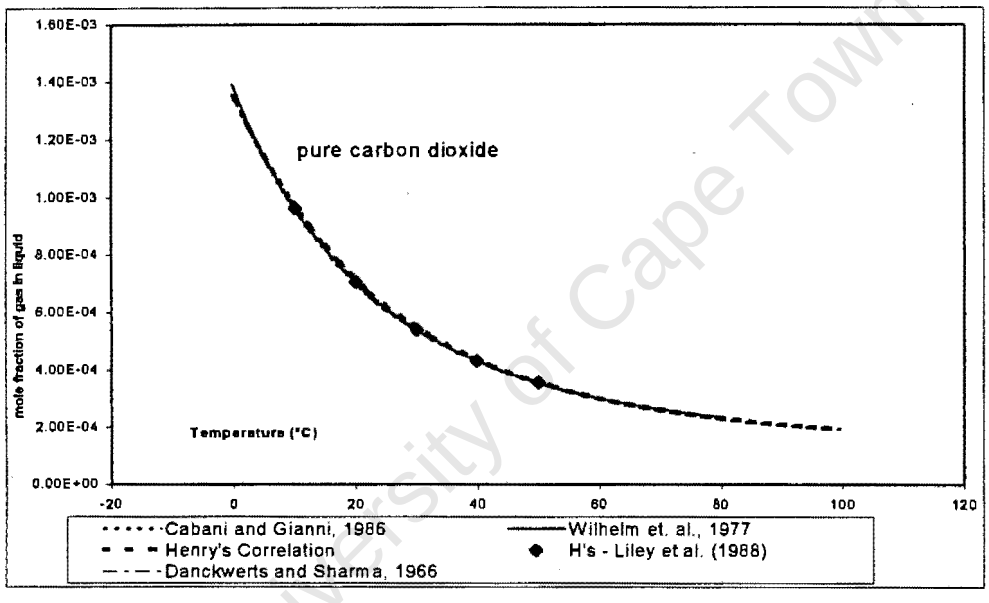


Figure B-2: Solubility of carbon dioxide in water, predicted by various methods.

B.5 VAPOUR PRESSURE OF LIQUID WATER

As temperature increases, the vapour pressure of water increases, as shown in Figure B-3 (Liley *et. al.*, 1988), according to:

$$p^*_{WG} = 5.034 + 0.1276 \cdot T + 0.0268 \cdot T^2 - 0.0002 \cdot T^3 + 7 \times 10^{-6} \cdot T^4 \qquad \textit{Equation B-9}$$

where

p^*_{WG}	vapour pressure of water	mmHg
T	temperature	°C

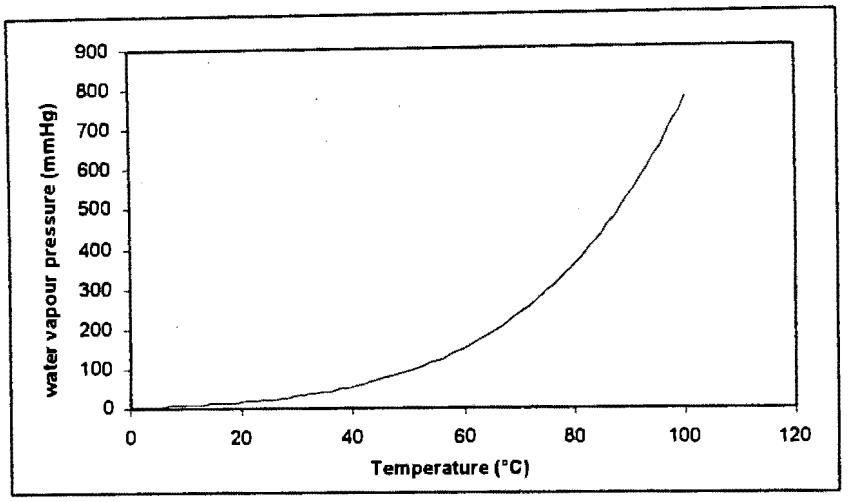


Figure B-3: Variation of the vapour pressure of liquid water with temperature (Liley et al., *1988).*

B.6 CORRECTED GAS SOLUBILITIES

Equilibrium solubilities need to be corrected to take the increase in water vapour pressure into account (Boogerd *et al.*, 1990). This can be incorporated into the Henry's law equation as follows:

$$x = \frac{p\left(1 - p^{*}_{WG}\right)}{p_T} \cdot \frac{p_T}{H} \qquad \textit{Equation B-10}$$

where

H	Henry's Law constant	atm
p	partial pressure of the gas in the mixture	atm
p_T	total pressure	1 atm
p^{*}_{WG}	vapour pressure of water	atm

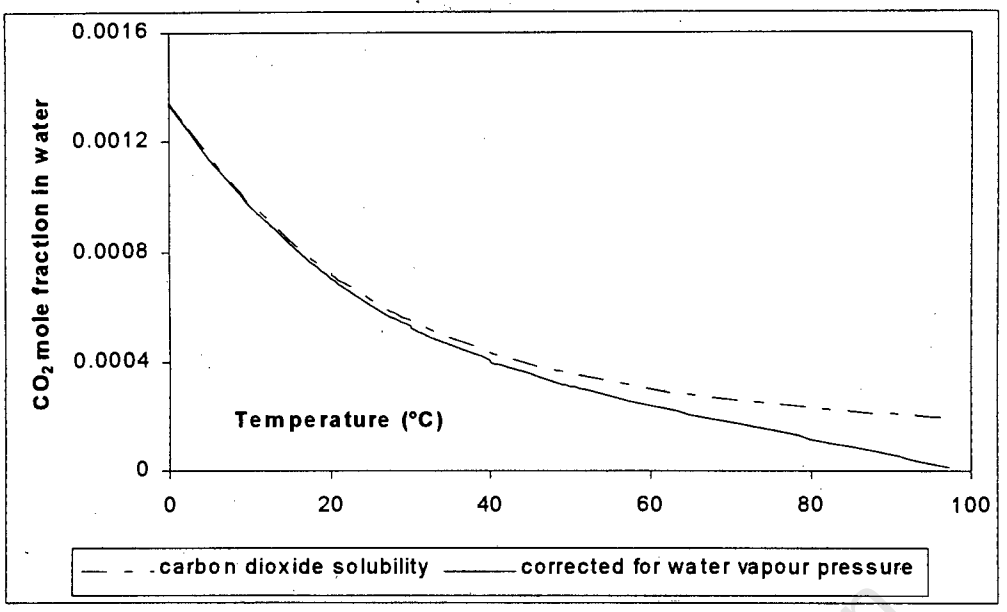


Figure B-4: Carbon dioxide solubility with and without water vapour pressure correction.

Figures B-4 and B-5 display the solubilities of oxygen and carbon dioxide across the temperature range 0-100°C with and without water vapour correction.

B.7 EQUILIBRIUM OF GASES IN DRY AIR

The equilibrium partial pressures of carbon dioxide and oxygen in dry air are 0.00033 and 0.2096 atm repectively (Weast and Astle, 1982).

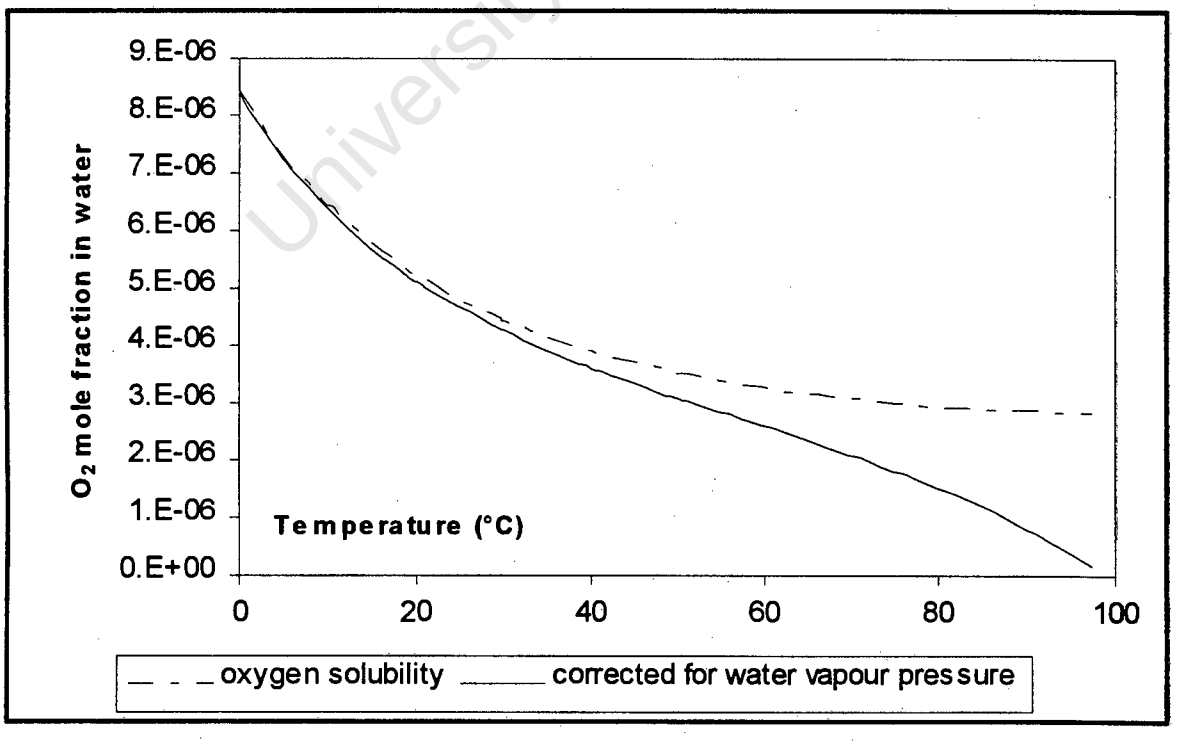


Figure B-5: Oxygen solubility with and without water vapour pressure correction.

B.8 NOMENCLATURE

symbol	*description*	*units*
C^*	saturation concentration of O_2	$mg \cdot l^{-1}$
H	Henry's constant	atm x 10^{-4}
ΔH_a	heat of absorption (negative) of the gas, at temperature T	
R	Universal Gas constant	
T	temperature	°C, K
a,b,c,d and e	coefficients	
p	partial pressure of the gas	atm
p_T	total pressure	1 atm
p^*_{WG}	vapour pressure of water	mmHg
x	mole fraction of the solute in the liquid phase	

B.9 REFERENCES

Atkinson, B. and F. Mavituna (1992): "Biochemical Engineering and Biotechnology Handbook", Macmillan, New York, Chapter 12, p 697-751

Boogerd, F.C., P.Bos, J.G. Kuenen, J.J. Heijnen and R.G.J.M. van der Lans (1990): "Oxygen and carbon dioxide mass transfer and the aerobic, autotrophic cultivation of moderate and extreme thermophiles: A case study related to the microbial desulfurisation of coal", *Biotechnology and Bioengineering*, **35**, 1111-1119

Cabani, S. and P.Gianni (1986): in *Thermodynamic Data for Biochemistry and Biotechnology*, Ed. H.J. Hintz, Springer Verlag, Berlin; p 261-262; Cited in Sandler (1989) p 439-440

Danckwerts, P.V. (1970): "Gas-Liquid Reactions", McGraw-Hill Inc., United States of America, p 6-20

Danckwerts, P.V. and M.M. Sharma (1966): "The adsorption of carbon dioxide into solutions of alkalis and amines", *The Chemical Engineer*, October, CE244-CE280

Montgomery, H.A.C., N.S. Thom and A. Cockburn (1964): "Determination of dissolved oxygen solubility by the Winkler method and the solubility of oxygen in pure water and sea water", *J. Appl. hem.*, **14**, 280-296; Cited in Atkinson and Mavituna (1992)

Liley, P.E., R.C. Reid and E.Buck (1988): "Physical and chemical data", in *Perry's Chemical Engineers' Handbook, 6th Edition*, Eds R.H. Perry, D.W. Green and J.O. Maloney", McGraw-Hill Book Company, United States of America, p 3-273-274, 3-278-282, 3-285-289

Sandler, S.I. (1989): "Chemical and Engineering Thermodynamics, 2nd Edition", John Wiley & Sons, Canada, p 439-440

Schumpe, A., G. Quicker and W.D. Deckwer (1982): "Gas Solubilities in Microbial Culture Media", *Advances in Biochemical Engineering*, **24**, 1-38, Springer Verlag, Berlin

Weast, R.C. and M.J. Astle (1982): "Handbook of Chemistry and Physics 63rd Edition", CRC Press Inc. Florida

Wilhelm, E., R. Battino and R.J. Wilcock (1977): "Solubility of Gases in Water", *Chemical Reviews*, **77**, 219-262

Winkler, M. (1981): "Biological Treatment of Waste Water", Ellis Horwood Ltd. Publishers, Chicester, p 42-76

C. Effect of Electrolytes on Gas Solubilities

C.1 SECHENOV EQUATION

For low concentrations of electrolytes (C_{el}) and gases $\gamma(C_j)$, the interactions can be neglected and the logarithm of the gas activity coefficient can be expressed as a linear function of both concentrations:

$$\log\gamma_j = k_s C_{el} + k_j C_j$$

$$\gamma_j C_j = \gamma_{jH_2O} C_{jH_2O}$$

$$\log\gamma_j = \log\frac{C_{jH_2O}}{C_j} + \log\gamma_{jH_2O} \qquad \text{Equation C-1}$$

$$k_s C_{el} + k_j C_j = \log\frac{C_{jH_2O}}{C_j} + \log\gamma_{jH_2O}$$

$$\log\frac{C_{jH_2O}}{C_j} = k_s C_{el} + k_j (C_j - C_{jH_2O})$$

If k_j or $C_j - C_{jH_2O}$ are small, then this transforms into the well known empirical Sechenov equation (Schumpe *et al.*, 1982; Atkinson and Mavituna, 1992):

$$\log\frac{C_{jH_2O}}{C_j} = K_s C_{el} \qquad \textit{Equation C-2}$$

where

C_{el}	electrolyte concentration	$mol \cdot l^{-1}$
C_j, C_{jH_2O}	concentration of the gas, and concentration of the gas in water	$mol \cdot l^{-1}$
γ_j	activity coefficient of the gas	
γ_{jH_2O}	activity coefficient of the gas in water	
k_s	specific salt constant	$l \cdot mol^{-1}$
k_j	specific gas constant	$l \cdot mol^{-1}$
K_s	Sechenov constant	$l \cdot mol^{-1}$

Salting-out effects can usually be fairly well described by the so-called Sechenov constants, K_s, which are specific with respect to gas, temperature and salt. (Schumpe *et al.*, 1982; Danckwerts, 1970).

C.2 PREDICTING SALTING OUT COEFFICIENTS (K_S)

Several theories predict salting out-coefficients. The fundamental ideas can be classified into hydration, electrostatic, internal pressure and scaled-particle approach. A detailed review of this is covered in McDevit and Long (1952).

The **hydration** theories are based on the idea that the decrease in gas solubility is due to the hydration of the ions which reduces the volume of water available to the solute gas. These theories do not explain the dependency of salting-out effects on the type of gas. The **electrostatic** approach relates to the work of Debye

and McAuley. Debye is well acclaimed for his work with activity coefficients (Atkins, 1983). This theory allows for the effect of the dissolved gas on the dielectric constant of the solution and correctly predicts K to increase with increasing charge and decreasing ionic radius. The **internal pressure** concept introduced by McDevit and Long (1952) considers an effective pressure related to volume and compressibility changes due to electrolyte-solvent interaction. Although many of the effects are correctly described, its applicability suffers from lack of model parameters of sufficient accuracy, as do most theoretical approaches. A promising concept is based on the **scaled-particle** theory, which considers the dissolution process to consist of the creation of a cavity in the solvent and the introduction of a gas molecule, which then interacts with the solvent (Schumpe *et al.*, 1982).

C.3 SALTING OUT EFFECT IN TERMS OF IONIC STRENGTH

When implementing the Sechenov equation:

$$\log \frac{C_{jH_2O}}{C_j} = K_s C_{el} \qquad \textit{Equation C-3}$$

where

C_{el}	electrolyte concentration	$mol \cdot l^{-1}$
C_j, C_{jH_2O}	concentration of the gas, and concentration of the gas in water	$mol \cdot l^{-1}$
K_s	Sechenov constant	$l \cdot mol^{-1}$

a deviation from linearity occurs at high electrolyte concentrations. The solubility predicted by this equation becomes too low. The critical electrolyte concentration at which the Sechenov equation is no longer valid varies with the electrolyte mixture. Hence Van Krevelen and Hoftijzer (1948) correlated $\left(\log \frac{C_{jH_2O}}{C_j}\right)$ with respect to the ionic strength instead of merely the electrolyte concentration:

$$\log \frac{C_{jH_2O}}{C_j} = h \cdot I$$

$$h = h_+ + h_- + h_G$$

$$I = \frac{1}{2}\sum_i C_i z_i^2 \qquad \text{Equation C-4}$$

$$C_i = x_i C_{el}$$

$$K_s = h\frac{1}{2}\sum_i x_i z_i^2$$

where

I	ionic strength
C_i	ionic species concentration
z_I	ionic species charge
I	ionic species being considered
h_+, h_- and h_G	empirical parameters specific for the cations, anions and the gas respectively
x_i	fraction of ions as type i in the salt

Except for very high solubilities, the logarithm of concentrations is equal to the equivalent logarithm of Henry constants (Section C-1.1) according to (Schumpe *et al.*, 1982):

$$\log \frac{C_{jH_2O}}{C_j} = \log \frac{H_j}{H_{jH_2O}} \qquad \text{Equation C-5}$$

For a mixed electrolyte, van Krevelen and Hoftijzer (1948):

$$\log \frac{C_{jH_2O}}{C_j} = h_1 \cdot I_1 + h_2 \cdot I_2 + \ldots \qquad \text{Equation C-6}$$

Parameters for the van Krevelen and Hoftijzer model are detailed in Tables C-1 and C-2.

Table C-1: h_G (1 mole^{-1}) for the van Krevelen-Hoftijzer model (Schumpe, et al., 1982).

Temperature (°C)	H_2	N_2	O_2	CO_2
0			-0.1653	-0.2110
5	-0.2106			
10	-0.2170			
15	-0.2197		-0.1786	-0.2222
20	-0.2132		-0.1771	
25	-0.2115	-0.1904	-0.1892	-0.2277
40				-0.2327

Table C-2: Ion specific parameters (h_+, h_-) for the van Krevelen-Hoftizer model (Schumpe et al., 1982).

Cation	h_+, 1 mole^{-1}	Anion	h_-, 1 mole^{-1}
H^+	-0.1110	Cl^-	0.3416
Li^+	-0.0416	Br^-	0.331
Na^+	-0.0183	J^-	0.3124
K^+	-0.0362	OH^-	0.3875
Rb^+	-0.0449	NO_3^-	0.3230
Cs^+	-0.0584	CNS^-	0.2612
NH_4^+	-0.0737	HS^-	0.3718
		HSO_3^-	0.3869
Mg^{2+}	-0.0568	HCO_3^-	0.4286
Ca^{2+}	-0.0547		
Ba^{2+}	-0.0473	SO_4^{2-}	0.3446
Sr^{2+}	-0.0445	SO_3^{2-}	0.3275
Mn^{2+}	-0.0625	CO_3^{2-}	0.3754
Fe^{2+}	-0.0602		
Co^{2+}	-0.0534	PO_4^{3-}	0.3265
Ni^{2+}	-0.0520		
Zn^{2+}	-0.0590		
Cd^{2+}	-0.0062		
Al^{3+}	-0.0726		
Cr^{3+}	-0.0986		

Schumpe *et al.* (1978) observed that this method of prediction (Equation C-4) was physically inconsistent in mixed electrolyte solutions. Hence, they suggested the following relationship:

$$\log \frac{C_{jH_2O}}{C_j} = \sum_i H_i I_i \qquad \text{Equation C-7}$$

$$I_i = \frac{1}{2} C_i z_i^2$$

In this correlation, I_i is the ionic strength attributable to a single ion i and the parameter H_i is specific to the gas, the ion and the temperature. For single salt solutions, the Sechenov constants are then given by:

$$K_s = \frac{1}{2} \sum_i H_i x_i z_i^2 \qquad \text{Equation C-8}$$

H_i parameters are listed in Tables C-3 and C-4.

Table C-3: Parameters H_I (of cations in 1 mole^{-1}) for the model suggested by Schumpe et al. (1978).

Cations	H_i (O_2) 20°C	H_i (O_2) 25°C	H_i (O_2) 37°C	H_i (CO_2) 25°C
H^+	-0.771	-0.776	-0.803	-0.319
Li^+	-0.655	-0.675	-0.636	-0.178
Na^+	-0.570	-0.568	-0.577	-0.130
K^+	-0.593	-0.587	-0.578	-0.196
Rb^+		-0.618	-0.604	-0.217
Cs^+	-0.613	-0.659	-0.612	-0.243
NH_4^+		-0.704	-0.681	-0.252
Net_4^+			-0.709	
Mg^{2+}	-0.308	-0.297	-0.321	-0.078
Ca^{2+}	-0.293	-0.309	-0.316	-0.073
Ba^{2+}		-0.291	-0.299	-0.064
Mn^{2+}		-0.324	-0.325	-0.084
Fe^{2+}				-0.078
Co^{2+}			-0.317	
Ni^{2+}	-0.302		-0.318	
Cu^{2+}	-0.312		-0.325	-0.090
Zn^{2+}	-0.295		-0.310	
Cd^{2+}	-0.316		-0.320	
Al^{3+}	-0.210		-0.221	
La^{3+}			-0.216	
Ce^{3+}			-0.216	
Fe^{3+}			-0.244	
Th^{4+}			-0.168	

Table C-4: Parameters H_i (of anions in 1 mole^{-1}) for the model suggested by Schumpe et al. (1978)

Anions	H_i (O_2)			H_i (CO_2)
	20°C	25°C	37°C	25°C
F^-			0.867	
Cl^-	0.843	0.849	0.861	0.339
Br^-	0.840	0.820	0.822	0.324
J^-		0.784		0.309
OH^-	0.955	0.943	0.917	
NO_3^-	0.827	0.802	0.821	0.293
SCN^-			0.791	
BF_4^-			0.775	
ClO_4^-		0.890		
HSO_4^-	0.945	0.955	0.935	0.436
HSO_3^-				0.400
HCO_3^-		1.076	0.861	
$H_2PO_4^-$		0.997		
C_6H_5-O-Ac$^-$		0.755		
SO_4^{2-}	0.458	0.460	0.448	0.213
$S_2O_3^{2-}$		0.455		0.211
SO_3^{2-}				
CO_3^{2-}		0.467	0.447	
HPO_4^{2-}		0.477		
PO_4^{3-}		0.308		
$Mo_7O_{24}^{6-}$			0.155	

C.4 NOMENCLATURE

Symbol	*Description*	*Units*
I	ionic strength	
C_i	ionic species concentration	
z_I	ionic species charge	
i	ionic species being considered	
h_+, h_- and h_G	empirical parameters specific for the cations, anions and the gas respectively	
x_i	fraction of ions as type i in the salt	
C_{el}	electrolyte concentration	mol·l^{-1}
C_j, C_{jH_2O}	concentration of the gas, and concentration of the gas in water	mol·l^{-1}
γ_j	activity coefficient of the gas	
γ_{jH_2O}	activity coefficient of the gas in water	
k_s	specific salt constant	l·mol^{-1}
k_j	specific gas constant	l·mol^{-1}
K_s	Sechenov constant	l·mol^{-1}

C.5 REFERENCES

Danckwerts, P.V. (1970): "Gas-Liquid Reactions", McGraw-Hill Inc., United States of America, p 6-20

Danckwerts, P.V. and M.M. Sharma (1966): "The adsorption of carbon dioxide into solutions of alkalis and amines", *The Chemical Engineer*, October, CE244-CE280

McDevit, W.F. and F.A. Long (1952): "Effect of Salts on Activity Coefficients", *J. Am. Chem. Soc.*, **74**, 1773

Reid, R.C., J.M. Prausnitz and T.K. Sherwood (1977): "The Properties of Gases and Liquids, 3rd Edition", McGraw-Hill Book Company, United States of America

Schumpe, A., G. Quicker and W.D. Deckwer (1982): "Gas Solubilities in Microbial Culture Media", *Advances in Biochemical Engineering*, **24**, 1-38, Springer Verlag, Berlin

Schumpe, A., I. Adler and W. –D. Deckwer (1978): "Solubility of Oxygen in Electrolyte Solutions", *Biotechnology and Bioengineering*, **20**, 145-150

Van Krevelen, D.W. and D.W. Hoftijzer (1948): *Chimie et Industrie*, p. 168, Numéro Spéciale du XXIe Congrés International de Chimie Industrielle, Bruxelles; cited in Schumpe *et al.*, 1982

Winkler, M. (1981): "Biological Treatment of Waste Water", Ellis Horwood Ltd. Publishers, Chicester, p 42-76

D. Debye-Hückel Activity Coefficient Model

D.1 MODEL DESCRIPTION

The Debye-Hückel model, one of the first theoretical models developed has been described as the corner stone of recent activity coefficient models (Zemaitis *et al.*, 1986). An activity coefficient enables the conversion of concentrations to activities according to

$$a_i \approx \gamma_i C_i \qquad \textit{Equation D-1}$$

where

a_i	activity of species *i*
γ_i	activity coefficient of species *i*
C_i	concentration of species *i*

Electrolyte solutions are particularly non-ideal as there are substantial electrostatic interactions between the ions (Atkins, 1983). Since strong electrolytes are highly dissociative in solution, the ion concentration is higher with the resulting distance between them smaller than for weaker electrolytes. This increase in concentration results in the tendency towards an orderly distribution of ions. Electrostatic forces cause mutual attraction between oppositely charged ions. The Debye-Hückel model takes these coloumbic interactions into account and provides a platform upon which the effect of ionic changes can be investigated (Atkins, 1983). Their expression for γ_i is given by:

$$-\log_{10}\gamma_i = \frac{Az_i^2\sqrt{I}}{1+d_iB\sqrt{I}} \qquad \textit{Equation D-2}$$

where

γ_i	activity coefficient of species *i*	
A and B	Debye-Hückel constants at the specified temperature and pressure	
I	ionic strength of the solution	mol of ion/l
z_i	charge on the species	
d_i	measure of the effective ion diameter	Å

The ionic strength is given by

$$I = \frac{1}{2}\sum m_i z_i^2 \approx \frac{1}{2}\sum C_i z_i^2 \qquad \textit{Equation D-3}$$

where

m_i	molality of species *i*
C_i	concentration of species *i*

Although the Debye-Hückel model was originally derived for dilute solutions ($I < 0.1$), some success has been reported in applying it at higher concentrations, particularly when ion-pair equilibria are taken into account (Dry and Bryson, 1988).

D.2 MODEL CONSTANTS

The Debye-Hückel constants A and B are defined as follows (Zemaitis *et al.*, 1986):

$$A = \frac{1}{2.3026}\left(\frac{e}{\sqrt{k \cdot \varepsilon \cdot T}}\right)^3 \sqrt{\frac{2\pi \cdot N_A \cdot \rho_s}{1000}}$$

$$B = \sqrt{\frac{8\pi \cdot N_A \cdot e^2 \cdot \rho}{1000 \cdot k \cdot \varepsilon \cdot T}}$$

Equation D-4

where

T	absolute temperature	K
N_a	Avogadro's Number, $6.022\ 1367 \times 10^{23}$	mol^{-1}
e	elementary charge $1.602\ 177\ 33 \times 10^{-19}$	C
k	Boltzman constant, $[R/N_a]$ $1.380\ 658 \times 10^{-23}$	J/K
ε	dielectric constant	

D.3 LIMITATIONS OF THE DEBYE-HÜCKEL LIMITING LAW

Due to the simplifications and assumptions made in considering the potential of the ionic atmosphere, the Debye-Hückel model is theoretically valid for dilute solutions (ionic strength of 0.1 molal or less) (Zemaitis *et al.*, 1986). Despite this, Dry and Bryson (1988) recommend the use of this model for estimation of the activity coefficients of several divalent metal sulphates up to ionic strengths as high as 4. Bryson and Nicol (1996) concurs that the use of this model at high ionic strengths will introduce errors, however it is useful as a first approximation.

The effective ionic diameter is not a measurable quantity, but is chosen close to the value of the hydrated radius. This number is often between 3.5 and 6.2 Å (Zemaitis *et al.*, 1986).

D.4 DRY (1984) AND CRUNDWELL (1988) CONSTANT DATA

Dry (1984) and Crundwell (1988) made use of polynomial equations to calculate the values of the Debye-Hückel constants at different temperatures. The temperature parameter is specified according to *atemp* as follows:

$$atemp = t - 298.15$$

$$A = 0.4919 + 7.143 \times 10^{-4}(atemp) + 2.113 \times 10^{-6}(atemp)^2 + 1.173 \times 10^{-8}(atemp)^3$$

$$B = 0.3249 + 2.099 \times 10^{-4}(atemp) - 2.582 \times 10^{-8}(atemp)^2 + 2.589 \times 10^{-}(atemp)^3$$

Equation D-5

where

atemp	adjusted absolute temperature	K
t	temperature	°C

Values used for the effective ionic diameter, d_i, (Equation E-2) are listed in Table D-1.

Table D-1: Values of the effective ionic diameter parameter (Dry, 1984).

Species	d_i (Å)
Fe^{3+}	9
H^+	9
SO_4^{2-}	4
Fe^{2+}	6
OH^-	3
$FeH(SO_4)_2$	-
$FeHSO_4^{2+}$	8
$Fe(SO_4)_2^-$	8
$FeSO_4^+$	8
HSO_4^-	4
$FeSO_4$	-
$FeHSO_4^+$	4

D.5 NOMENCLATURE

symbol	*description*	*unit*
A and B	Debye-Hückel constants at the specified temperature and pressure	
C_i	concentration of species i	
I	ionic strength of the solution	mol of ion/l
N_a	Avogadro's Number, $6.022\ 1367 \times 10^{23}$	mol^{-1}
T	absolute temperature	K
atemp	adjusted absolute temperature	K
a_i	activity of species i	
d_i	measure of the effective ion diameter	Å
e	elementary charge $1.602\ 177\ 33 \times 10^{-19}$	C
k	Boltzman constant, $[R/N_a]$ $1.380\ 658 \times 10^{-23}$	J/K
m_i	molality of species i	
t	temperature	°C
z_i	charge on the species	
ε	dielectric constant	
γ_i	activity coefficient of species i	
π	pi, 3.1416	

D.6 REFERENCES

Atkins, P.W. (1983): "Physical Chemistry", Oxford University Press, Oxford

Bryson, A.W. and M. Nicol (1996): "Thermodynamics of Aqueous Solutions", Continuing Engineering Education Course, University of Witwatersrand, South Africa

Crundwell, F.K. (1988): "The role of charge-transfer in the oxidative and non-oxidative dissolution of sphalerite", PhD Thesis, University of Witwatersrand, South Africa

Dry, M.J. and A.W. Bryson (1988): "Prediction of Redox Potential in Concentrated Iron Sulphate Solutions", *Hydrometallurgy*, **21,** 59-72

Dry, M.J. (1984): "Kinetics of Leaching of a Low Grade Matte in Ferric Sulphate Solution", PhD Thesis, University of Witwatersrand, South Africa

Zemaitis, J.F. Jr; D.M. Clark, M. Rafal and N. C Scrivner (1986): "Handbook of Aqueous Electrolyte Thermodynamics", American Institute of Chemical Engineers, New York, pp 47-66

E. Pitzer's Correlation for Sulphuric Acid

Pitzer's correlation (Pitzer *et al.*, 1977) is used to express the thermodynamic properties for sulphuric acid. It can also be used to predict the extent of dissociation in solution and the related solute state. Pitzer's correlation looks particularly at dilute solutions of sulphuric acid. Dry (1984) and Crundwell (1988) implemented this correlation in their estimation of speciation in bioleaching.

E.1 CORRELATION EQUATIONS

Sulfuric acid dissociates to varying extents according to:

$$H_2SO_4 \leftrightarrow HSO_4^- + H^+$$

$$HSO_4^- \leftrightarrow SO_4^{2-} + H^+$$

$$K_2 = \left(\frac{m_H m_2}{m_1}\right)\left(\frac{\gamma_H \gamma_2}{\gamma_1}\right) \qquad \textit{Equation E-1}$$

where

K_2	equilibrium constant for the dissociation of the bisulphate ion
m_H	molality of H^+
m_1	molality of HSO_4^-
m_2	molality of SO_4^{2-}
γ_H, γ_1 & γ_2	activity coefficient of H^+, HSO_4^- and SO_4^{2-}

With two different anions although the second electrolyte arises from dissociation of the first, we have a mixed electrolyte. For the osmotic coefficient, ϕ', on a mixed electrolyte basis one finds:

$$\phi' - 1 = \left(\sum m_i\right)^{-1}\left\{2I \cdot f^{\phi} + 2m_H\left[m_i\left(B_{H_1}^{\phi} + m_H C_{H_1}^{\phi}\right) + m_2\left(B_{H_2}^{\phi} + \frac{m_H C_{H_2}^{\phi}}{\sqrt{2}}\right)\right] + m_1 m_2\left(\vartheta_{12} + m_H \psi_{H_{12}}\right)\right\}$$

$$f^{\phi} = -A_{\phi}\left(\frac{\sqrt{I}}{1+1.2\sqrt{I}}\right)$$

$$B_{MX}^{\phi} = \beta_{MX}^{(0)} + \beta_{MX}^{(1)} \exp\left(-\alpha\sqrt{I}\right) \qquad \textit{Equation E-2}$$

$$I = m + 2m_2$$

$$A_{\phi} = 0.3770 + 4.684\times10^{-4}\left(T - 273.15\right) + 3.74\times10^{-6}\left(T - 273.15\right)^2$$

where

ϕ'	osmotic coefficient
I	ionic strength
m	stoichiometric molality of sulphuric acid
m_H	molality of H^+
$m_1 = 2m - m_H$	molalitly of HSO_4^-
A_{ϕ}	Debye-Hückel limiting law parameter for the osmotic coefficient, equation valid between 0 and 55°C
α	general empirical parameter (equal to 2)

$\beta_{MX}^{(0)}$, $\beta_{MX}^{(1)}$	specific parameters for the appropriate sum of λ_{ij} for binary interactions M-X, M-M and X-X where the cation M is always H and the anion X is either HSO_4^- (=1) or SO_4^{2-} (= 2)
C_{MX}^{ϕ}	corresponding third virial coefficient for triple interactions μ_{ijk} for MMX and MXX
ϑ_{12}	the difference in binary interaction λ_{ij} of HSO_4^- with SO_4^{2-}
ψ_{H12}	corresponding difference in triple interactions μ_{ijk} involving a hydrogen ion with two anions

In sulphuric acid solutions of concentration greater than 0.1M, HSO_4^- is the more abundant ion. These equations reduce to:

$$\phi'-1=\left(\sum m_i\right)^{-1}\left\{2I\cdot f^{\phi}+2m_H\left[m_i B_{H_1^{\phi}}+m_2\left(\beta_{H2}^{(0)}+\frac{m_H C_{H_2^{\phi}}}{\sqrt{2}}\right)\right]\right\} \qquad \textit{Equation E-3}$$

$$B_{H1}^{\phi}=\beta_{H1}^{(0)}+\beta_{H1}^{(1)}\exp\left(-2\sqrt{I}\right)$$

The corresponding equations for the two combinations of activity coefficients of interest are:

$$\ln\left(\gamma_H^2\gamma_{SO_4}\right)=6f^{\gamma}+4m_1B_{H1}+\left(4m_2+2m_H\right)\beta_{H2}^{(0)}+\left(8m_2+2m_H\right)m_HC_{H2}+6m_Hm_1B_{H1}'$$

$$\ln\left(\frac{\gamma_H\gamma_{SO_4}}{\gamma_{HSO_4}}\right)=4f^{\gamma}+2\left(m_1-m_H\right)B_{H1}+2\left(m_2+m_H\right)\beta_{H2}^{(0)}+2m_H\left(2m_2+m_H\right)C_{H2}+4m_Hm_1B_{H1}'$$

$$f^{\gamma}=-A_{\phi}\left[\frac{\sqrt{I}}{\left(1+1.2\sqrt{I}\right)+\frac{2}{1.2}\ln\left(1+1.2\sqrt{I}\right)}\right] \qquad \textit{Equation E-4}$$

$$B_{H1}=\beta_{H1}^{(0)}+\frac{\beta_{H1}^{(1)}}{2I}\left[1-\left(1+2\sqrt{I}\right)\exp\left(-2\sqrt{I}\right)\right]$$

$$B_{H1}'=\frac{\beta_{H1}^{(1)}}{2I^2}\left[-1+\left(1+2\sqrt{I}+2I\right)\exp\left(-2\sqrt{I}\right)\right]$$

$$C_{H2}=\frac{C_{H2}^{\phi}}{2^{\frac{3}{2}}}$$

E.2 TEMPERATURE DEPENDENT CORRELATIONS

Pitzer *et al* (1997) correlated the above coefficients into absolute temperature (K) dependent correlations:

$$\ln K_2=-14.0321+\frac{2825.2}{T}$$

$$\beta_{H1}^{(0)}=0.05584+\frac{46.040}{T} \qquad \textit{Equation E-5}$$

$$\beta_{H1}^{(1)}=-0.65758+\frac{336.514}{T}$$

$$\beta_{H2}^{(0)} = -0.32806 + \frac{98.607}{T}$$

$$C_{H2}^{\phi} = 0.25333 - \frac{63.124}{T}$$

Equation E-6

E.3 DELPHI™ IMPLEMENTATION PROGRAM LISTING

The Delphi™ code, which implements this correlation in the simulation, is given in the box.

```
function PitzerCorrelation(Temp, mH, m1, m2, I : Single) : Double;
{ "Thermodynamics of Electrolytes. 7. Sulfuric Acid"
  K.S. Pitzer, R.N. Roy and L.F. Silvester
  J of American Chemical Society, 99:15 July 20 1977, p4930-4936
  correlation returns (yH+ySO4--/yHSO4)
  i.e. for rxn:        H+ + SO4-- = HSO4-
                       Krxn := exp( 14.0.3231 -2825.2/Temp(K) )
  molalities:          mH -> H+, m1 -> HSO4-, m2 -> SO4--
}

var
  Ao : Single;                      // Debye-Huckel parameter
  bH10, bH11, bH20, CH20 : Single; // initial parameters
  fy : Double;                      // function of ionic strength (I)
  BH1, BH1b, CH2 : Single;          // improved parameters

begin
  // Temperature Dependent Equations
  // note this function is only valid between 0-55°C
  Ao := 0.3770 + 4.684e-4*(Temp - 273.15) + 3.74e-6*sqr((Temp-273.15));
  bH10 := -14.0321 + 2825.2/Temp;
  bH11 := -0.65758 + 336.514/Temp;
  bH20 := -0.32806 + 98.607/Temp;
  CH20 := 0.25333 - 63.124/Temp;
  // Ionic strength dependent functions
  fy := -Ao*(sqrt(I)/(1+1.2*sqrt(I)) + (2/1.2)*ln(1+1.2*sqrt(I)) );
  BH1 := bH10 + (bH11/(2*I))*(1-(1+2*sqrt(I))*exp(-2*sqrt(I)));
  BH1b := (bH11/(2*sqr(I)))*(-1+(1+2*sqrt(I)+2*I)*exp((-2*sqrt(I))));
  CH2 := CH20/(Power(2,3/2));
  // Result := exp ln (yH+ySO4--/yHSO4)
  // This Equation from Frank Crundwell's Equil program
  Result := exp(4*fy+2*(m1 - mH)*BH1 + 2*(mH+m2)*bH20 + 2*mH*(2*m2+mH)*CH2 +
                4*m1*mH*BH1b);
end;
```


E.4 NOMENCLATURE

Symbol	Description
A_ϕ	Debye-Hückel limiting law parameter for the osmotic coefficient, equation valid between 0 and 55°C
C_{MX}^{ϕ}	corresponding third virial coefficient for triple interactions μ_{ijk} for MMX and MXX
I	ionic strength
K_2	equilibrium constant for the dissociation of the bisulphate ion

Symbol	Description
m	stoichiometric molality of sulphuric acid
m_H	molality of H^+
m_1	molality of HSO_4^-
m_2	molality of SO_4^{2-}
α	general empirical parameter (equal to 2)
$\beta_{MX}^{(0)}$, $\beta_{MX}^{(1)}$	specific parameters for the appropriate sum of λ_{ij} for binary interactions M-X, M-M and X-X where the cation M is always H and the anion X is either HSO_4^- (=1) or SO_4^{2-} (= 2)
ϕ'	osmotic coefficient
γ_H, γ_1 & γ_2	activity coefficient of H^+, HSO_4^- and SO_4^{2-}
ϑ_{12}	the difference in binary interaction λ_{ij} of HSO_4^- with SO_4^{2-}
ψ_{H12}	corresponding difference in triple interactions μ_{ijk} involving a hydrogen ion with two anions

E.5 REFERENCES

Crundwell, F.K. (1988); "The role of charge-transfer in the oxidative and non-oxidative dissolution of sphalerite", PhD Thesis, University of Witwatersrand, South Africa

Dry, M.J. (1984); "Kinetics of Leaching of a Low Grade Matte in Ferric Sulphate Solution", PhD Thesis, University of Witwatersrand, South Africa

Pitzer, K.S., R.N. Roy and L.F. Silvester (1977): "Thermodynamics of Electrolytes. 7. Sulfuric Acid", *Journal of the American Chemical Society*, **99**(15), 4930-4936

F. Criss and Cobble Correlation

F.1 CRISS COBBLE CONSTANTS

Original data from Criss and Cobble (1964a, 1964b) is presented in Table F-1. The units of a(T) and $C°_P(T)$ are that of entropy, J/mol·K and b(T) is dimensioless. There are five classes of ions exemplified as follows (Bryson, 1996):

- Hydrogen ion: H^+
- Cations: Fe^{3+}, $Fe^{2+}$and $Cu^{=}$
- Simple Anions: OH^-, Cl^- and I^-
- Oxy-anions: SO_4^{2-}, CO_3^{2-} and NO_3^-
- Acid Oxy-anions: HSO_4^- and HCO_3^-

Table F-1: Original Criss Cobble data (1964a, 1964b).

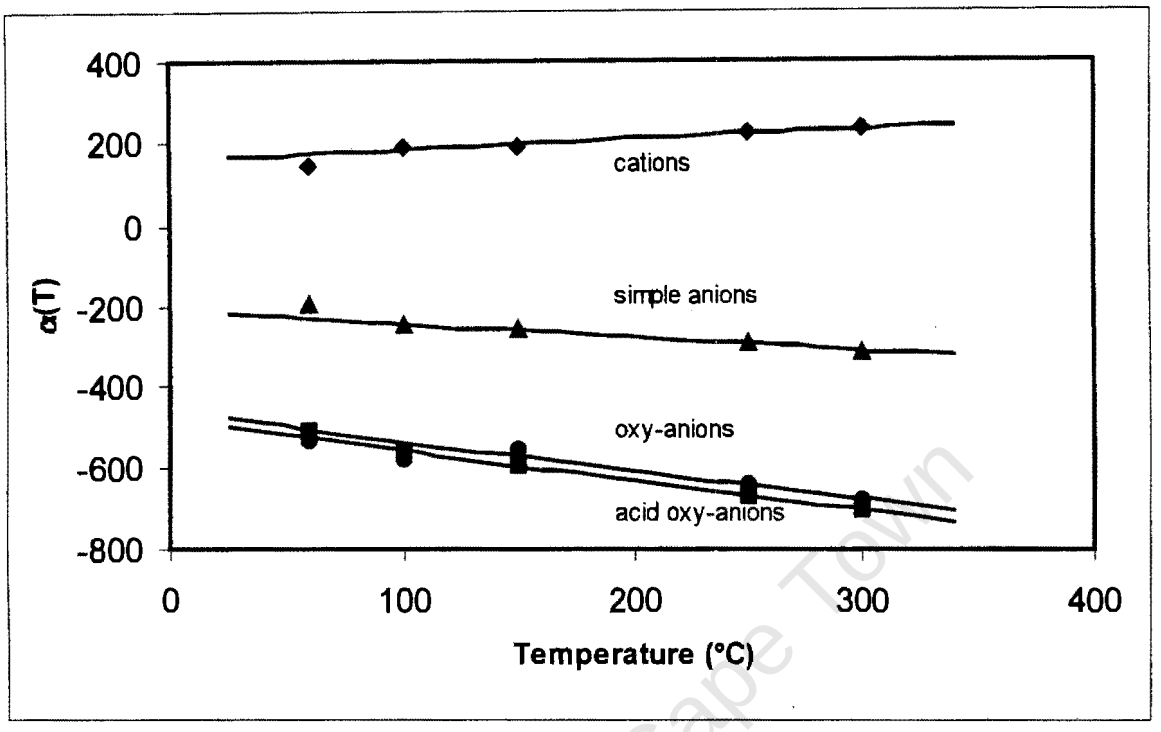
T	H+		Cations		Simple Anions		oxy-anions		acid oxy-anions	
°C	$C°_P(T)$†	$S_a(T)$	a(T)	b(T)	a(T)	b(T)	a(T)	b(T)	a(T)	b(T)
25		-5	0.0	1.00	0.0	1.00	0.0	1.00	0.0	1.00
60	96	-2.5	16.3	0.95	-21.4	0.96	-58.6	1.21	-56.5	1.38
100	130	2	43.1	0.87	-54.4	1.00	-129.8	1.47	-126.9	1.89
150	138	6.5	67.8	0.79	-89.2	0.98	-194.3	1.68	-209.3	2.38
200	155	11.1	97.6	0.71	-126.4	0.98	-280.5	2.02	-293.1	2.96
250	163	16.1	125.2	0.63	-162.0	0.97	-362.2	2.32	-376.8	3.53
300	147	20.7	153.2	0.55	-206.0	0.97	-443.8	2.61	-460.6	4.10

† Bryson, 1996

These constants, a(T), α(T), b(T) and β(T), were fitted with linear regressions (Figures F-1 and F-2), the resulting coefficients are presented in Table F-2.

Table F-2: Criss Cobble regression coefficients.

Description	*slope (m)*	*y intercept (c) (Kelvin scale)*	*Regression Coefficient (R^2)*
cations a(T)	0.5553	-165.4794	0.9994
cations b(T)	-0.0017	1.5066	0.9992
simple anions a(T)	-0.731	217.838	0.9989
simple anions b(T)	-0.00007	1.02086	0.2763
oxy-anions a(T)	-1.6084	479.3032	0.9992
oxy-anions b(T)	0.0058	-0.7284	0.9983
acid oxy-anions a(T)	-1.6736	498.7328	1.0000
acid oxy-anions b(T)	0.0112	-2.3376	0.9996
absolute entropy (J/mol·K) of H^+	0.3954	-140.439	0.9977
heat capacity (J/mol·K) of H^+	0.1729	55.242	0.9995



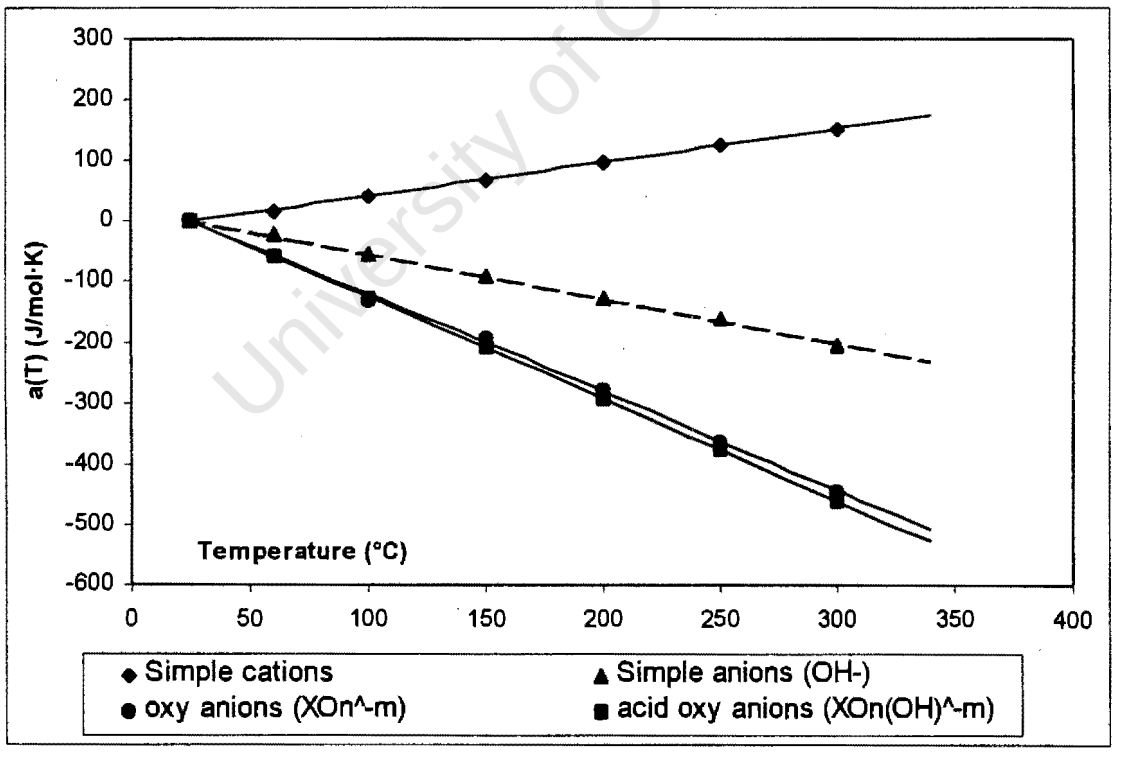
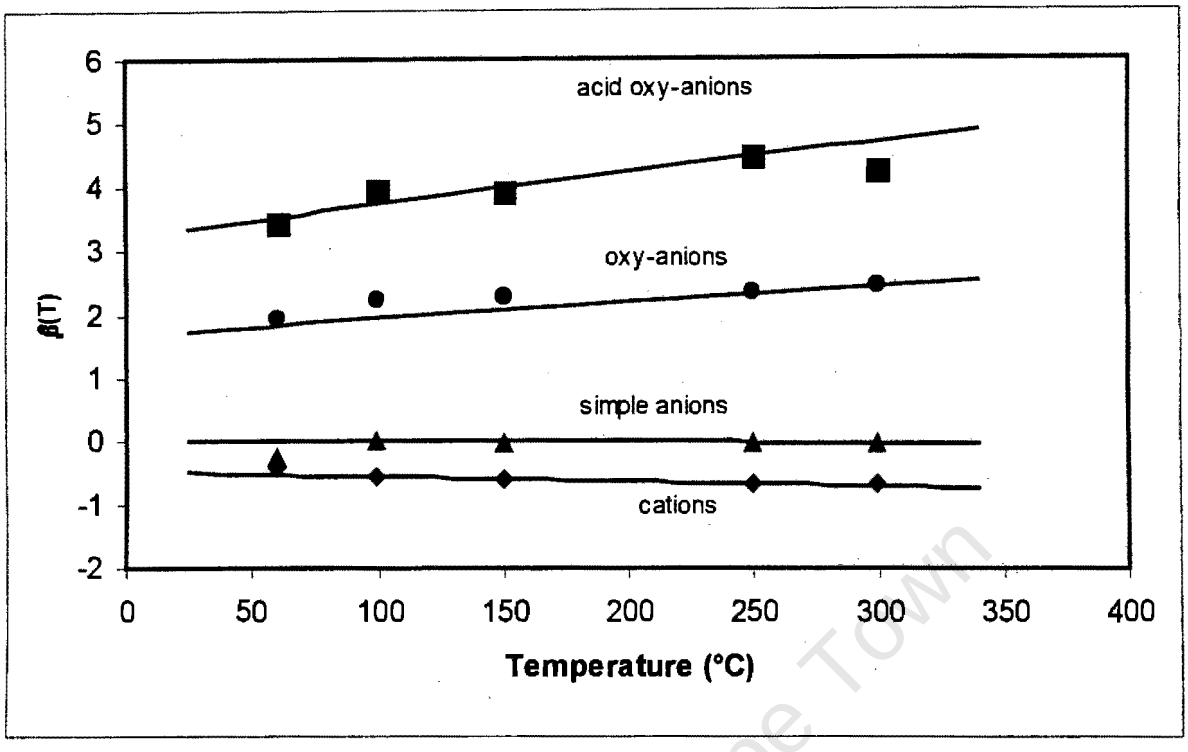


Figure F-1: Criss Cobble a(T) and α(T) constants.



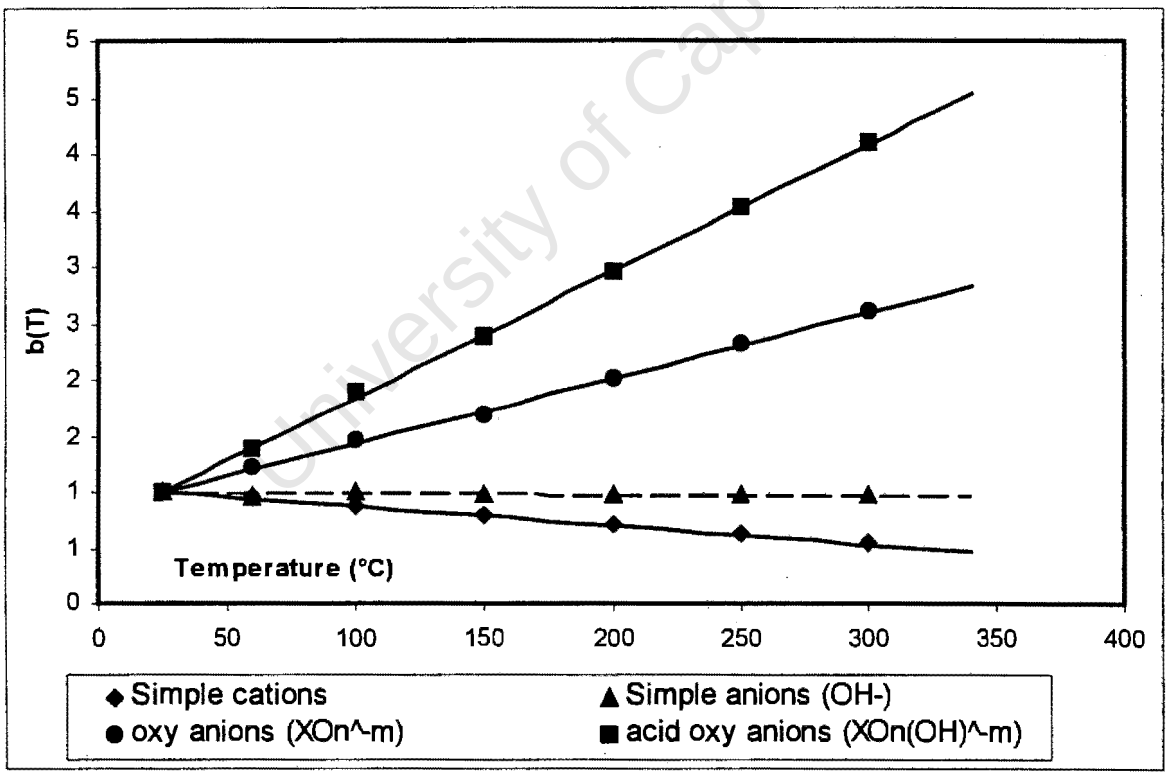


Figure F-2: Criss cobble b(T) and β(T) constants.

F.2 CRISS COBBLE SAMPLE CALCULATION

AIM: To determine log K at 100°C for the reaction: $Fe^{3+} + SO_4^{2-} \Leftrightarrow FeSO_4^+$

The value of log K at 25°C is 4.25.

CALCULATIONS:

$$S_a^o(T^o) = S^o(T^o) - 20.9z$$

$$\overline{C}_P^o(T) = \alpha(T) + \beta(T)S_a^o(T^o)$$

$$\Delta G_R^o(T) = \Delta G_R^o(T^o) - (T - T^o)\Delta S_R^o(T^o) + (T - T^o)\Delta \overline{C}_P^o(T) - T\left(\ln\frac{T}{T^o}\right)\Delta \overline{C}_P^o(T)$$

Table F-3: Details of sample calculation.

component	ΔG° kJ/mol	S°(T°) J/mol·K	z	S_a°(T°) J/mol·K	α(T)	β(T)	C°P(T) J/mol·K	C°P(T°) J/mol·K
Fe^{3+}	-4.6	-316.1	+3	-378.8	193	-0.55	401.34	24.7
SO_4^{2-}	-744.4	20.1	-2	61.9	-578	2.24	-439.34	-293.1
$FeSO_4^+$	-773.3	-129.8	+1	-150.7	193	-0.55	275.89	

Gibbs free energy of reaction (298K):	-24.3	kJ/mol
log K derived from ΔG°R(298K):	4.25	
Entropy of reaction (298K):	166.2	J/mol·K
Mean Heat Capacity of reaction (373K):	313.9	J/mol·K
Gibbs free energy of reaction (373K):	-39.5	kJ/mol
log K derived from ΔG°R(373K):	6.93	

F.3 REFERENCES

Barner, H.E. and R.V Scheuerman (1978): "Handbook of thermochemical data for compounds and aqueous species", John Wiley & Sons, New York

Bryson, A.W. (1996): "Thermodynamics of Aqueous Solutions", Continuing Engineering Education Course, University of Witwatersrand, South Africa

Criss, C.M. and J.W. Cobble (1964a): "The thermodynamic properties of high temperature aqueous solutions. IV. Entropies of the ions up to 200°C and the correspondence principle.", *J. Amer. Chem. Soc.* **86** 5390-5393

Criss, C.M. and J.W. Cobble (1964b): "The thermodynamic properties of high temperature aqueous solutions. V. The calculations of ionic heat capacities up to 200°C. Entropies and heat capacities above 200°C.", *J. Amer. Chem. Soc.* **86** 5390-5393

Zemaitis, J.F. Jr; D.M. Clark, M. Rafal and N. C Scrivner (1986): "Handbook of Aqueous Electrolyte Thermodynamics", American Institute of Chemical Engineers, New York, pp 3-43

G. Component Thermodynamic Data

G.1 THERMODYNAMIC DATA

Table G-1 contains thermodynamic data from Bard (1976) and Bryson and Nicol (1996) for Fe^{2+}, H^+, SO_4^{2-}, HSO_4^-, $FeSO_4^+$, $FeSO_4$ and Fe^{3+}.

Table G-1: Thermodynamic data of importnat components.

Formula	state	enthalpy (kJ/mol)	Gibbs (KJ/mol)	Entropy (J/mol·K)	Reference
Fe^{2+}	aq	-89.1	-78.87	-137.7	Bard (1976)
Fe^{2+}	aq	-92.5	-91.2	-107.1	Bard (1976)
Fe^{2+}	aq	-92.7	-91.8		Bard (1976)
Fe^{2+}	aq		-91.2	-107	Bard (1976)
Fe^{2+}	aq		-88.9	-107.1	Bard (1976)
Fe^{2+}	aq	-92.4	-91.5	-106	Bard (1976)
Fe^{2+}	aq	-89.2	-78.9	-137.8	Bryson and Nicol (1996)
H^+	aq	0	0	0	Bryson and Nicol (1996)
SO_4^{2-}	aq	-909.27	-744.63	20.1	Bard (1976)
SO_4^{2-}	aq	-909.9	-744.4	20.1	Bryson and Nicol (1996)
HSO_4^-	aq	-888	-756.5	131.9	Bryson and Nicol (1996)
HSO_4^-	aq	-887.34	-756.01	131.8	Bard (1976)
$FeSO_4^+$	aq	-932.4	-773.3	-129.8	Bryson and Nicol (1996)
$FeSO_4^+$	aq	-931.8	-772.8	-130	Bard (1976)
$FeSO_4$	c	-928.4	-820.9	107.5	Bard (1976)
$FeSO_4$	c	-928.8	-824.996	120.96	Bard (1976)
$FeSO_4$	aq	-998.3	-823.49	-117.6	Bard (1976)
Fe^{3+}	aq	-48.5	-4.6	-315.9	Bard (1976)
Fe^{3+}	aq	-50.2	-16.7	-280.3	Bard (1976)
Fe^{3+}	aq		-16.8		Bard (1976)
Fe^{3+}	aq	-48.6	-4.6	-316.1	Bryson and Nicol (1996)

The following thermodynamic data (Table G-2) is taken from a Russian collection (Naumov *et al.*, 1974). The heat capacity data is as follows:

$$C_P = a + bT \times 10^{-3} - cT^{-2} \times 10^{-5} \qquad \textit{Equation G-1}$$

where

C_P	heat capacity	cal/mol·K
a, b, c	coefficients	
T	temperature	K

Table G-2: Thermodynamic data from Naumov et al. (1974).

Description	state	m. weight g/mol	Hf° kcal/mol	Gf° kcal/mol	S° cal/mol.°C	Cp° cal/mol.°C	a	b	c	T range K	name
ELEMENTS											
As	c	74.92			8.51	5.9	5.23	2.22		298-1100	
As	amorph	74.92	3.24								
Cu	c	63.54			7.97	5.86	5.41	1.5		298-1357	
α-Fe	c	55.85			6.49	5.97	3.04	7.58	0.6	298-1033	
β-Fe	c	55.85					11.13			1033-1183	
γ-Fe		55.85					5.8	1.98		1183-1673	
H2	g	2.016			31.208	6.89	6.52	0.78	-0.12	298-3000	
N2	g	28.01			45.769	6.961	6.83	0.9	0.12	298-3000	
O2	g	32			49.003	7.02	7.16	1	0.4	298-3000	
S -rhombic	c	32.06			7.6	5.41	3.58	6.24		298-368.6	
S -monoclinic	c	32.06	0.08	0.03	7.78	5.65	6.2			368.6-392	
S	l	32.06					8.73			392-717.8	
Zn	c	65.37			9.95	6.07	5.35	2.4		298-692.7	
Mg	c	24.31			7.81	5.92	4.97	3.04	-0.04	298-922	
α-Ni	c	58.71			7.14	6.23	4.06	7.04		298-633	
β-Ni	c	58.71					6	1.8		633-1725	
OXIDES											
α-As2O3 - octahed.	c	197.84	-159.2	-140.6	27.89	22.86	8.37	48.6		298-548	arsenolite
β-As2O3 - octahed.	c	197.84	-158.7	-140.8	30.3						claudetite
As2O3	glassy	197.84			30.49						

Description	state	m. weight g/mol	Hf° kcal/mol	Gf° kcal/mol	S° cal/mol.°C	Cp° cal/mol.°C	a	b	c	T range K	name
As2O5	c	229.84	-221	-186.9	25.2	27.85					
CO2	glassy	44.01	-94.051	-94.255	51.06	8.57	10.57	2.1	2.06	298-2500	
CuO	c	79.54	-37.23	-30.59	10.19	10.11	11.53	1.88	1.76	298-1400	tenorite
Cu2O	c	143.08	-40.83	-35.36	22.08	14.96	14.08	5.88	0.746	298-1515	cuprite
Fe0.95O	c	69.05	-63.64	-58.59	13.74	11.5	11.66	2	0.67	298-1650	wustite
α-Fe2O3	c	159.69	-197	-177.44	20.89	24.8	23.49	18.6	3.55	298-950	hematite
SO2	g	64.06	-70.944	-71.748	59.3	9.53	11.04	1.88	1.84	298-2000	
SO3	g	80.06	-94.58	-88.69	61.34	12.11	13.9	6.1	3.22	298-1500	
ZnO	c	81.37	-33.82	-76.66	10.43	9.62	11.71	1.22	2.18	298-2000	zincite
HYDROXIDES											
Fe(OH)2	c	89.86	-137	-117.84	22.1						
Fe(OH)3	c	106.87	-201.8	-170.8	23						
α-FeO(OH)	c	88.85	-133.6	-117.2	16.1						goethite
Mg(OH)2	c	58.33	-220.97	-199.23	15.09	18.43	13.04	15.8		298-600	brasite
Zn(OH)2	amorph	99.38		-131.54							
β1-Zn(OH)2	c	99.38		-132.52							
ε-Zn(OH)2	c	99.38		-132.83							
SULPHIDES											
AsS	c	106.99	-8.7	-8.4	15.18	11.24					realgar
As2S3	c	246.04	-23	-22.8	39.1	27.8					orpiment
α-FeS	c	87.91	-23.92	-24.02	14.42		20.32			298-411	hex. troilite
β-FeS	c	87.91					17.4			411-598	

Description	state	m. weight g/mol	Hf° kcal/mol	Gf° kcal/mol	S° cal/mol.°C	Cp° cal/mol.°C	a	b	c	T range K	name
γ-FeS	c	87.91					12.2	2.38		598-1468	
Fe0.877S	c	81.04			14.53	11.92					pyrrhotite
FeS2	c	119.98	-41.6	-38.9	12.65	14.86	17.88	1.32	3.05	298-1000	pyrite
FeS2	c	119.98	-36								marcasite
Fe2S3	c	207.89	-66.9	-67.1	36.4						
H2S	g	34.08	-4.93	-8.02	49.16	8.18	7.81	2.96	0.46	298-2300	
MgS	c	56.38	-84.4								
ZnS	c	97.43	-49.8	-48.7	13.8	11.08	11.77	1.26	1.16	298-1300	sphalerite
Zn0.898Fe0.102S	c	96.46					11.8	1.2	1.04	298-1300	sphalerite
Zn0.75Fe0.25S	c	95.05					11.78	1.36	0.92	298-1200	sphalerite
Zn0.498Fe0.302S	c	92.65					11.77	1.68	0.7	298-1100	sphalerite
ZnS	c	97.43	-46.6	-46.2	16.2		11.82	1.16	1.04	1298-1250	wurtzite
Zn0.55Fe0.35S	c	94.1					11.16	2.22	0.32	298-1400	wurtzite
FeAsS	c	162.83	-25.2	-26.2	25.9		15.03	9.7	0.34	298-1100	arsenopyrite
CuFeS2	c	183.52	-42.3	-42.7	28.3		24			298-321	chalcopyrite
SULPHATES											
FeSO4	c	151.91	-221.9	-196.1	25.7	24.04					
FeSO4·H2O	c	169.92	-297.32								
FeSO4·7H2O	c	278.02	-720.48	-599.95	97.8	94.28	96.2			291-319	
Fe2(SO4)3	c	399.88	-617.1	-537	61.9		66.2			273-373	
INORGANIC SUBSTANCES IN AQUEOUS SOLUTION - IRON											
Fe(+2)	aql	55.847	-22.13	-22.05	-25	7.9		26.4			

Description	state	m. weight g/mol	Hf° kcal/mol	Gf° kcal/mol	S° cal/mol.°C	Cp° cal/mol.°C	a	b	c	T range K	name
Fe(+3)	aql	55.847	-12.13	-4.27	-66.7	5.9		19.7			
Fe(OH)(+1)	aql		-78.45	-69.54	1.1						
Fe(OH)2(0)	aqN			-109.75							
Fe(OH)3(-1)	aql			-148.48							
Fe(OH)4(-2)	aql			-185.52							
Fe(HS)2(0)	aqN			-28.5							
Fe(HS)3(-1)	aql			-28.4							
Fe(OH)(+2)	aql		-70	-58	-24.9						
Fe(OH)2(+1)	aql			-108.42							
Fe(OH)3(-0)	aqN			-161.9							
Fe(OH)4(-1)	aql			-201.7							
FeSO4(+)	aql		-223.25	-187.71	-22.7						
SULPHUR											
HSO4(-1)	aql		-212.42	-180.48	29.7	-17		-57			
SO4(-2)	aql		-217.32	-177.78	4.2	-71.6		-240			
HSO3(-1)	aql		149.67	-126.15	33.4	-16.7		-56			
SO3(-2)	aql		-151.9	-116.3	-7	-64.1		-215			
HYDROGEN											
H(+1)	aql		0	0	0	0		0			
OXYGEN											
OH(-1)	aql		-54.97	-37.564	-2.57	-35.5		-119			

G.2 NOMENCLATURE

Symbol	*Description*	*Units*
C_P	heat capacity	cal/mol·K
a, b, c	coefficients	
T	temperature	K

G.3 REFERENCES

Bard, A.J., R. Parsons and J. Jordan (1985): "Standard Potentials in Aqueous Solution", Marcel Dekker, New York

Bryson, A and M. (1996): "Thermodynamics of Aqueous Solutions", *Continuing Engineering Education Course*, University of Witwatersrand, South Africa

Naumov, G.B., B. N. Ryzhenko and I.L. Khodakovski (1974): "Handbook of Thermodynamic Data", United States Geology Survey, Menlo Park, California

H. Procedure Flow Sheet Symbols

A simple way of representing procedures is using flowcharts. Figure H-1 displays a list of flowchart symbols implemented in subsequent procedure schematics.

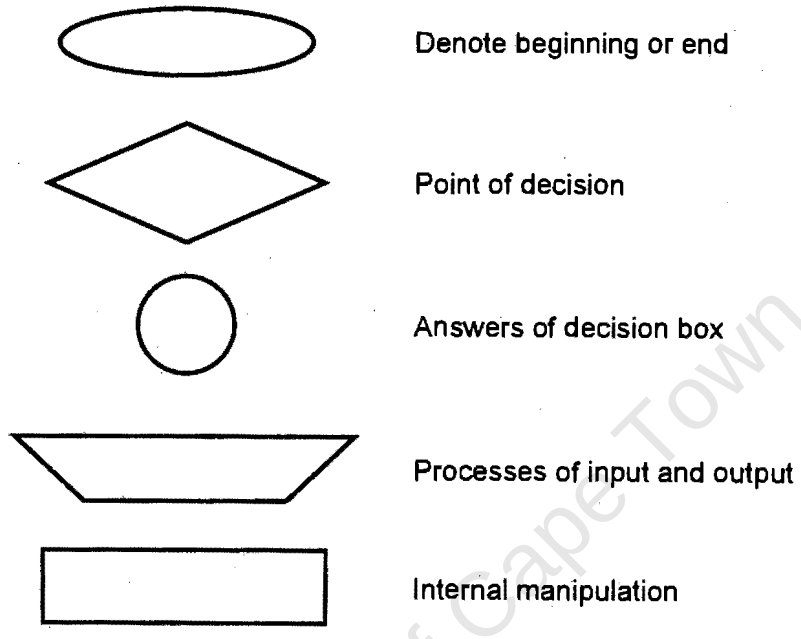


Figure H-1: Procedure flow sheet symbols

I. Crundwell's 'Equil' Program Listing

Figure I-1 describes the iteration process implemented to solve the equilibrium concentrations as dictated by a series of reactions.

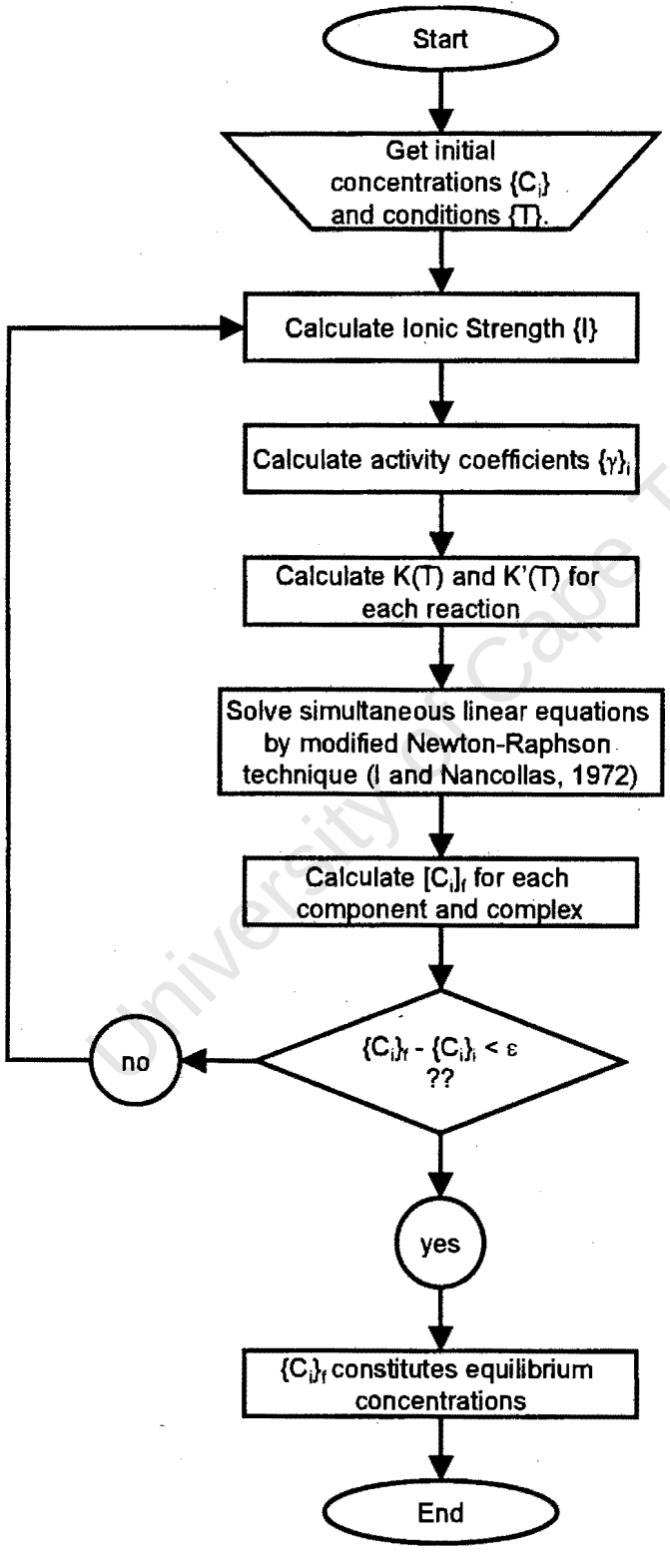


Figure I-1: Equilibrium concentration solution procedure (I and Nancollas, 1972; Dry, 1984; Crundwell, 1988).

I.1 PROGRAM LISTING

```
/* PROGRAM EQUIL                                    */

/* This program calculates the equilibrium distribution of the ferric and
   ferrous species in solution given the total concentrations of ferric
   ferrous, H+ and Zn2+. The sulphate is also input but is completely
   specified in terms of the electroneutrality of the solution.
                                                    */

/* NOMENCLATURE

   c(1)=Fe3+++
   c(2)=Fe2++
   c(3)=H+
   c(4)=SO4--
   c(5)=Zn2+
   c(6)=FeHSO4++
   c(7)=Fe(SO4)2 -
   c(8)=FeSO4+
   c(9)=HSO4-
   c(10)=FeHSO4+
   c(11)=FeOH++
   c(12)=Fe(OH)2+
   c(13)=Fe2(OH)2++++
   c(14)=FeOH+
   c(15)=Fe(OH)2
*/

#include "f:\cpp\equil\nrutil.h"

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "f:\cpp\equil\ludcmp.cpp"
#include "f:\cpp\equil\lubksb.cpp"


#define NP 20
#define MAXSTR 80

static float sqrarg, cubearg;
#define SQR(ab) ((sqrarg=(ab)) == 0.0 ? 0.0 : sqrarg*sqrarg)
#define CUBE(ab) ((cubearg=(ab)) == 0.0 ? 0.0 : cubearg*cubearg*cubearg)



/* prototypes                                       */

void lubksb (float **a, int n, int *indx, float b[]);
void ludcmp (float **a, int n, int *indx, float *d);
void equilb (float *tc, float *c, float temp);

int main (void)

{
  clrscr();
  int i;
  float temp, *tc, *c;
```

```
  char filename[MAXSTR], dummy[MAXSTR];
  FILE *fp;

  tc=vector(1,5);
  c=vector(1,15);

  /* entering the data file name                */

  //printf("enter the name of the data file:\n");
  //scanf("%s",filename);
  //printf("Filename: %s",filename,"\n");

  /* read data from file                        */

  //if ((fp = fopen(filename,"r")) == NULL)
  if ((fp = fopen("terry.dat","r")) == NULL)
     nrerror("Data file not found\n");
     fgets(dummy,MAXSTR,fp);
     fscanf(fp,"%f ",&temp);printf("temperature %f\n",temp);
     for (i=1;i<=5;i++) fscanf(fp,"%f ",&tc[i]);
  fclose(fp);

  /* calculate the equilibrium concentrations          */

  equilb(tc,c,temp);

  /* write out results to screen                */

  printf("Equilbrium concentrations\n");
            //for (i=1;i<=10;i++) printf ("%d %f ",i,c[i]);
  printf("\nFe+++ \t\t%e\n",c[1]);
  printf("Fe++ \t\t%f\n",c[2]);
  printf("H+ \t\t%f\n",c[3]);
  printf("SO4-- \t\t%f\n",c[4]);
  printf("Zn++ \t\t%f\n",c[5]);
  printf("FeHSO4++ \t%e\n",c[6]);
  printf("FeSO4+ \t\t%f\n",c[7]);
  printf("HSO4- \t\t%f\n",c[8]);
  printf("FeSO4 \t\t%f\n",c[9]);
  printf("FeHSO4+ \t%f\n",c[10]);

  free_vector(tc,1,5);
  free_vector(c,1,15);

  return 0;

}
/*=========================================================================*/

void equilb (float *tc,float *c,float temp)

/* Program to evaluate the equilibrium conditions of a solution given the
   total species concentration and the equilibrium constants. The activity
   coefficients are evaluated using the Debye-Huckel equation.
                          */
{
   int i,j,n,itr,*indx;
   float alfamn; int nerror;
   float eq[14],ceq[14],diag[5],y[10];
```

```
   float atemp,a,b,xions,p;
   float **q,*x,*g,*ga,a0[10],z[10];

   q=matrix(1,5,1,5);
   x=vector(1,5);
   g=vector(1,5);
   ga=vector(1,5);

   /* calculate equilibrium constants                    */

   eq[1]=0.0;
   eq[2]=exp(33.334-7629.9/temp);
   eq[3]=0.0;
   eq[4]=exp(-900.421+39110.926/temp+136.626*log(temp));
   eq[6]=pow(10.0,(3.339-337.37/temp));
   eq[7]=exp(30.692-7290.4/temp);
   eq[8]=pow(10.0,2.38)*exp(-754.8/temp+2.533);
   eq[9]=exp(10.9609-4727.1523/temp);
   eq[10]=exp(35.5599-15515.302/temp);
   eq[11]=exp(13.4335-5965.1007/temp);
   eq[12]=0.0;
   eq[13]=pow(10.0,(-14.01));
   eq[14]=1.6505*pow(10.0,-14.01);
   eq[9]=0.0;
   eq[10]=0.0;
   eq[11]=0.0;
   eq[12]=0.0;
   //for(i=1;i<=10;i++) printf("%d Equilb const %e\n",i,eq[i]);

   /* calculate constants for the Debye-Huckel equation        */

   atemp=temp-298.15;


a0[1]=9.;a0[2]=6.;a0[3]=9.;a0[4]=4.;a0[5]=4.;a0[6]=8.;a0[7]=8.;a0[8]=8.;a0[
9]=4.;a0[10]=4.;

z[1]=3.;z[2]=2.;z[3]=1.;z[4]=2.;z[5]=2.;z[6]=2.;z[7]=1.;z[8]=1.;z[9]=1.;z[1
0]=1.;


   a=0.4919+7.143e-4*atemp+2.113e-6*SQR(atemp)+1.172e-8*CUBE(atemp);
   b=0.3249+2.099e-4*atemp-2.582e-8*SQR(atemp)+2.589e-9*CUBE(atemp);


   /* assume total dissociation, and calculate ionic strength, and set
      guess for the initial concentrations as the total dissociation
      concentration                                    */

   tc[4]=3./2.*tc[1]+tc[2]+0.5*tc[3]+tc[5];

   xions=0.0;

   for (i=1;i<=5;i++)
      {
      c[i]=tc[i];
      xions=xions+SQR(z[i])*c[i];
      }
   xions=xions/2.;
```

```
c[9]=c[3]/2.;

/* conditions for loop                    */

itr=0;//nerror=1;

/* begin loop                          */

do {

itr += 1;

/* calculate activity coefficients from Debye-Huckel equation  */

for (i=1;i<=10;i++)
   {
   y[i]=pow(10.0,(-a*z[i]*z[i]*sqrt(xions)/(1.0+a0[i]*b*sqrt(xions))));

   }

/* calculate the equilibrium constants in terms of concentration  */

ceq[1]=eq[1]*y[1]*y[3]*SQR(y[4]);
ceq[2]=eq[2]*y[1]*y[3]*y[4]/y[6];
ceq[3]=eq[3]*y[1]*SQR(y[4])/y[7];
ceq[4]=eq[4]*y[1]*y[4]/y[8];
ceq[6]=eq[6]*y[2]*y[4];
ceq[7]=eq[7]*y[2]*y[3]*y[4]/y[10];
ceq[8]=eq[8]*y[5]*y[4];
ceq[9]=eq[9];
ceq[10]=eq[10];
ceq[11]=eq[11];
ceq[14]=eq[14];


/* calculate the concentration quotient for the bisulphate equilibrium
   using Pitzer's correlation                */

float k2,mh,m1,m2,bh10,bh11,bh20,ch20,a1,u,f,bh1,bh2,ch2,actvty;

k2=exp(14.03231-2825.2/temp);
bh10=0.05584+46.04/temp;
bh11=-0.65758+336.514/temp;
bh20=-0.32806+98.607/temp;
ch20=0.25333-63.124/temp;
a1=0.377+4.68e-4*(temp-273.15)+3.74e-6*SQR(temp-273.15);
u=sqrt(xions)/(1.0+1.2*sqrt(xions));
f=-a1*(u+(2./1.2)*log(1.0+1.2*sqrt(xions)));
bh1=bh10+bh11/xions/2.*(1.-(1.+2.*sqrt(xions))*exp(-2.*sqrt(xions)));
bh2=-1.+(1.+2.*sqrt(xions)+2.*xions)*exp(-2.*sqrt(xions));
bh2=bh11/(SQR(xions)/2.)*bh2;
ch2=ch20/(pow(2.,1.5));
mh=c[3];
m1=c[9];
m2=c[4];
actvty=4.*f+2.*(m1-mh)*bh1+2.*(m2+mh)*bh20;
actvty=exp(actvty+2.*mh*(2.*m2+mh)*ch2+4.*m1*mh*bh2);

ceq[5]=k2*actvty;
```

```
   /* calculate vector for the solution of the non-linear equations
      using the Newton-Raphson technique          */

   g[1]=c[1]+ceq[1]*c[1]*c[3]*SQR(c[4])+ceq[2]*c[1]*c[3]*c[4];
   g[1]=g[1]+ceq[3]*c[1]*SQR(c[4])+ceq[4]*c[1]*c[4]-tc[1];

   g[2]=c[2]+ceq[6]*c[2]*c[4]+ceq[7]*c[2]*c[3]*c[4]-tc[2];

   g[3]=c[3]+ceq[1]*c[1]*c[3]*SQR(c[4])+ceq[2]*c[1]*c[3]*c[4];
   g[3]=g[3]+ceq[5]*c[3]*c[4]+ceq[7]*c[2]*c[3]*c[4]-tc[3];

   g[4]=c[4]+2.0*ceq[1]*c[1]*c[3]*SQR(c[4])+ceq[2]*c[1]*c[3]*c[4];
   g[4]=g[4]+2.0*ceq[3]*c[1]*SQR(c[4])+ceq[4]*c[1]*c[4];
   g[4]=g[4]+ceq[5]*c[3]*c[4]+ceq[6]*c[2]*c[4];
   g[4]=g[4]+ceq[7]*c[2]*c[3]*c[4]+ceq[8]*c[5]*c[4]-tc[4];

   g[5]=c[5]+ceq[8]*c[5]*c[4]-tc[5];


   for (i=1;i<=5;i++)
      {
      ga[i]=g[i];
      g[i]=-g[i];
      }

   /* calculate the matrix Q                    */

   q[1][1]=1.+ceq[1]*c[3]*SQR(c[4])+ceq[2]*c[3]*c[4]+ceq[3]*SQR(c[4])+
      ceq[4]*c[4];
   q[1][2]=0.0;
   q[1][3]=ceq[1]*c[1]*SQR(c[4])+ceq[2]*c[1]*c[4];
   q[1][4]=2.*ceq[1]*c[1]*c[3]*c[4]+ceq[2]*c[1]*c[3]+2.*ceq[3]*c[1]*c[4]+
      ceq[4]*c[1];
   q[1][5]=0.0;

   q[2][1]=0.0;
   q[2][2]=1.0+ceq[6]*c[4]+ceq[7]*c[3]*c[4];
   q[2][3]=ceq[7]*c[2]*c[4];
   q[2][4]=ceq[6]*c[2]+ceq[7]*c[2]*c[3];
   q[2][5]=0.0;

   q[3][1]=ceq[1]*c[3]*SQR(c[4])+ceq[2]*c[3]*c[4];
   q[3][2]=ceq[7]*c[3]*c[4];
   q[3][3]=1.0+ceq[1]*c[1]*SQR(c[4])+ceq[2]*c[1]*c[4]+ceq[5]*c[4]+
      ceq[7]*c[2]*c[4];
   q[3][4]=2.0*ceq[1]*c[1]*c[3]*c[4]+ceq[2]*c[1]*c[3]+ceq[5]*c[3]+
      ceq[7]*c[2]*c[3];
   q[3][5]=0.0;

   q[4][1]=2.*ceq[1]*c[3]*SQR(c[4])+ceq[2]*c[3]*c[4]+2.0*ceq[3]*SQR(c[4])+
      ceq[4]*c[4];
   q[4][2]=ceq[6]*c[4]+ceq[7]*c[3]*c[4];
   q[4][3]=2.*ceq[1]*c[1]*SQR(c[4])+ceq[2]*c[1]*c[4]+ceq[5]*c[4]+
      ceq[7]*c[2]*c[4];

q[4][4]=1.0+4.*ceq[1]*c[1]*c[3]*c[4]+ceq[2]*c[1]*c[3]+4.*ceq[3]*c[1]*c[4]+
      ceq[4]*c[1]+ceq[5]*c[3]+ceq[6]*c[2]+ceq[7]*c[2]*c[3]+ceq[8]*c[5];
   q[4][5]=ceq[8]*c[4];
```

```
q[5][1]=0.0;
q[5][2]=0.0;
q[5][3]=0.0;
q[5][4]=ceq[8]*c[5];
q[5][5]=1.0+ceq[8]*c[4];

/* arrange in terms of fractional shifts          */

for (i=1;i<=5;i++)
  {
  for (j=1;j<=5;j++)
{
q[i][j]=q[i][j]*c[j];//printf("%d %d %f \n",i,j,q[i][j]);
}
  diag[i]=q[i][i];
  if (c[i]<=0.0) diag[i]=1.0;
  }

/* matrix scaling                    */

for (i=1;i<=5;i++)
   {
   g[i]=g[i]/sqrt(diag[i]);
   for (j=1;j<=5;j++)
 {
 q[i][j]=q[i][j]/sqrt(diag[i]*diag[j]);
 }
   }

/* calculate the vector X in the equation QX = G
   the method of LU decomposition is first called, and then the back-
   substitution procedure is used to calculate G

   ludcmp performs the LU decomposition of the matrix q, and then
   lubksb solves the set of linear equations QX=G where Q is input
   as the LU decomposition of Q, G is the input of the right-hand
   side vector, and output as the solution.
   There are limited error checking routines in the programs from
   Numerical Recipes                    */

n=5;
ludcmp(q,n,indx,&p);
lubksb(q,n,indx,g);

/* calculate the fractional shifts              */

for (i=1;i<=5;i++)
    {
    x[i]=g[i]/sqrt(diag[i]);
    if (x[i]>1.0) x[i]=0.95;
    if (x[i]<-1.0) x[i]=-0.95;
    }

/* calculate concentrations c[1] to c[5] for next iteration    */

alfamn=1.0; nerror=0;
for (i=1;i<=5;i++)
    {
```

```
        c[i]=c[i]*(1.+alfamn*x[i]);
        if (c[i]<0.0) c[i]=1.0e-10;
        if (fabs(alfamn*x[i])>0.00001) nerror=nerror+1;
        }

   /* calculate concentrations c[6] thru c[10] for next iteration */

   c[6]=ceq[2]*c[1]*c[3]*c[4]; //FeHSO4++
   c[7]=ceq[3]*c[1]*SQR(c[4]); //Fe(SO4)-
   c[8]=ceq[4]*c[1]*c[4];      //FeSO4+
   c[9]=ceq[5]*c[3]*c[4];      //HSO4-
   c[10]=ceq[7]*c[2]*c[3]*c[4];//FeHSO4+

   //for(i=1;i<=10;i++) printf("%d c[i] %e ceq[i] %e\n",i,c[i],ceq[i]);
   //printf("mass Ferric total in %e Mass Ferric total out
%e\n",tc[1],(c[1]+c[6]+c[8]));

   /* calculate the ionic strength                */

   xions=0.0;
   for (i=1;i<=10;i++)
       {
       xions=xions+SQR(z[i])*c[i];
       }
   xions=xions/2.;

   } while (nerror>0); // end of do while loop

   /* calculate concentrations for values of interest      */

   c[6]=ceq[2]*c[1]*c[3]*c[4]; //FeHSO4++
   c[7]=ceq[4]*c[1]*c[4];      //FeSO4+
   c[8]=ceq[5]*c[3]*c[4];      //HSO4-
   c[9]=ceq[6]*c[2]*c[4];      //FeSO4
   c[10]=ceq[7]*c[2]*c[3]*c[4];//FeHSO4+

  free_vector(x,1,5);
  free_vector(g,1,5);
  free_vector(ga,1,5);
  free_matrix(q,1,5,1,5); printf("number of iterations %d\n",itr);

}    // end of routine equilb

/*=========================================================================*/
```


I.2 REFERENCES

Crundwell, F. (1991): " The role of charge-transfer mechanisms in the oxidative and non-oxidative dissolutions of sphalerite" , PhD thesis, University of Witwatersrand, Johannesburg

Dry, M.J. (1984): "Kinetics of leaching of a low grade matte in ferric sulphate solution", PhD thesis, University of Witwatersrand, Johannesburg

I, T and G.H. Nancollas (1972): "EQUIL – A general computational method for the calculation of solution equilibria", *Analytical Chemistry*, **44**(12), 1940-1950

J. Database Normalisation

J.1 GUIDELINES GOVERNING NORMALISATION

Five *normal forms* are commonly recognized, together with a number of special variations. The first three of these are of general interest; the others are much more esoteric and will not be discussed (taken from Ehrmann, 1995).

1. The first rule of database design says: *Eliminate repeating groups.* For each group of related fields, make a separate table and give that table a Primary key. A table is said to be in first normal form if all fields contain atomic values only.
2. The second rule of database design reduces redundancy: *If a field in a table is related to only part of a multi-field key, remove it to a separate table.* In a table where the primary key consists of more than one field, every other field in the table must be an attribute of the complete primary key. No other fields in this table must describe the combination of these fields, since it is this combination which determines uniqueness. A table is said to be in second normal form if it is already in first normal form and if every non-key field is functionally dependent on the complete Primary key.
3. Third normal form is similar to second normal form in that it is designed to reduce update complications. It specifically addresses relationships in tables that have only one key field: *If fields do not contribute to a description of the table's key, they should be removed to a separate table.* A table is in third normal form if each record consists of a primary key that identifies an entity and a set of zero or more mutually independent fields that further describe that primary key.

J.2 REFERENCES

Ehrmann, D. (1995): "Paradox Queries - A Complete Reference", M&T Books, New York

K. Species Database SQL Implementation

The 'Species' database consists of several tables *e.g.* Comp_Thermo, linked by ID fields, *e.g.* Comp_No. The database is constructed using SQL code as listed below.

```
/* Extract Database \work\SQLDataBase\Species.gdb */
CREATE DATABASE "\work\SQLDataBase\Species.gdb" PAGE_SIZE 1024
;

/* Domain definitions */
CREATE DOMAIN DCLASS_NO AS INTEGER
         CHECK (VALUE>1000);
CREATE DOMAIN DFORMULA AS VARCHAR(20);
CREATE DOMAIN DNAME AS VARCHAR(50);
CREATE DOMAIN DDESCR AS VARCHAR(100);
CREATE DOMAIN DSINGLE5 AS NUMERIC(15, 5);
CREATE DOMAIN DSSINT AS SMALLINT
         DEFAULT 0

         CHECK (VALUE BETWEEN -15 AND 15);
CREATE DOMAIN DSSPINT AS SMALLINT
         DEFAULT 0

         CHECK (VALUE BETWEEN -1 AND 30);
CREATE DOMAIN DSINT AS SMALLINT
         CHECK (VALUE BETWEEN -50 AND 50);
CREATE DOMAIN DCHAR_CODE AS CHAR(10);

/* Table: COMPGEN, Owner: KAREN */
CREATE TABLE COMPGEN (COMPNO DCLASS_NO NOT NULL,
        CFORM DFORMULA NOT NULL,
        CNAME DNAME NOT NULL,
        CHARGE DSSINT NOT NULL,
        MWEIGHT DSINGLE5 NOT NULL,
        IONSIZE DSSPINT,
        ST_CODE CHAR(10),
        CRISS_CODE DCHAR_CODE,
UNIQUE (CFORM),
UNIQUE (CNAME),
CONSTRAINT PCOMPNO PRIMARY KEY (COMPNO));

/* Table: COMP_THERMO, Owner: KAREN */
CREATE TABLE COMP_THERMO (COMPNO DCLASS_NO NOT NULL,
        GIBBS_F FLOAT,
        ENTHALPY_F FLOAT,
        ENTROPY FLOAT,
        HEATCAP FLOAT,
        REFCODE CHAR(10) NOT NULL);

/* Table: CP_DATA, Owner: KAREN */
CREATE TABLE CP_DATA (COMPNO DCLASS_NO NOT NULL,
        COEFF_A DSINGLE5  DEFAULT 0
,
        COEFF_B DSINGLE5  DEFAULT 0
,
```

```
        COEFF_C DSINGLE5  DEFAULT 0
,
        COEFF_D DSINGLE5  DEFAULT 0
,
        COEFF_E DSINGLE5  DEFAULT 0
,
        REFCODE CHAR(10) NOT NULL,
        ST_CODE DCHAR_CODE);

/* Table: CRISS_COBBLE, Owner: KAREN */
CREATE TABLE CRISS_COBBLE (ATM DSINGLE5,
        ATC DSINGLE5,
        BTM DSINGLE5,
        BTC DSINGLE5,
        HSATM DSINGLE5,
        HSATC DSINGLE5,
        REF_CODE CHAR(10) NOT NULL,
        CRISS_CODE DCHAR_CODE NOT NULL);

/* Table: CRISS_DESCR, Owner: KAREN */
CREATE TABLE CRISS_DESCR (DESCRIPTION DDESCR,
        CRISS_CODE DCHAR_CODE NOT NULL);

/* Table: REF_DESCR, Owner: KAREN */
CREATE TABLE REF_DESCR (REF_CODE CHAR(10) NOT NULL,
        REF_DESCRIPTION DDESCR NOT NULL,
CONSTRAINT PREFCODE PRIMARY KEY (REF_CODE));

/* Table: RXN_DESCR, Owner: KAREN */
CREATE TABLE RXN_DESCR (RXNNO SMALLINT NOT NULL,
        DESCR DDESCR NOT NULL,
        ENTRIES DSSPINT NOT NULL,
CONSTRAINT PRXN_DESCR_ID PRIMARY KEY (RXNNO));

/* Table: RXN_STOICH, Owner: KAREN */
CREATE TABLE RXN_STOICH (RXNNO SMALLINT NOT NULL,
        COMPNO DCLASS_NO NOT NULL,
        STOICH_COEFF DSINT NOT NULL);

/* Table: STATE_DESCR, Owner: KAREN */
CREATE TABLE STATE_DESCR (ST_CODE CHAR(10) NOT NULL,
        ST_DESCR VARCHAR(100) NOT NULL,
UNIQUE (ST_CODE));

/*  Index definitions for all user tables */
CREATE UNIQUE INDEX FORMINDEX ON COMPGEN(CFORM, CNAME);
CREATE INDEX CTHERMX ON COMP_THERMO(COMPNO, REFCODE);
CREATE UNIQUE INDEX CP_DATAX ON CP_DATA(COMPNO, ST_CODE);
CREATE UNIQUE INDEX CRISSCOBBLEX ON CRISS_COBBLE(CRISS_CODE);
CREATE INDEX CRISSDESCRX ON CRISS_DESCR(CRISS_CODE);
CREATE UNIQUE INDEX STOICHX ON RXN_STOICH(RXNNO, COMPNO);

CREATE GENERATOR COMPNO_GEN;
CREATE GENERATOR RXNNO_GEN;

ALTER TABLE RXN_DESCR ADD
        CHECK ( RXNNO  > 0)
;
```

```
ALTER TABLE RXN_STOICH ADD
        CHECK (RXNNO >0)
;
SET TERM ^ ;

/* Triggers only will work for SQL triggers */
CREATE TRIGGER SET_COMPNO FOR COMPGEN
ACTIVE BEFORE INSERT POSITION 0
AS

BEGIN

  NEW.COMPNO = GEN_ID(COMPNO_GEN,1);

END
 ^
CREATE TRIGGER SET_RXNNO FOR RXN_DESCR
ACTIVE BEFORE INSERT POSITION 0
AS

BEGIN

  NEW.RXNNO = GEN_ID(RXNNO_GEN, 1);

END
 ^
COMMIT WORK ^
SET TERM ; ^

/* Grant permissions for this database */
```

L. Running Database SQL Implementation

The 'Running' database acts as temporary storage for calculation variables. As such the thermodynamic database and working variables (*e.g.* Std_Gibbs_Rxn) remain separate from the component data. The database is constructed using SQL code as listed below.

```
/* Extract Database \Work\SQLDataBase\Running.gdb */
CREATE DATABASE "\Work\SQLDataBase\Running.gdb" PAGE_SIZE 1024
;

/* Domain definitions */
CREATE DOMAIN DCLASS_NO AS INTEGER
         check (value>1000);
CREATE DOMAIN DSINGLE5 AS NUMERIC(15, 5);
CREATE DOMAIN DFORMULA AS VARCHAR(20);
CREATE DOMAIN DSINGLE AS NUMERIC(9, 5);
CREATE DOMAIN DSINGLE5P AS NUMERIC(15, 5)
          default 999

          CHECK (VALUE > 0);
CREATE DOMAIN DSINGLEPL1 AS NUMERIC(9, 7)
          DEFAULT 1

          CHECK (VALUE BETWEEN 0 AND 1.0001);
CREATE DOMAIN DCHAR_CODE AS CHAR(10);
CREATE DOMAIN BOOLEAN AS SMALLINT
          default 0

          check (value in (0, 1));

/* Table: CALCOUTPUT, Owner: KAREN */
CREATE TABLE CALCOUTPUT (RXNNO SMALLINT NOT NULL,
         TEMPK DSINGLE5 NOT NULL,
         COMPNO DCLASS_NO,
         ALPHAT DSINGLE5,
         BETAT DSINGLE5,
         CPMEANT DSINGLE5);

/* Table: COMPCONC, Owner: KAREN */
CREATE TABLE COMPCONC (CCONC DSINGLE5P NOT NULL,
         CACTCOEFF DSINGLEPL1,
         CACT DSINGLE5P,
         CFORM DFORMULA NOT NULL,
         CRNT_CALC BOOLEAN,
         COMP_CATGRY BOOLEAN,
CONSTRAINT PCFORM PRIMARY KEY (CFORM));

/* Table: MEAN_CP, Owner: KAREN */
CREATE TABLE MEAN_CP (COMPNO DCLASS_NO NOT NULL,
         TFINAL DSINGLE NOT NULL,
         CALC_VALUE DSINGLE5 NOT NULL,
         METHOD_CODE DCHAR_CODE NOT NULL,
         CALC_DATE DATE NOT NULL);

/* Table: RXN_INFO, Owner: KAREN */
```

```
CREATE TABLE RXN_INFO (RXNNO SMALLINT NOT NULL,
        STD_GIBBS_RXN FLOAT,
        STD_ENTROPY_RXN FLOAT,
        STD_K_RXN FLOAT,
CONSTRAINT PRXN_INFO_ID PRIMARY KEY (RXNNO));

/*  Index definitions for all user tables */
CREATE UNIQUE INDEX MEAN_CP_INDEX ON MEAN_CP(COMPNO, TFINAL);


/* Grant permissions for this database */
```

M. Object Oriented Programming

Superficially the term "object-oriented" means software is organised as a collection of discrete objects that incorporates both data structure and behaviour. This is in contrast to conventional programming in which data structure and behaviour are loosely connected. There is some dispute about exactly what characteristic areas required by an object-oriented approach, but they generally include four aspects: identity, classification, polymorphism and inheritance (Rumbaugh *et al.*,1991).

M.1 CHARACTERISTICS OF OBJECTS

Identity means that data is quantised into discrete, distinguishable entities termed objects. A paragraph in a document is an example of an object. Objects can be concrete or conceptual. Each object has its own inherent identity, hence objects are distinct even if all their attribute values are identical. Each object has a handle by which it can be uniquely referenced. Object references are uniform and independent of the contents of the objects, permitting mixed collections of objects to be created (Rumbaugh *et al.*,1991).

Classification means that objects with the same data structure (attributes) and behaviour (operations) are grouped into a class. A *class* is an abstraction that describes properties important to an application and ignores the rest. Any choice of classes is arbitrary and dependent upon the application (Rumbaugh *et al.*,1991).

Each class describes a possible infinite set of individual objects. Each object is said to be an *instance* of its class. Each instance of a class has its own value for each attribute, but it shares the same attribute names and operations with other instances of the class. Figure M-1 shows a class and some instances of the class. An object contains an implicit reference to its own class; it knows what kind of thing it is (van der Merwe, 1995).

Polymorphism means that the same operation may behave differently on different classes. The move operation (Figure M-1) for example may behave differently on various different classes. An *operation* is an action or transformation that an object performs or is subject to. A specific implementation of an operation by a certain class is called a *method*. Because an object-oriented operator is polymorphic, it may have more than one method implementing it (Rumbaugh *et al.*, 1991).

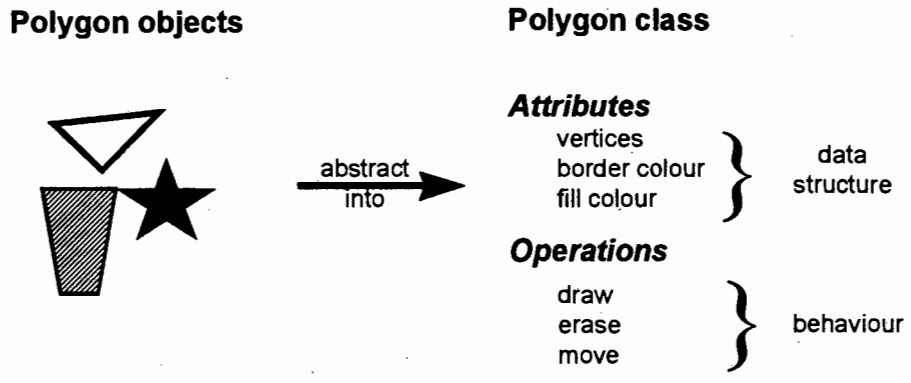


Figure M-1 Objects and classes (Rumbaugh et al., *1991)*

Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. A class can be defined broadly and then refined into successively finer *subclasses*. Each subclass incorporates, or *inherits* all of the properties of its *superclass* and adds its own unique properties. The properties of the superclass need not be repeated in each subclass. The ability to factor out common properties of several classes into a common superclass and to inherit the properties from the superclass can greatly reduce repetition within designs and programs and is one of the main advantages of an object-oriented system (Rumbaugh *et al.*,1991)..

M.2 OBJECT-ORIENTED THEMES

There are several themes underlying object-oriented technology. Although these themes are not unique to object-oriented systems, they are particularly well supported in object-oriented systems.

M.2.1 ABSTRACTION

Abstraction consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental properties. In system development, this means focusing on what an object is and does, before deciding how it should be implemented. Use of abstraction preserves the freedom to make decisions as long as possible by avoiding premature commitments to details (van der Merwe, 1995).

M.2.2 ENCAPSULATION

Encapsulation (also known as *information hiding*) consists of separating the external aspects of an object, which are accessible to other objects from the internal implementation details of the object which are hidden from other objects. Encapsulation prevents a program from becoming so interdependent that a small change has massive ripple effects. The implementation of an object can be changed without affecting the applications that use it (van der Merwe, 1995).

M.3 REFERENCES

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991): "Object-oriented modelling and design", Prentice Hall, United States of America

van der Merwe, J.J.N (1994): "Introduction to object-oriented programming", Course, Department of Electrical Engineering, University of Stellenbosch, South Africa

N. Program Listing

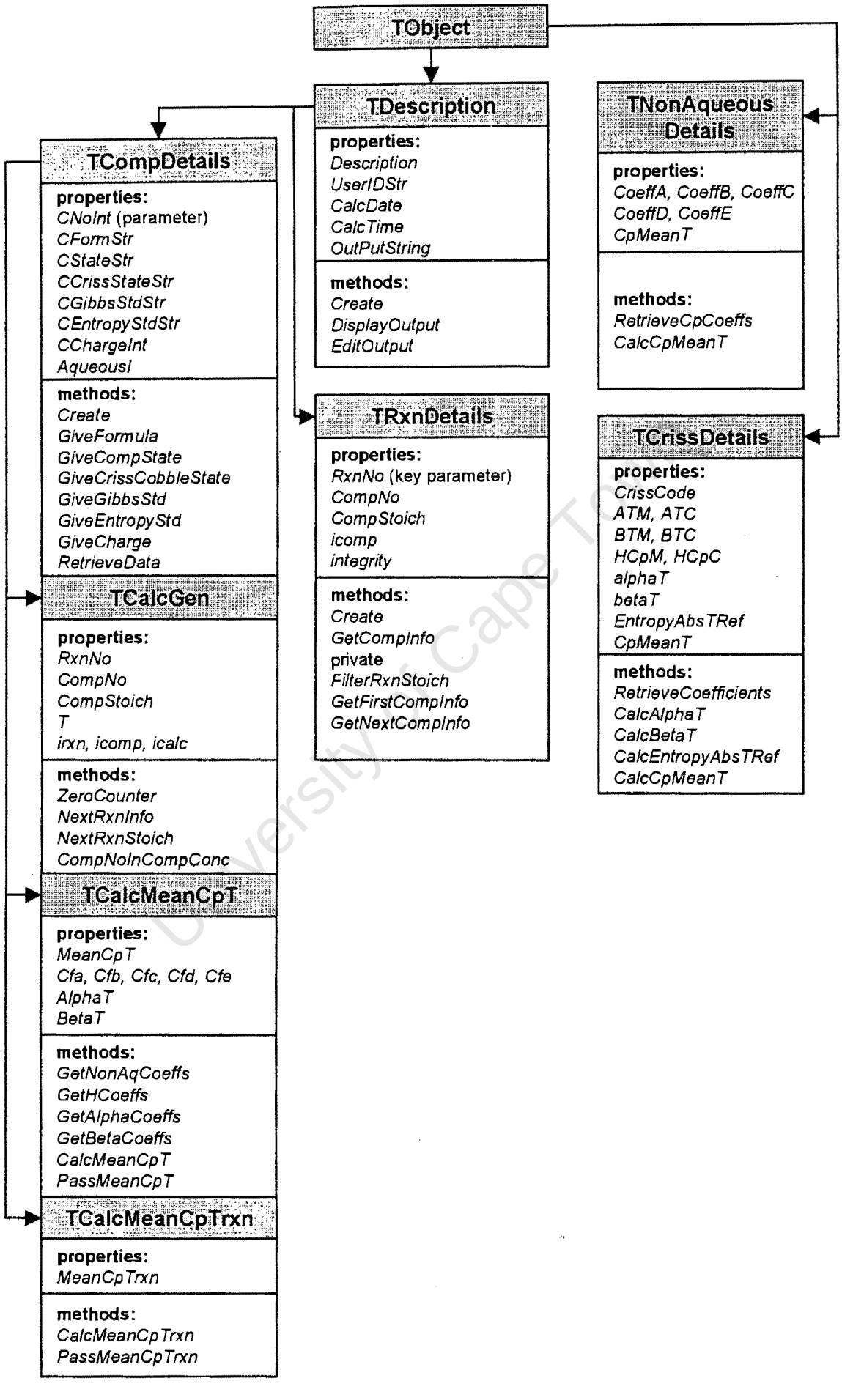
The name SimBioSys is derived from 'Simulation of the Bioleaching System'. It is indicative of the primary objective of the simulation. As such the program has been coded to ensure that other modules can be added to the existing framework with minimal effort. As a result the program consists of several units (Table N-1).

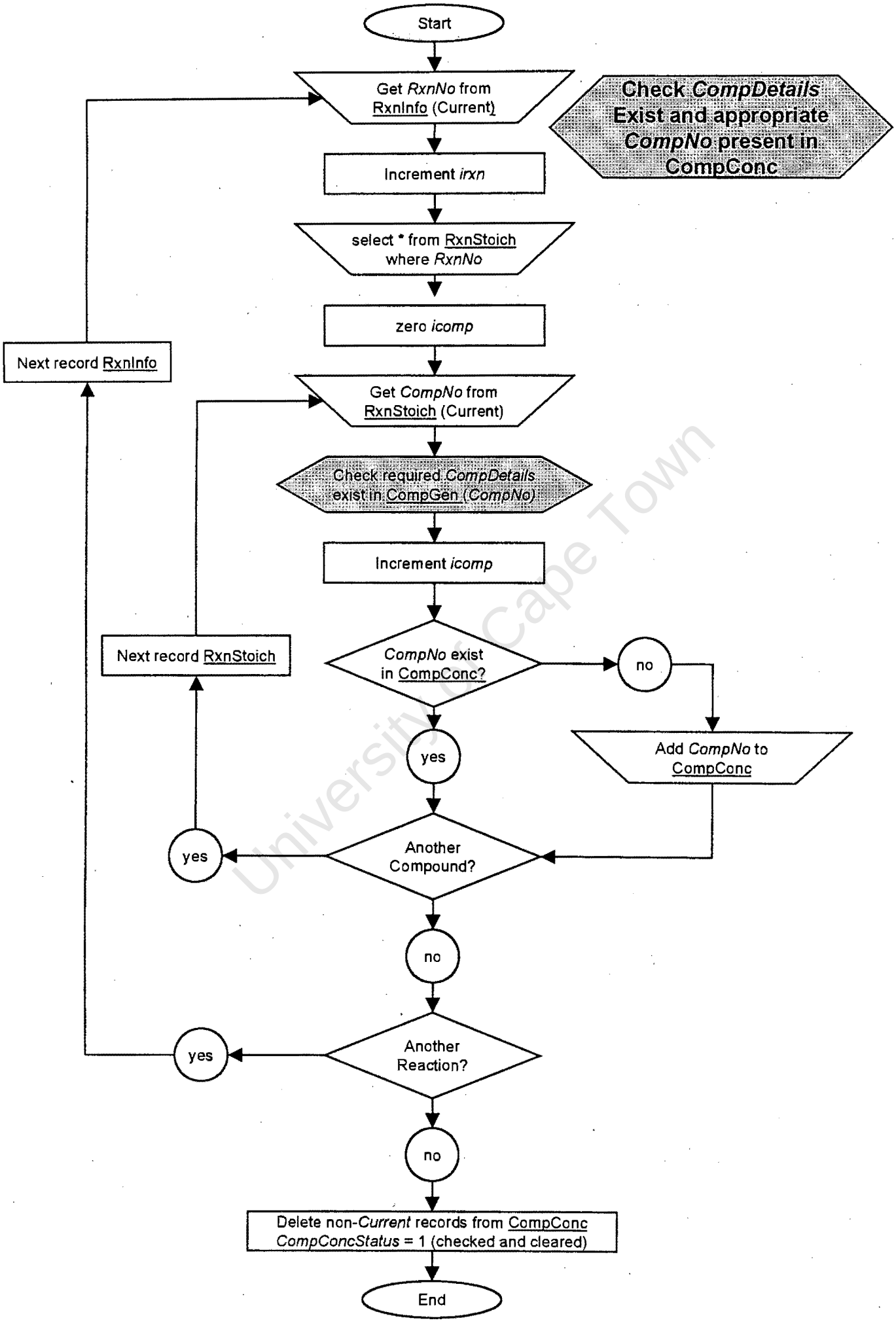
Table N-1: SimBioSys unit description.

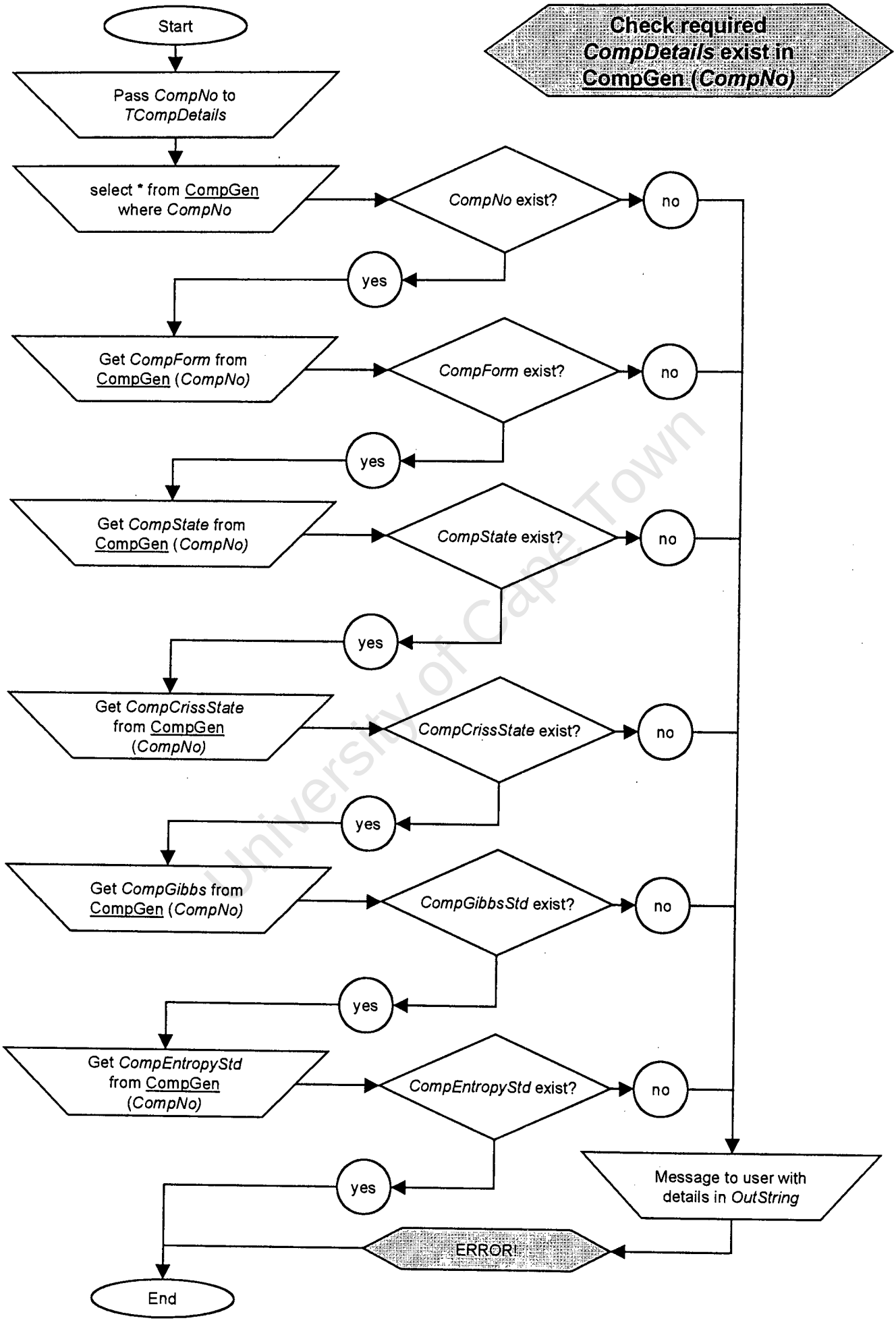
Unit	*Description*
SimBioSys	project file – contains information about forms and component units
CompClassUnit	Unit detailing the classes relating to the compounds, including data retrieval from the species database, Criss Cobble correlation, heat capacity data for aqueous and non-aqueous compounds.
CompGenDataFormUnit	Unit controlling access to the SQL species database, including updating or deleting compound information.
CompGenDMUnit	Unit defining the species database data module, including fundamental lookup operations.
Constants	Unit enabling easy constant definition.
DisplayRunningDataFormUnit	Unit controlling the running data display mechanism.
DisplaySpeciesDataFormUnit	Unit controlling the species data display mechanism.
Exceptns	Unit defining the classes for exception handling.
GibbsCalcClassUnit	TGibbsCalcClass definitions
GibbsCalcFormUnit	TGibbsCalcClass form – including reaction selection and output to memo and chart.
GibbsCalcProgressUnit	Progress bar to display calculation status to user.
LogInOutFormUnit	Unit to initialise application and open database tables and queries.
MainFormUnit	Main menu system
RxnClassUnit	TRxnDetails definitions
SelectionFormUnit	Form to assist reaction and compound selection for working with a system of reactions
UnitsCheckUnit	Unit to facilitate unit error handling – *i.e.* to check whether values are in an appropriate range.
UserClassUnit	TUser definitions

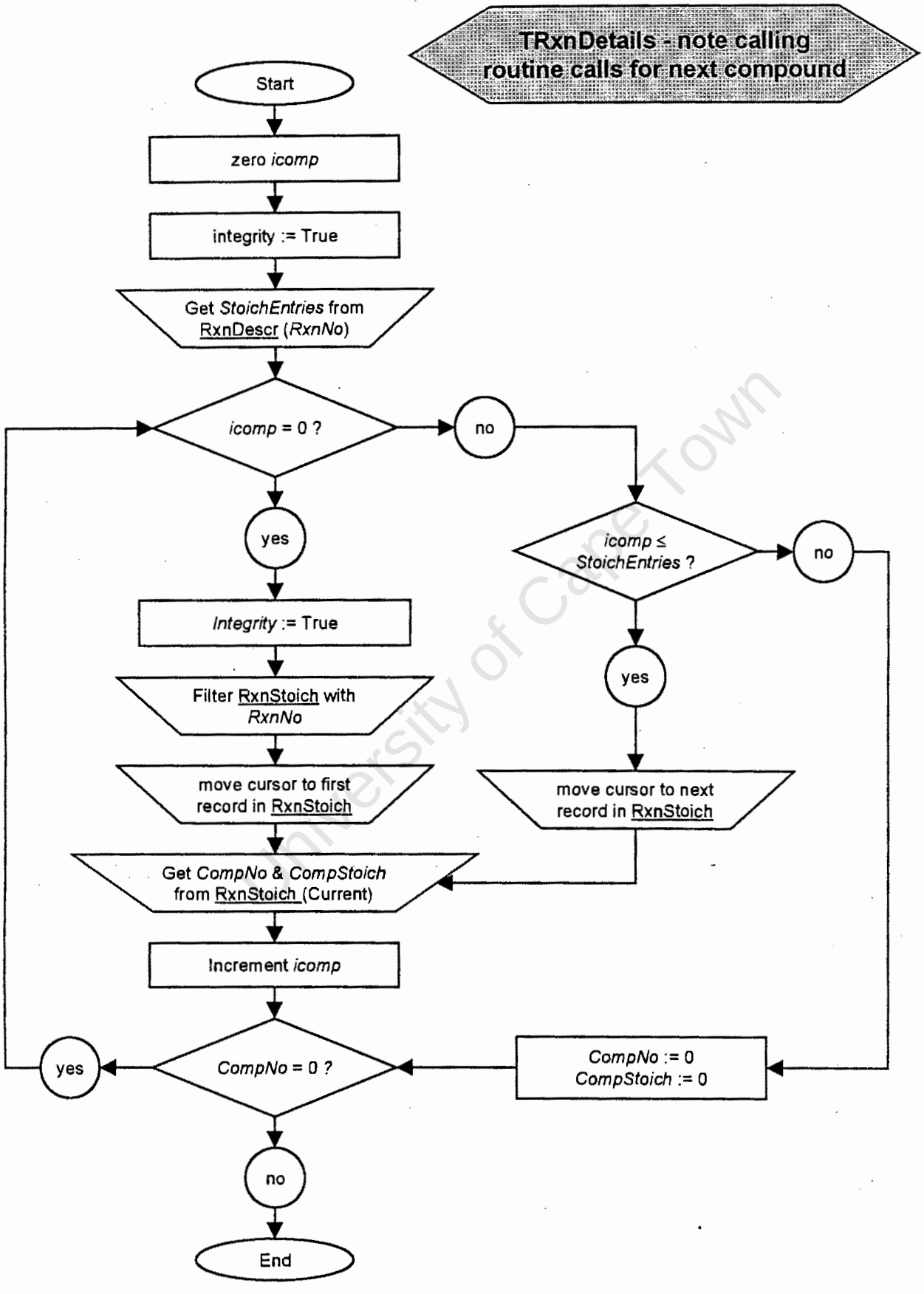
The remainder of this appendix contains:

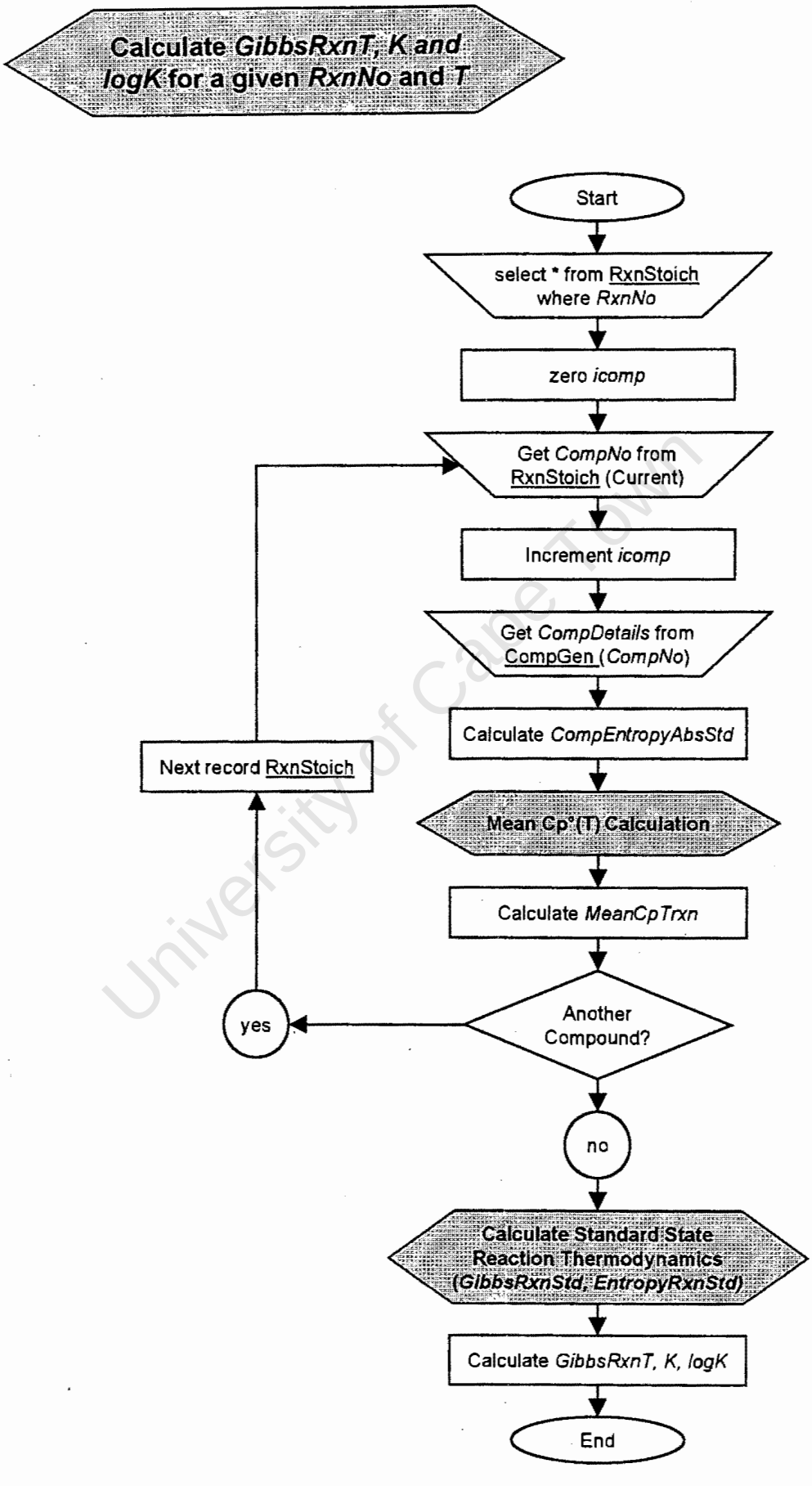
- a class diagram for *TCrissDetails, TNonAqueousDetails, TDescription, TCompDetails, TCalcGen, TCalcMeanCpT, TCalcMeanCpTRxn* and *TRxnDetails*
- Flowcharts describing:
  - The retrieval of compound details, an exception is raised if the necessary data does not exist.
  - Tracing through a set of reactions using *TRxnDetails*
  - Calculate *GibbsRxnT, K* and *logK* for a given *RxnNo* and *T*
  - Mean $Cp°(T)_{rxn}$ calculation for aqueous and non-aqueous compounds
  - Calculate standard state reaction thermodynamics
- Application screenshots and Delphi® 2 .0 pascal listing

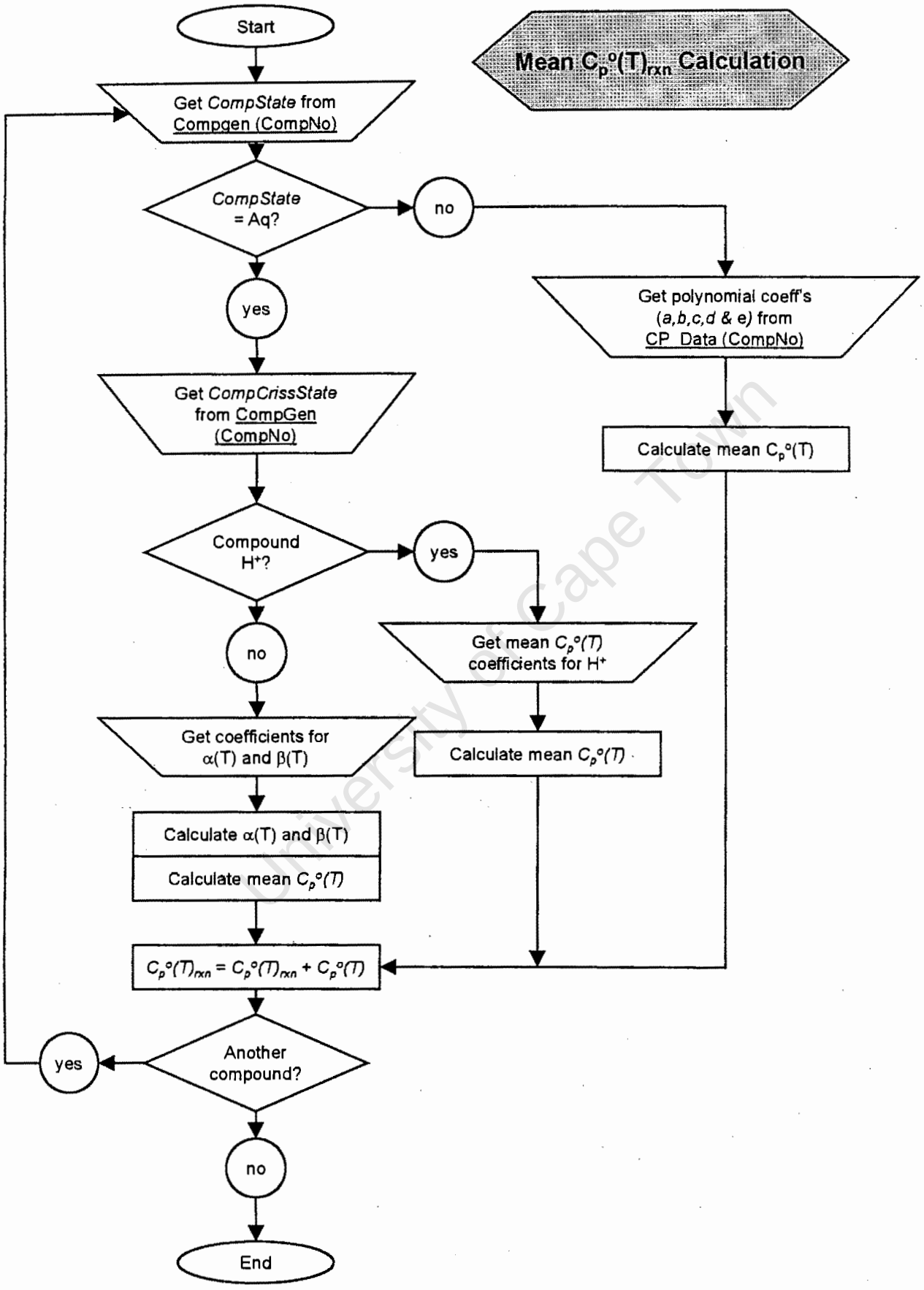


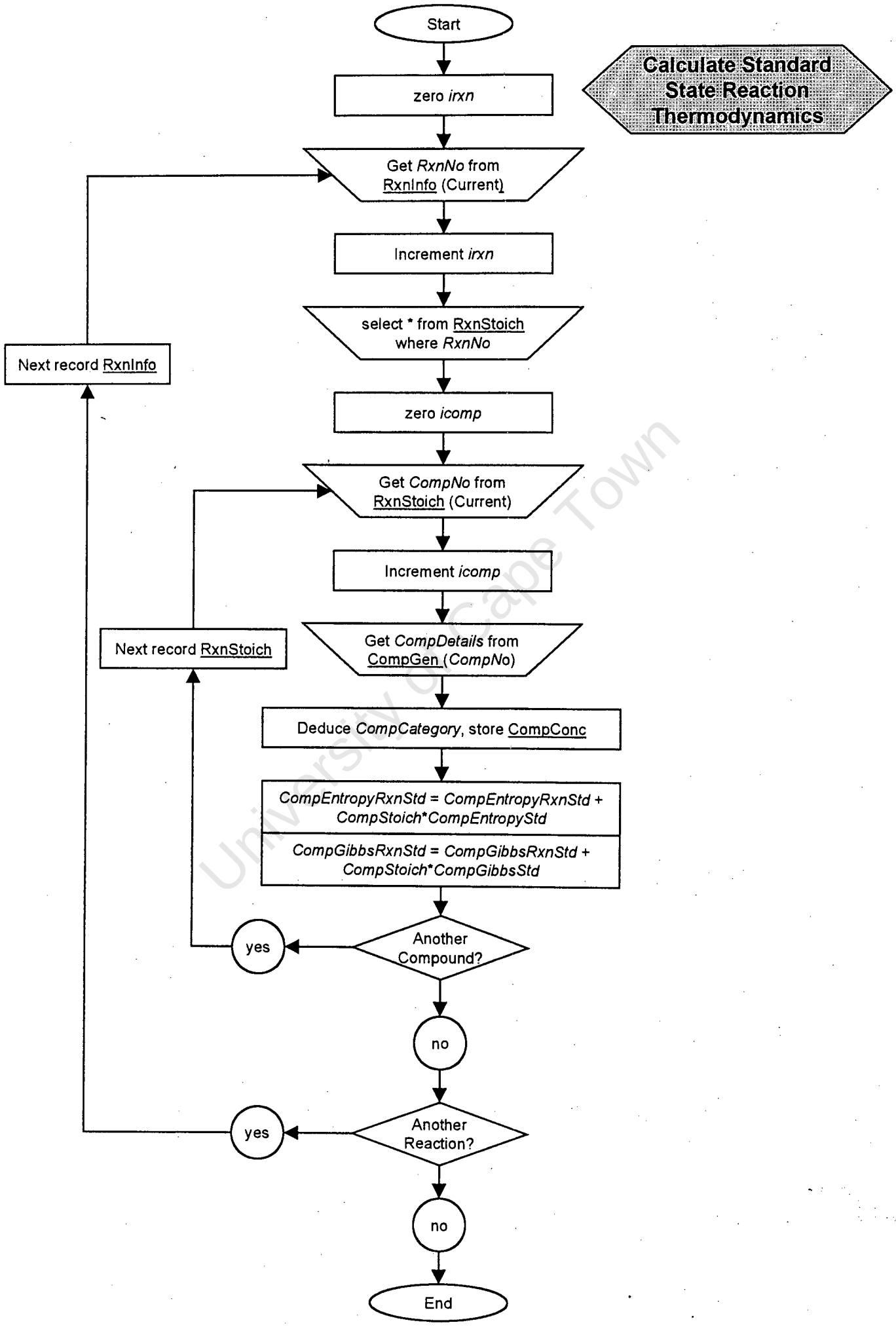


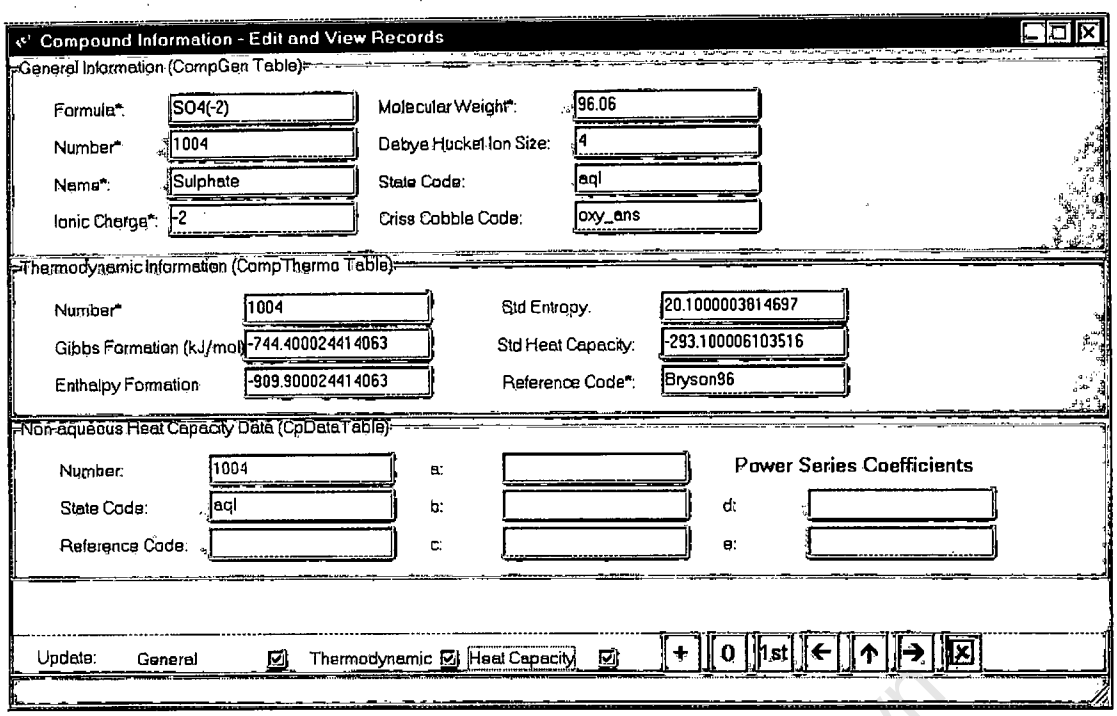


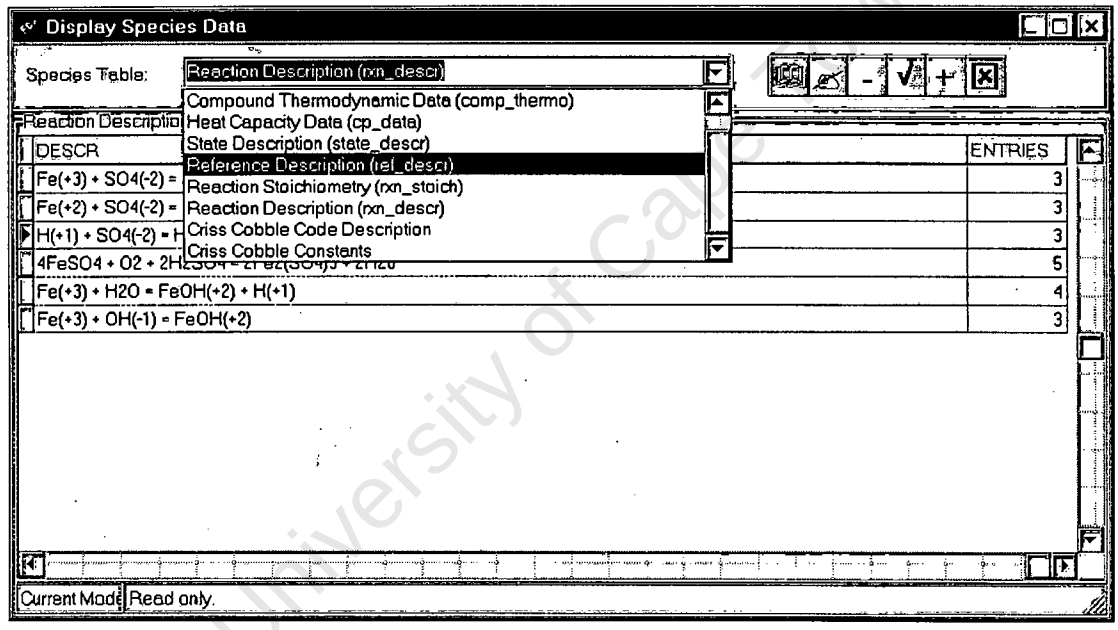


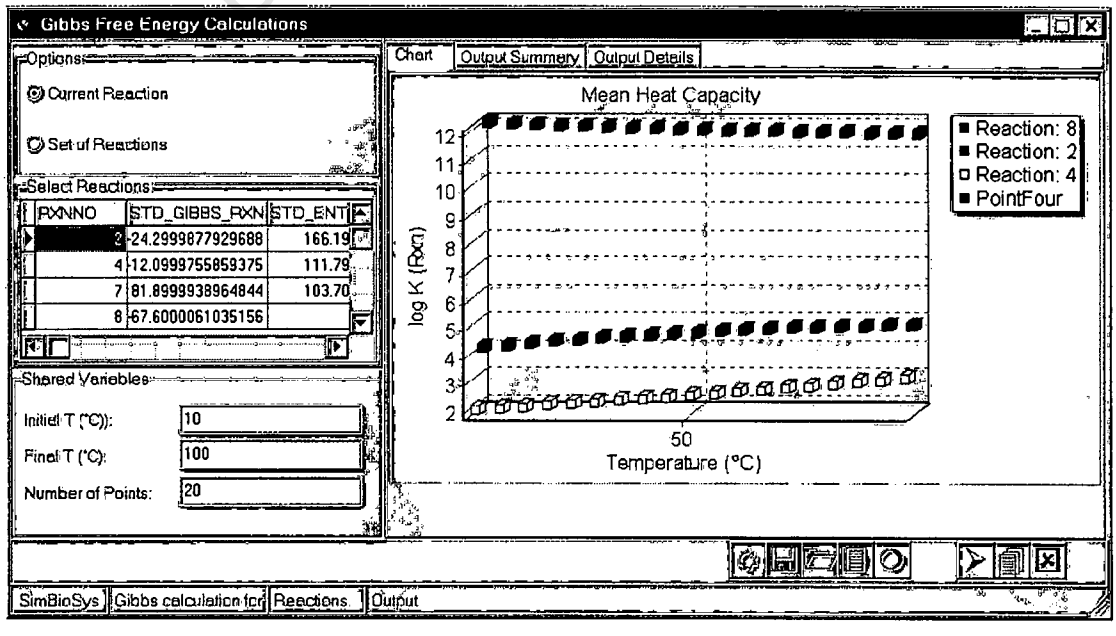


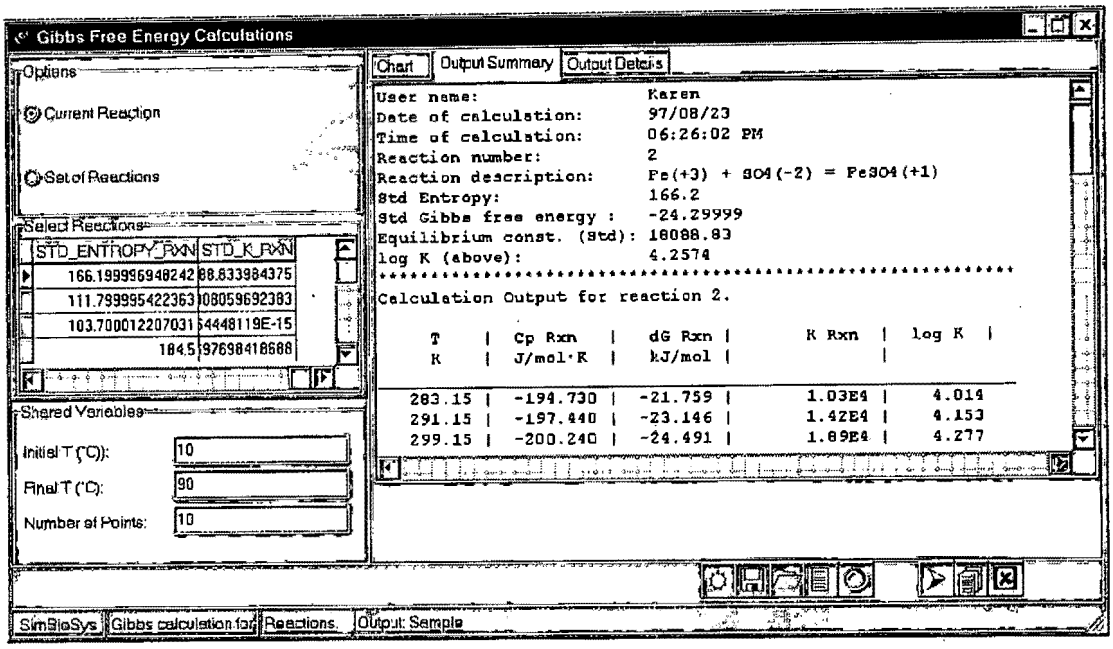


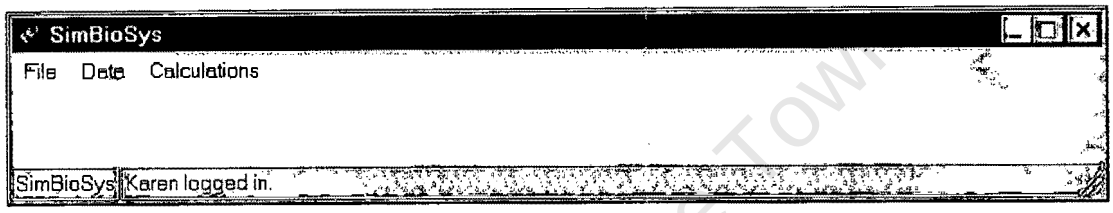


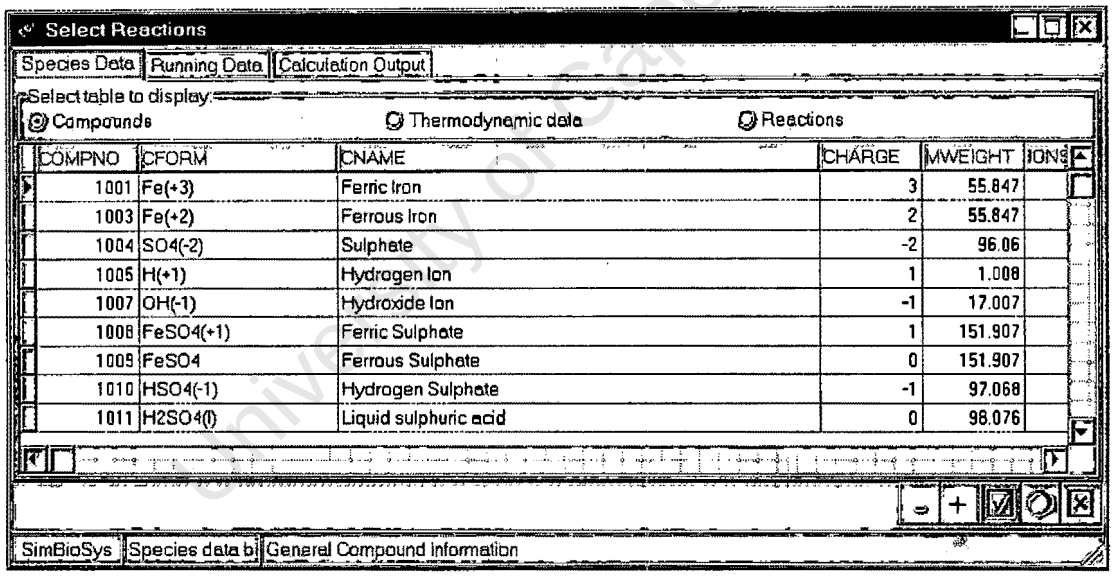


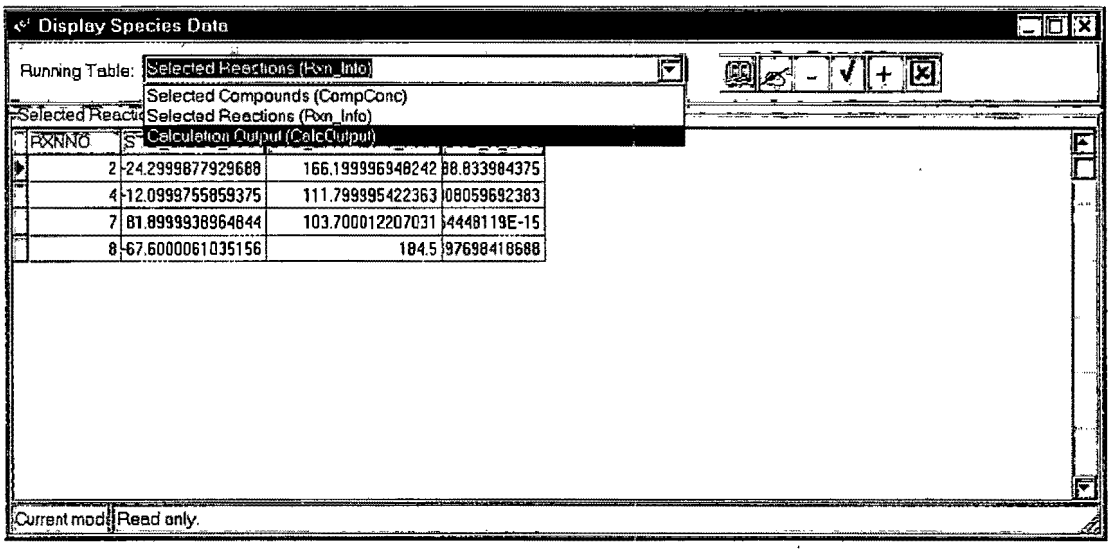





Gibbs Free Energy Calculations
Options
Current Reaction
Set of Reactions
Select Reactions
STD_ENTROPY_RXN
STD_K_RXN
166.199996948242
88.833984375
111.799995422363
08059692383
103.700012207031
4448119E-15
184.5
97698418688
Shared Variables
Initial T (°C):
10
Final T (°C):
90
Number of Points:
10
Chart
Output Summary
Output Details
User name: Karen
Date of calculation: 97/08/23
Time of calculation: 06:26:02 PM
Reaction number: 2
Reaction description: Fe(+3) + SO4(-2) = FeSO4(+1)
Std Entropy: 166.2
Std Gibbs free energy : -24.29999
Equilibrium const. (Std): 18088.83
log K (above): 4.2574
Calculation Output for reaction 2.
T K | Cp Rxn J/mol·K | dG Rxn kJ/mol | K Rxn | log K
283.15 | -194.730 | -21.759 | 1.03E4 | 4.014
291.15 | -197.440 | -23.146 | 1.42E4 | 4.153
299.15 | -200.240 | -24.491 | 1.89E4 | 4.277
SimBioSys
Gibbs calculation for
Reactions.
Output: Sample


SimBioSys
File Data Calculations
SimBioSys
Karen logged in.


Select Reactions
Species Data
Running Data
Calculation Output
Select table to display:
Compounds
Thermodynamic data
Reactions
COMPNO | CFORM | CNAME | CHARGE | MWEIGHT | IONS
1001 | Fe(+3) | Ferric Iron | 3 | 55.847
1003 | Fe(+2) | Ferrous Iron | 2 | 55.847
1004 | SO4(-2) | Sulphate | -2 | 96.06
1005 | H(+1) | Hydrogen Ion | 1 | 1.008
1007 | OH(-1) | Hydroxide Ion | -1 | 17.007
1008 | FeSO4(+1) | Ferric Sulphate | 1 | 151.907
1009 | FeSO4 | Ferrous Sulphate | 0 | 151.907
1010 | HSO4(-1) | Hydrogen Sulphate | -1 | 97.068
1011 | H2SO4(l) | Liquid sulphuric acid | 0 | 98.076
SimBioSys
Species data bi
General Compound Information


Display Species Data
Running Table:
Selected Reactions (Rxn_Info)
Selected Compounds (CompConc)
Selected Reactions (Rxn_Info)
Calculation Output (CalcOutput)
Selected Reacti
RXNNO | S
2 | -24.2999877929688 | 166.199996948242 | 88.833984375
4 | -12.0999755859375 | 111.799995422363 | 08059692383
7 | 81.8999938964844 | 103.700012207031 | 4448119E-15
8 | -67.6000061035156 | 184.5 | 97698418688
Current mod
Read only.

Program Listing

SimBioSys.dpr

CompClassUnit.pas

CompGenDataFormUnit.pas

CompGenDMUnit.pas

Constants.pas

DisplayRunningDataFormUnit.pas

DisplaySpeciesDataFormUnit.pas

Exceptns.pas

GibbsCalcClassUnit.pas

GibbsCalcFormUnit.pas

GibbsCalcProgressUnit.pas

LogInOutFormUnit.pas

MainFormUnit.pas

RxnClassUnit.pas

SelectionFormUnit.pas

UnitsCheckUnit.pas

UserClassUnit.pas

```
 1: program SimBioSys;
 2:
 3: uses
 4:   Forms,
 5:   LogInOutFormUnit in 'LogInOutFormUnit.pas' {LogInOutForm},
 6:   CompGenDMUnit in 'CompGenDMUnit.pas' {CompGenDM: TDataModule},
 7:   RunningDMUnit in 'RunningDMUnit.pas' {RunningDM: TDataModule},
 8:   Exceptns in 'Exceptns.pas',
 9:   DisplayRunningDataFormUnit in 'DisplayRunningDataFormUnit.pas'
     {DisplayRunningDataForm},
10:   DisplaySpeciesDataFormUnit in 'DisplaySpeciesDataFormUnit.pas'
     {DisplaySpeciesDataForm},
11:   SelectionFormUnit in 'SelectionFormUnit.pas' {SelectionForm},
12:   GibbsCalcFormUnit in 'GibbsCalcFormUnit.pas' {GibbsCalcForm},
13:   RxnClassUnit in 'Classes\RxnClassUnit.pas',
14:   GibbsCalcClassUnit in 'Classes\GibbsCalcClassUnit.pas',
15:   UserClassUnit in 'Classes\UserClassUnit.pas',
16:   UnitsCheckUnit in '\Work\SimII\UnitsCheckUnit.pas',
17:   CompClassUnit in 'Classes\CompClassUnit.pas',
18:   MainFormUnit in 'MainFormUnit.pas' {MainForm},
19:   CompGenDataFormUnit in 'CompGenDataFormUnit.pas' {CompGenDataForm},
20:   Constants in 'Constants.pas',
21:   GibbsCalcProgressUnit in 'GibbsCalcProgressUnit.pas' {GibbsCalcProgress};
22:
23: {$R *.RES}
24:
25: begin
26:   Application.Initialize;
27:   Application.Title := 'SimBioSys';
28:   Application.CreateForm(TMainForm, MainForm);
29:   Application.CreateForm(TLogInOutForm, LogInOutForm);
30:   Application.CreateForm(TCompGenDM, CompGenDM);
31:   Application.CreateForm(TRunningDM, RunningDM);
32:   Application.CreateForm(TDisplayRunningDataForm, DisplayRunningDataForm);
33:   Application.CreateForm(TDisplaySpeciesDataForm, DisplaySpeciesDataForm);
34:   Application.CreateForm(TSelectionForm, SelectionForm);
35:   Application.CreateForm(TGibbsCalcForm, GibbsCalcForm);
36:   Application.CreateForm(TCompGenDataForm, CompGenDataForm);
37:   Application.CreateForm(TGibbsCalcProgress, GibbsCalcProgress);
38:   Application.Run;
39: end.
```

```
unit CompClassUnit;

interface

uses
  SysUtils, DB;

type
  { TDescription class, provides containers for generic calculation information
    e.g. Description, UserID, Date, Time, OutputString (for messages)  }
  TDescription = class (TObject)
  private
    UserIDStr     : string;
    OutPutString : string;
  public
    CalcDate      : string;
    CalcTime      : string;
  protected
    Description   : string;

  public
    constructor Create (Descrip : string);
    procedure DisplayOutput;
    procedure EditOutput (NewOutputStr : string);
    function PassCalcDate : string;
    function PassCalcTime : string;
  end; {end Tdescription = class (TObject) }

  { TCrissDetails class declaration }
  TCrissDetails = class (TObject)
  private
    ATM,                                  // a(T) m coefficient (gradient)
    ATC             : single;             // a(T) c coefficient (intercept)
    BTM,                                  // b(T) m coefficient (gradient)
    BTC             : single;             // b(T) c coefficient (intercept)
    alphaT          : single;             // alpha coefficient (from a(T)
    betaT           : single;             // beta coefficieint (from b(T)
    EntropyAbsTRef : single;              // absolute entropy for compound
    CpMeanT         : single;             // mean Cp(T) for particular compound
    HCPM,                                 // H+ - Cp mean T gradient
    HCPC            : single;             // H+ - Cp mean T intercept
    CrissCode       : string;             // ion classification code string

    procedure RetrieveCoefficients;
    procedure CalcAlphaT (TemperatureK : single);
    procedure CalcBetaT (TemperatureK : single);
    procedure CalcEntropyAbsTRef (EntropyTRef : single; Charge : integer);
    procedure CalcCpMeanT(TemperatureK : single);

  public
    Constructor Create (CrissCodeStr : string);
    procedure CalcAlphaBetaT (TemperatureK : single);
    function PassAlphaT : single;
    function PassBetaT : single;
    function PassCpMeanT : single;

  end; {end TCrissDetails = class (TObject) }

  { TNonCrissCpDetails class which serves as a container for the coefficients
    for the Cp power series expression. }
  TNonAqueousDetails = class (TObject)
  private
    CoeffA,                       // a
    CoeffB,                       // bT
    CoeffC,                       // c/T^2
    CoeffD,                       // DT^2
    CoeffE          : single;     // ET^3
    CpMeanT         : single;
    LocateOptions   : TLocateOptions;

  public
    constructor Create;
```

```
73:     procedure CalcCpMeanT(TemperatureK : single);
74:     function PassCpMeanT : single;
75:     procedure RetrieveCpCoeffs(CNoInt : integer);
76:
77:   end; {end TNonAqueousDetails = class (TObject) }
78:
79:   { TCompDetails class with inherited properties and methods from TDesctription
80:     control all data retrieval from CompGen and Thermo for calculations
81:     store current compound variables.  }
82:   TCompDetails = class (TDescription)
83:   protected                              // available to descendant classes
84:     CNoInt         : integer;            // compound number - primary key
85:     CFormStr       : string;             // compound formula - string
86:     CStateStr      : string;             // compound state - e.g. aqI
87:     CCrissStateStr : string;             // criss cobble code for aqI. compounds
88:     CGibbsStdStr   : string;             // compound Gibbs energy (kJ/mol)
89:     CEntropyStdStr : string;             // compound Entropy (J/mol.K)
90:     CChargeInt     : integer;            // compound charge for abs entropy calc.
91:     AqueousI       : boolean;            // is compound aqueous?? -> CrissCobble
92:     Ignore         : boolean;            // is compound to be ignored? (electrons)
93:     CrissDetails   : TCrissDetails;      // Criss Cobble information for compound
94:     NonAqueousDetails : TNonAqueousDetails; // Non Aqueous information
95:
96:   public
97:     constructor Create ;
98:     destructor Destroy;
99:     procedure NewCompound (CompNo : integer);
100:    function GiveFormula : string;
101:    function GiveCompState : string;
102:    function GiveCrissCobbleState : string;
103:    function GiveGibbsStd : string;
104:    function GiveEntropyStd : string;
105:    function GiveCharge : integer;
106:    function CalcCpMeanT(TemperatureK : single) : single;
107:
108:  private
109:    procedure RetrieveData;                    // called by constructor Create
110:    procedure RetrieveCrissDetails;            // called by RetrieveData
111:    procedure RetrieveNonAqueousDetails;       // called by RetrieveData
112:    procedure CalcCrissCoefficients (const TempK : single; var alpha : single;
113:     var beta : single; var CpMeanT : single);
114:
115:
116:  end;  {end TCompDetails = class (TDescription) }
117:
118:
119: implementation
120:
121: uses Dialogs,
122:      Exceptns, CompGenDMUnit, MainFormUnit, Constants;
123:
124: { TDescription constructor
125: ********************************************************************************* }
126: constructor TDescription.Create(Descrip : string);
127: begin
128:   Description    := Descrip;
129:   UserIDStr      := MainForm.User.PassUserID;
130:   OutputString   := 'Nothing to report.';
131:   // format date short date format
132:  { DateOrder      := doDMY;
133:   DateSeparator := '-';
134:   DateFullYear  := True;
135:   DateLeadZero  := True;
136:  }
137:   CalcDate       := DateToStr(Date);
138:   CalcTime       := TimeToStr(Time);
139: end; {end constructor TDescription.Create(DescripStr, UserIDStr, ... );}
140:
141: { TDescription procedure which displays OutputStr in a message dialog box
142: ********************************************************************************* }
143: procedure TDescription.DisplayOutput;
144: begin
```

```
145:   MessageDlg (OutputString, mtInformation, [mbOK], 0);
146: end;  {end TDescription.DisplayOutput }
147:
148: { TDescription procedure to edit the output string which will be displayed
149: ********************************************************************** }
150: procedure TDescription.EditOutput (NewOutputStr : string);
151: begin
152:   OutputString := NewOutputStr;
153: end;  {end procedure TDescription.EditOutput (NewOutputStr : string); }
154:
155:
156: { procedure to pass the stored date to the calling routine
157: ********************************************************************** }
158: function TDescription.PassCalcDate :string;
159: begin
160:   Result := CalcDate;
161: end; {end function TDescription.PassCalcDate :string; }
162:
163: { procedure to pass the stored date to the calling routine
164: ********************************************************************** }
165: function TDescription.PassCalcTime :string;
166: begin
167:   Result := CalcTime;
168: end; {end function TDescription.PassCalcTime :string; }
169:
170: { constructor for TCompDetails (TDescription)
171: ********************************************************************** }
172: constructor TCompDetails.Create ();
173: begin
174:   // set TDescription details
175:   TDescription.Create ('CompoundDetails');
176:   AqueousI  := False;  // assume compound is not aqueous
177: end; {end constructor TCompDetails.Create (CompNo : integer); }
178:
179: { destructor for TCompDetails, mainly to free owned classes
180: ********************************************************************** }
181: destructor TCompDetails.Destroy;
182: begin
183:   CrissDetails.Free;
184: end; {end destructor TCompDetails.Destroy; }
185:
186: { procedure to calculate alpha and beta for given temperature
187:   as well as the corresponding CpMeanT (J/mol·K) for the component
188: ********************************************************************** }
189: procedure TCompDetails.CalcCrissCoefficients (const TempK : single;
190:  var alpha : single; var beta : single; var CpMeanT : single);
191: begin
192:   CrissDetails.CalcAlphaBetaT (TempK);
193:   alpha   := CrissDetails.PassAlphaT;
194:   beta    := CrissDetails.PassBetaT;
195:   {  CEntropyStdStr : string;         // compound Entropy (J/mol.K)
196:      CChargeInt     : integer;        // compound charge for abs entropy calc. }
197:   CrissDetails.CalcEntropyAbsTRef( StrToFloat(CEntropyStdStr), CChargeInt);
198:   CrissDetails.CalcCpMeanT(TempK);
199:   CpMeanT := CrissDetails.PassCpMeanT;
200: end; {end procedure TCompDetails.CalcCoefficients (TempK : single); }
201:
202: { procedure to point CompDetails to another compound
203: ********************************************************************** }
204: procedure TCompDetails.NewCompound (CompNo : integer);
205: begin
206:   CNoInt   := CompNo;
207:   // check compound number is valid
208:   if CNoInt < 1001 then raise ECompNoInvalid.Create
209:    ('Compound number is invalid.');
210:   AqueousI  := False;  // assume compound is not aqueous
211:   // check compound number is valid
212:   { retrieve data }
213:   RetrieveData;
214: end; {end procedure TCompDetails.NewCompound (CompNo : integer); }
215:
216: { procedure to retrieve data for given component.
```

```
217: ********************************************************************************* }
218: procedure TCompDetails.RetrieveData;
219: begin
220:   // Compound Formula from CompGen
221:   if (CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'CFORM')= Null) then
222:     raise EDataNotExist.Create('Compound no '+ IntToStr(CNoInt) +
223:      ' formula data does not exist in CompGenQy.')
224:   else CFormStr := CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'CFORM');
225:
226:   // Compound State from CompGen
227:   if (CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'ST_CODE')= Null) then
228:     raise EDataNotExist.Create('Compound '+ CFormStr +
229:      ' state code data does not exist in CompGenQy.')
230:   else begin
231:     CStateStr := CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'ST_CODE');
232:     if CStateStr = 'aqI' then AqueousI := True
233:       else AqueousI := False;
234:     if CStateStr = 'ignore' then Ignore := True
235:       else Ignore := False;
236:   end; {end else for if (CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, ... }
237:
238:   // CrissCobbleState for aqueous compounds from CompGen
239:     if CStateStr = 'aqI' then
240:     if (CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'CRISS_CODE')= Null) then
241:       raise EDataNotExist.Create('Compound '+ CFormStr +
242:        ' Criss state code does not exist in CompGenQy.')
243:     else CCrissStateStr := CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt,
244:      'CRISS_CODE');
245:
246:   // Compound standard state Gibbs free energy of formation from Comp Thermo
247:   if (CompGenDM.CompThermoQy.Lookup('COMPNO', CNoInt, 'GIBBS_F')= Null) then
248:     raise EDataNotExist.Create(    'Compound ' + CFormStr +
249:     ' std. Gibbs free energy of formation data does not exist in CompThermoQy.')
250:   else CGibbsStdStr :=
251:    CompGenDM.CompThermoQy.Lookup('COMPNO', CNoInt,'GIBBS_F');
252:
253:   // Compound standard entropy from Comp Thermo
254:   if (CompGenDM.CompThermoQy.Lookup('COMPNO', CNoInt, 'ENTROPY')= Null) then
255:     raise EDataNotExist.Create('Compound ' + CFormSTr +
256:      'Entropy data does not exist in CompThermoQy.')
257:   else CEntropyStdStr :=
258:    CompGenDM.CompThermoQy.Lookup('COMPNO', CNoInt,'ENTROPY');
259:
260:    // Compound charge from CompGen
261:   if (CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'CHARGE')= Null) then
262:     raise EDataNotExist.Create('Compound '+ CFormStr +
263:      ' charge data does not exist in CompGenQy.')
264:   else CChargeInt := CompGenDM.CompGenQy.Lookup('COMPNO', CNoInt, 'CHARGE');
265:
266:   // if compound is aqueous, retrieve Criss Cobble constant information
267:   if AqueousI then
268:     RetrieveCrissDetails              // instantiate CrissDetails object
269:   else if CStateStr <> 'aqN' then
270:     RetrieveNonAqueousDetails;        // instantiate NonAqueousDetails object
271: end;  {end procedure TCompDetails.RetrieveData; }
272:
273: { procedure to instantiate CrissDetails object and contain Criss Cobble constant
274:   information within CrissDetails,  called by RetrieveData if aqueous is true
275: ********************************************************************************* }
276: procedure TCompDetails.RetrieveCrissDetails;
277: begin
278:   { note exception handling controlled by calling function, in the same manner
279:     as with RetrieveData - TCrissDetails raises exceptions when neccessary }
280:   // instantiate CrissDetails object
281:   CrissDetails := TCrissDetails.Create(CCrissStateStr);
282:   // retrieve data
283:   CrissDetails.RetrieveCoefficients;
284: end; {end procedure TCompDetails.RetrieveCrissDetails; }
285:
286:  { procedure to instantiate NonAqueousDetails object to contain heat capacity
287:    coefficients NonAqueousDetails,
288:    called by RetrieveData if AqueousI is False and if state is not AqN
```

```
289: ********************************************************************************* }
290: procedure TCompDetails.RetrieveNonAqueousDetails;
291: begin
292:   NonAqueousDetails := TNonAqueousDetails.Create;
293: end; {end procedure TCompDetails.RetrieveNonAqueousDetails; }
294:
295: { function to return the compound state, assuming that the compound number has
296:   been passed correctly, constructor checksto see that CompNo > 1000,
297:   exceptions were raised in retrieval, if neccessary.
298: ********************************************************************************* }
299: function TCompDetails.GiveCompState : string;
300: begin
301:   Result := CStateStr;
302: end; { end function TCompDetails.GiveCompState : string; }
303:
304: { function to return the Criss Cobble state for an aqueous species.
305: ********************************************************************************* }
306: function TCompDetails.GiveCrissCobbleState : string;
307: begin
308:   Result := CCrissStateStr;
309: end;  {end function TCompDetails.GiveCrissCobbleState : string; }
310:
311: { function to return the std Gibbs free energy of formation for the compound
312: ********************************************************************************* }
313: function TCompDetails.GiveGibbsStd : string;
314: begin
315:   Result := CGibbsStdStr;
316: end;  {end function TCompDetails.GiveGibbsStd : string; }
317:
318: { function to return the std Entropy of the compound
319: ********************************************************************************* }
320: function TCompDetails.GiveEntropyStd : string;
321: begin
322:   Result := CEntropyStdStr;
323: end;  {end function TCompDetails.GiveEntropyStd : string; }
324:
325: { function to return the compound formula
326: ********************************************************************************* }
327: function TCompDetails.GiveFormula : string;
328: begin
329:   Result := CFormStr;
330: end;  {end function TCompDetails.GiveFormula : string; }
331:
332: { function to return the compound charge
333: ********************************************************************************* }
334: function TCompDetails.GiveCharge : integer ;
335: begin
336:   Result := CChargeInt;
337: end;  {end function TCompDetails.GiveCharge : string; }
338:
339:
340: { Constructor for TCrissDetails
341: ********************************************************************************* }
342: constructor TCrissDetails.Create (CrissCodeStr : string);
343: begin
344:   CrissCode := CrissCodeStr;
345: end; {end constructor TCrissDetails.Create (CrissCode : string); }
346:
347: { routine to retrieve Criss Cobble coefficient slopes and intercepts
348: ********************************************************************************* }
349: procedure TCrissDetails.RetrieveCoefficients;
350: var
351:   LocateOptions : TLocateOptions;
352: begin
353:   LocateOptions := [loCaseInsensitive];      // note whole word only
354:   // locate appropriate record in Criss Cobble table
355:   if not CompGenDM.CrissCobbleQy.Locate('CRISS_CODE', CrissCode, LocateOptions)
356:   then begin
357:     MessageDlg ('Criss Cobble code not valid, please check compound data.',
358:      mtError, [mbOK], 0);  // this can be deleted when within a try catch loop
359:     raise ECrissDataError.Create
360:      ('Criss Cobble code not valid, please check compound data. ');
```

```
361:   end; {end if not CompGenDM.CrissCobbleQy.Locate('CRISS_CODE', CrissCode,  }
362:
363:   with CompGenDM.CrissCobbleQy do begin
364:     // retrieve coefficient data
365:     if CrissCode = 'H+' then begin
366:       HCPM := FieldByName('HCPM').AsFloat;
367:       HCPC := FieldByName('HCPC').AsFloat;
368:     end {end if CrissCode = 'H+' then begin }
369:     else begin {else for if CrissCode = 'H+' then begin }
370:       ATM := FieldByName ('ATM').AsFloat;
371:       ATC := FieldByName ('ATC').AsFloat;
372:       BTM := FieldByName ('BTM').AsFloat;
373:       BTC := FieldByName ('BTC').AsFloat;
374:     end; {end else for if CrissCode = 'H+' then begin }
375:   end; {end with CompGenDM.CrissCobbleQy do begin }
376: end; {end procedure TCrissDetails.RetrieveCoefficients;}
377:
378: { routine to calculate alpha and beta coefficients, given the temperature
379: ************************************************************************** }
380: procedure TCrissDetails.CalcAlphaBetaT (TemperatureK : single);
381: begin
382:   CalcAlphaT ( TemperatureK );
383:   CalcBetaT ( TemperatureK );
384: end; {end procedure TCrissDetails.CalcAlphaBetaT (TemperatureK : single);
385:
386: { routine to calc Alpha (T) as a single, given T(K)
387: ************************************************************************** }
388: procedure TCrissDetails.CalcAlphaT (TemperatureK : single);
389: var
390:   aT      : single;                          // container for constant 'a'
391: begin
392:   aT      := ATM*(TemperatureK) + ATC;   // note Temperature in °C
393:   alphaT := aT/(ln(TemperatureK/CrissTRef));    // note Temperature in K
394: end; {end function TCrissDetails.CalcAlphaT (TemperatureK : single) : single; }
395:
396: { routine to pass Alpha (T) as a single, given T(K)
397: ************************************************************************** }
398: function  TCrissDetails.PassAlphaT : single;
399: begin
400:   if (alphaT = 0) then
401:     raise ECrissError.Create('Criss coefficients alpha not calculated.')
402:   else
403:     Result := AlphaT;
404: end; {end function  TCrissDetails.PassAlphaT : single; }
405:
406:
407: { routine to calc Beta (T) as a single, given T(K)
408: ************************************************************************** }
409: procedure TCrissDetails.CalcBetaT (TemperatureK : single) ;
410: var
411:   bT      : single;                          // container for constant 'b'
412: begin
413:   bT      := BTM*(TemperatureK) + BTC;   // note Temperature in °C
414:   betaT  := -(1-bT)/(ln(TemperatureK/CrissTRef));    // note Temperature in K
415: end; {end function TCrissDetails.PassBetaT (TemperatureK : single) : single; }
416:
417: { routine to pass Alpha (T) as a single, given T(K)
418: ************************************************************************** }
419: function  TCrissDetails.PassBetaT : single;
420: begin
421:   if (betaT = 0) then
422:     raise ECrissError.Create('Criss coefficients beta not calculated.')
423:   else
424:     Result := BetaT;
425: end; {end function  TCrissDetails.PassAlphaT : single; }
426:
427: { routine to calcultate EntropyAbsTRef from std. entropy (TRef) and charge
428:   units - J/mol
429: ************************************************************************** }
430: procedure TCrissDetails.CalcEntropyAbsTRef (EntropyTRef : single ;
431:   Charge : integer);
432: begin
```

```
433:  EntropyAbsTRef := EntropyTRef -20.9 * Charge;
434: end; {end procedure TCrissDetails.CalcEntropyAbsTRef; }
435:
436: { routine to calculate the CpMean(T) for the aqueous component using Criss
437:   Cobble alpha and beta constants and the absolute entropy of the compound
438:   requires that EntropyAbsTRef be calculated, units - J/mol·K
439: ********************************************************************** }
440: procedure TCrissDetails.CalcCpMeanT(TemperatureK : single);
441: begin
442:  if CrissCode = 'H+' then begin
443:    CpMeanT := HCPM*TemperatureK + HCPC;
444:  end {end if CrissCode = 'H+' then begin }
445:  else {else for if CrissCode = 'H+' then begin }
446:    CpMeanT := alphaT + betaT * EntropyAbsTRef;  // non H+
447: end; {end procedure TCrissDetails.CalcCpMeanT; }
448:
449: { routine to pass CpMean (T) as a single, requires that EntropyAbsTRef
450:   units - J/mol·K
451: ********************************************************************** }
452: function TCrissDetails.PassCpMeanT : single;
453: begin
454:   Result := CpMeanT;
455: end; {end function TCrissDetails.PassCpMeanT : single; }
456:
457: { routine to calculate the CpMean (T) as a single
458:   units - J/mol·K
459: ********************************************************************** }
460: function TCompDetails.CalcCpMeanT(TemperatureK : single) : single;
461: begin
462:   { depending on the classification of the component, e.g. aqueous Ionic or
463:     crystalline a different method of calculating the heat capacity is used.
464:   }
465:   if not Ignore then begin
466:     if CStateStr =  'aqI' then begin       // ionic -> Criss Cobble Correlation
467:       // use the Criss Cobble Correlation to caluclate the mean Cp
468:       CrissDetails.CalcAlphaBetaT (TemperatureK);
469:     {  CEntropyStdStr : string;          // compound Entropy (J/mol.K)
470:        CChargeInt     : integer;         // compound charge for abs entropy calc}
471:       CrissDetails.CalcEntropyAbsTRef( StrToFloat(CEntropyStdStr), CChargeInt);
472:       CrissDetails.CalcCpMeanT(TemperatureK);
473:       Result := CrissDetails.PassCpMeanT;
474:     end {end   if CStateStr =  'aqI' then begin }
475:     else if CstateStr = 'aqN' then begin  // neutral aqueous complex
476:      {else for if CStateStr =  'aqI' then begin }
477:       raise ECpError.Create ('This is a neutral, aqueous complex. '+
478:        'It requires special heat capacity handling.');
479:     end {end   else if CstateStr = 'aqN' then begin }
480:     else begin                            // non-aqueous (ionic or otherwise)
481:      {else for if CStateStr =  'aqI' then , if CStateStr =  'aqN' then begin }
482:       NonAqueousDetails.RetrieveCpCoeffs(CNoInt);
483:       NonAqueousDetails.CalcCpMeanT(TemperatureK);
484:       NonAqueousDetails.PassCpMeanT;
485:     end {end else begin }
486:   end {end if not Ignore then begin }
487:   else begin {else for if not Ignore then begin - i.e. ignore }
488:     // essentially set CpMeanT := 0;
489:     Result := 0;
490:   end; {end for else for if not Ignore then begin - i.e. ignore }
491: end; {end function TCompDetails.CalcCpMeanT : single; }
492:
493: { routine to calculate the CpMean (T) as a single
494:   units - J/mol·K
495: ********************************************************************** }
496: procedure TNonAqueousDetails.RetrieveCpCoeffs (CNoInt : integer);
497: begin
498:   with CompGenDM.CpDataQy do begin
499:     if not Locate('COMPNO', CNoInt, LocateOptions) then
500:       raise ECpError.Create ('Heat capacity data for compound, '+
501:        IntToStr(CNoInt) + ' is not available.')
502:     else begin {else for if not Locate('COMPNO', CNoInt, LocateOptions) then }
503:       // note coefficients should exist for all records as default is 0
504:       CoeffA := FieldByName('Coeff_A').AsFloat;
```

```
505:       CoeffB := FieldByName('Coeff_B').AsFloat;
506:       CoeffC := FieldByName('Coeff_C').AsFloat;
507:       CoeffD := FieldByName('Coeff_D').AsFloat;
508:       CoeffE := FieldByName('Coeff_E').AsFloat;
509:     end; {end else for if not Locate('COMPNO', CNoInt, LocateOptions) then }
510:   end; {end with CompGenDM.CpDataQy do begin }
511: end; {end procedure TCompDetails.RetrieveNonAqCpCoeffs; }
512:
513: { initialise NonAqueousDetails container
514: ******************************************************************************* }
515: constructor TNonAqueousDetails.Create;
516: begin
517:   // initialise all coefficients
518:   CoeffA := 0;
519:   CoeffB := 0;
520:   CoeffC := 0;
521:   CoeffD := 0;
522:   CoeffE := 0;
523:   LocateOptions := [loCaseInsensitive];      // note whole word only
524: end; {end constructor TNonAqueousDetails.Create; }
525:
526: { calculate CpMean for TNonAqueousDetails
527: ******************************************************************************* }
528: procedure TNonAqueousDetails.CalcCpMeanT(TemperatureK : single);
529: begin
530:   CpMeanT := CoeffA + CoeffB*TemperatureK + CoeffC/(TemperatureK * TemperatureK)
531:    + CoeffD*TemperatureK*TemperatureK +
532:    CoeffE*TemperatureK*TemperatureK*TemperatureK;
533: end; {end procedure TNonAqueousDetails.CalcCpMeanT(TemperatureK : single); }
534:
535: { pass CpMean for TNonAqueousDetails to calling routine
536: ******************************************************************************* }
537: function TNonAqueousDetails.PassCpMeanT : single;
538: begin
539:   Result := CpMeanT;
540: end; {end function TNonAqueousDetails.PassCpMeanT : single; }
541:
542: end.
543:
544:
545:
```

```
 1: unit CompGenDataFormUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   StdCtrls, ExtCtrls, ComCtrls, Buttons;
 8:
 9: type
10:   TCompGenDataForm = class(TForm)
11:     StatusBar1: TStatusBar;
12:     PlSpeedButtons: TPanel;
13:     GBCompGen: TGroupBox;
14:     LlNumber: TLabel;
15:     LlFormula: TLabel;
16:     LlName: TLabel;
17:     LlIonicCharge: TLabel;
18:     LlMolecularWeight: TLabel;
19:     LlDHIonSize: TLabel;
20:     LlStateCode: TLabel;
21:     LlCrissCobbleCode: TLabel;
22:     EBName: TEdit;
23:     EBNumber: TEdit;
24:     EBFormula: TEdit;
25:     EBIonicCharge: TEdit;
26:     EBMolecularWeight: TEdit;
27:     EBDHIonicSize: TEdit;
28:     EBStateCode: TEdit;
29:     EBCrissCobbleCode: TEdit;
30:     GBComp_Thermo: TGroupBox;
31:     LlNumber2: TLabel;
32:     EBNumber2: TEdit;
33:     LlGibbsF: TLabel;
34:     EBGibbsF: TEdit;
35:     LlEnthalpyF: TLabel;
36:     EBEntropyS: TEdit;
37:     EBCpS: TEdit;
38:     LlCpS: TLabel;
39:     EBEnthalpyF: TEdit;
40:     EBThermosRefCode: TEdit;
41:     LlEntropyS: TLabel;
42:     LlThermoRefCode: TLabel;
43:     GBCp_Data: TGroupBox;
44:     EBNumber3: TEdit;
45:     LlNumber3: TLabel;
46:     EBStateCode2: TEdit;
47:     LlStateCode2: TLabel;
48:     EBCpRefCode: TEdit;
49:     LlCpRefCode: TLabel;
50:     EBCoeffA: TEdit;
51:     EBCoeffB: TEdit;
52:     EBCoeffC: TEdit;
53:     EBCoeffD: TEdit;
54:     EBCoeffE: TEdit;
55:     LlCoeffA: TLabel;
56:     LlCoeffB: TLabel;
57:     LlCoeffC: TLabel;
58:     LlCoeffD: TLabel;
59:     LlCoeffE: TLabel;
60:     LlCoefficients: TLabel;
61:     BtnClose: TSpeedButton;
62:     BtnUpdate: TSpeedButton;
63:     CBGeneral: TCheckBox;
64:     CBThermos: TCheckBox;
65:     CBCp: TCheckBox;
66:     Label1: TLabel;
67:     BtnFirst: TSpeedButton;
68:     BtnNext: TSpeedButton;
69:     BtnPrevious: TSpeedButton;
70:     BtnAddComp: TSpeedButton;
71:     BtnCancel: TSpeedButton;
72:     procedure BtnCloseClick(Sender: TObject);
```

```
73:     procedure EBFormulaExit(Sender: TObject);
74:     procedure BtnUpdateClick(Sender: TObject);
75:     procedure EBStateCodeChange(Sender: TObject);
76:     procedure EBNumberChange(Sender: TObject);
77:     procedure FormCreate(Sender: TObject);
78:     procedure BtnNextClick(Sender: TObject);
79:     procedure BtnFirstClick(Sender: TObject);
80:     procedure BtnPreviousClick(Sender: TObject);
81:     procedure EBNumberExit(Sender: TObject);
82:     procedure BtnAddCompClick(Sender: TObject);
83:     procedure BtnCancelClick(Sender: TObject);
84:   private
85:     { Private declarations }
86:     procedure ReadInCompoundData;
87:     procedure UpdateGeneralFields(Update : Boolean);
88:     procedure BlankGeneralFields;
89:     procedure BlankThermosFields;
90:     procedure UpdateThermosFields(Update : Boolean);
91:     procedure BlankCpFields;
92:     procedure EnableCpFields (state : Boolean);
93:     procedure UpdateCpFields(Update : Boolean);
94:
95:
96:   public
97:     { Public declarations }
98:
99:   end;
100:
101: var
102:   CompGenDataForm: TCompGenDataForm;
103:
104: implementation
105:
106: uses
107:   CompGenDMUnit,
108:   DB, Exceptns;
109:
110: {$R *.DFM}
111:
112: var
113:   LocateOptions : TLocateOptions;   // initialised in 'initialization' section
114:
115: { procedure to close the form, and pass active focus to the main form
116: ************************************************************************** }
117: procedure TCompGenDataForm.BtnCloseClick(Sender: TObject);
118: begin
119:   Close;
120: end; {end procedure TCompGenDataForm.BtnCloseClick(Sender: TObject); }
121:
122: { routine to check whether component formula exists, or whether it needs to be
123:   inserted into the database table, if it exists, does it need to be edited?
124: ************************************************************************** }
125: procedure TCompGenDataForm.EBFormulaExit(Sender: TObject);
126: begin
127:   if CompGenDM.CompGenQy.Locate('CForm', EBFormula.Text, LocateOptions) then
128:   begin
129:     // compound already exists
130:     ReadInCompoundData;
131:   end {end if CompGenDM.CompGenQy.Locate('CForm', EBFormula.Text,  ...}
132:   else begin {else for if CompGenDM.CompGenQy.Locate('CForm', EBFormula....}
133:     // compound does not exist
134:     BlankGeneralFields;
135:     BlankThermosFields;
136:     BlankCpFields;
137:   end; {end else for if CompGenDM.CompGenQy.Locate('CForm', EBFormula....}
138:
139: end; {end procedure TCompGenDataForm.EBFormulaExit(Sender: TObject); }
140:
141: { procedure to read in the compound specific data for the given Formula
142: ************************************************************************** }
143: procedure TCompGenDataForm.ReadInCompoundData;
144: begin
```

```
145:   // note use locate routine to ensure correct implementation of FieldByName
146:   if CompGenDM.CompGenQy.Locate('CForm', EBFormula.Text, LocateOptions) then
147:   begin
148:     with CompGenDM.CompGenQy do begin
149:     {  SHOW TABLE compgen
150:          COMPNO                          (DCLASS NO) INTEGER Not Null
151:                                          CHECK (VALUE>1000)
152:          CFORM                           (DFORMULA) VARCHAR(20) Not Null
153:          CNAME                           (DNAME) VARCHAR(50) Not Null
154:          CHARGE                          (DSSINT) SMALLINT Not Null DEFAULT 0
155:                                          CHECK (VALUE BETWEEN -15 AND 15)
156:          MWEIGHT                         (DSINGLE5) NUMERIC(15, 5) Not Null
157:          IONSIZE                         (DSSPINT) SMALLINT Nullable DEFAULT 0
158:                                          CHECK (VALUE BETWEEN -1 AND 30)
159:          ST CODE                         CHAR(10) Nullable
160:          CRISS CODE                      (DCHAR_CODE) CHAR(10) Nullable
161:          CONSTRAINT INTEG 6:
162:            Unique key (CFORM)
163:          CONSTRAINT INTEG 8:
164:            Unique key (CNAME)
165:          CONSTRAINT PCOMPNO:
166:            Primary key (COMPNO)
167:
168:          Triggers on Table COMPGEN:
169:          SET COMPNO, Sequence: 0, Type: BEFORE INSERT, Active }
170:       EBNumber.Text             := FieldByName('COMPNO').AsString;
171:       EBName.Text               := FieldByName('CNAME').AsString;
172:       EBIonicCharge.Text        := FieldByName('CHARGE').AsString;
173:       EBMolecularWeight.Text    := FieldByName ('MWEIGHT').AsString;
174:       EBDHIonicSize.Text        := FieldByName ('IONSIZE').AsString;
175:       EBStateCode.Text          := FieldByName ('ST_CODE').AsString;
176:       EBCrissCobbleCode.Text    := FieldByName ('CRISS_CODE').AsString;
177:     end; {end with CompGenDM.CompGenQy do begin }
178:     with CompGenDM.CompThermoQy do begin
179:      { SHOW TABLE comp thermo
180:          COMPNO                          (DCLASS NO) INTEGER Not Null
181:                                          CHECK (VALUE>1000)
182:          GIBBS F                         FLOAT Nullable
183:          ENTHALPY F                      FLOAT Nullable
184:          ENTROPY                         FLOAT Nullable
185:          HEATCAP                         FLOAT Nullable
186:          REFCODE                         CHAR(10) Not Null }
187:       // locate the appropriate record in Comp_Thermo table
188:       if not Active then Open;
189:       if Locate ('COMPNO', StrToInt(EBNumber.Text), LocateOptions) then begin
190:         EBNumber2.Text          := FieldByName('COMPNO').AsString;
191:         EBGibbsF.Text           := FieldByName('GIBBS_F').AsString;
192:         EBEnthalpyF.Text        := FieldByName('ENTHALPY_F').AsString;
193:         EBEntropyS.Text         := FieldByName('ENTROPY').AsString;
194:         EBCpS.Text              := FieldByName('HEATCAP').AsString;
195:         EBThermosRefCode.Text := FieldByName('REFCODE').AsString;
196:       end {end if Locate ('COMPNO', StrToInt(EBNumber.Text), LocateOptions) }
197:       else begin {else for if Locate ('COMPNO', StrToInt(EBNumber.Text), ... }
198:         BlankThermosFields;
199:         MessageDlg('Thermodynamic data missing for compound ' + EBFormula.Text
200:           + '.', mtWarning, [mbOK],0);
201:       end; {end else for if Locate ('COMPNO', StrToInt(EBNumber.Text), ... }
202:     end; {end with CompGenDM.CompThermoQy do begin }
203:     with CompGenDM.CpDataQy do begin
204:      { SHOW TABLE cp data
205:          COMPNO                    (DCLASS NO) INTEGER Not Null
206:                                    CHECK (VALUE>1000)
207:          COEFF A                   (DSINGLE5) NUMERIC(15, 5) Nullable  DEFAULT 0
208:          COEFF B                   (DSINGLE5) NUMERIC(15, 5) Nullable  DEFAULT 0
209:          COEFF C                   (DSINGLE5) NUMERIC(15, 5) Nullable  DEFAULT 0
210:          COEFF D                   (DSINGLE5) NUMERIC(15, 5) Nullable  DEFAULT 0
211:          COEFF E                   (DSINGLE5) NUMERIC(15, 5) Nullable  DEFAULT 0
212:          REFCODE                   CHAR(10) Not Null
213:          ST_CODE                   (DCHAR_CODE) CHAR(10) Nullable }
214:
215:       // first of all, this data only applies to non-aqueous compounds
216:       if EBStateCode.Text <> 'aqI' then begin
```

```
217:         EnableCpFields (True);
218:         if not Active then Open;
219:         // locate appropriate record in Cp Data Table
220:         if Locate ('COMPNO', StrToInt(EBNumber.Text), LocateOptions) then begin
221:           EBNumber3.Text     := FieldByName('COMPNO').AsString;
222:           EBStateCode2.Text  := FieldByName('ST_CODE').AsString;
223:           EBCpRefCode.Text   := FieldByName('REFCODE').AsString;
224:           EBCoeffA.Text      := FieldByName('COEFF_A').AsString;
225:           EBCoeffB.Text      := FieldByName('COEFF_B').AsString;
226:           EBCoeffC.Text      := FieldByName('COEFF_C').AsString;
227:           EBCoeffD.Text      := FieldByName('COEFF_D').AsString;
228:           EBCoeffE.Text      := FieldByName('COEFF_E').AsString;
229:         end {end if Locate ('COMPNO', StrToInt(EBNumber.Text), LocateOptions) }
230:         else begin {else for if Locate ('COMPNO', StrToInt(EBNumber.Text), ...}
231:           BlankCpFields;
232:           MessageDlg('Heat capacity data missing for compound ' + EBFormula.Text
233:            + '.', mtWarning, [mbOK],0);
234:         end {end else for if Locate ('COMPNO', StrToInt(EBNumber.Text), ...}
235:       end { end if EBStateCode.Text <> 'aqI', then begin }
236:       else begin {else for if EBStateCode.Text <> 'aqI', then begin }
237:         BlankCpFields;
238:         EnableCpFields (False);
239:       end; {end else for if EBStateCode.Text <> 'aqI', then begin }
240:     end; {end with CompGenDM.CpDataQy do begin }
241:   end; {end if CompGenDM.CompGenQy.Locate('CForm', EBFormula.Text, ... }
242: end; {end procedure TCompGenDataForm.ReadInCompoundData; }
243:
244: { procedure to place blanks in General fields
245: ********************************************************************************* }
246: procedure TCompGenDataForm.BlankGeneralFields;
247: begin
248:    // blank all General fields, except Formula fields
249:    EBNumber.Text             := '';
250:    EBName.Text               := '';
251:    EBIonicCharge.Text        := '';
252:    EBMolecularWeight.Text    := '';
253:    EBDHIonicSize.Text        := '';
254:    EBStateCode.Text          := '';
255:    EBCrissCobbleCode.Text    := '';
256: end; {end procedure TCompGenDataForm.BlankGeneralFields; }
257:
258:
259:
260: { procedure to place blanks in Thermos fields
261: ********************************************************************************* }
262: procedure TCompGenDataForm.BlankThermosFields;
263: begin
264:    // blank all thermos fields
265:    EBNumber2.Text       := EBNumber.Text;
266:    EBGibbsF.Text        := '';
267:    EBEnthalpyF.Text     := '';
268:    EBEntropyS.Text      := '';
269:    EBCpS.Text           := '';
270:    EBThermosRefCode.Text := '';
271: end; {end procedure TCompGenDataForm.BlankThermosFields; }
272:
273: { procedure to place blanks in Cp fields
274: ********************************************************************************* }
275: procedure TCompGenDataForm.BlankCpFields;
276: begin
277:    // blank all heat capacity fields
278:    EBNumber3.Text    := EBNumber.Text;
279:    EBStateCode2.Text := EBStateCode.Text;
280:    EBCpRefCode.Text  := '';
281:    EBCoeffA.Text     := '';
282:    EBCoeffB.Text     := '';
283:    EBCoeffC.Text     := '';
284:    EBCoeffD.Text     := '';
285:    EBCoeffE.Text     := '';
286: end; {end procedure TCompGenDataForm.BlankCpFields; }
287:
288: { procedure to enable or disable Cp fields according to Boolean passed
```

```
289: ********************************************************************************* }
290: procedure TCompGenDataForm.EnableCpFields (state : Boolean);
291: begin
292:    EBNumber3.Enabled    := State;
293:    EBStateCode2.Enabled := State;
294:    EBCpRefCode.Enabled  := State;
295:    EBCoeffA.Enabled     := State;
296:    EBCoeffB.Enabled     := State;
297:    EBCoeffC.Enabled     := State;
298:    EBCoeffD.Enabled     := State;
299:    EBCoeffE.Enabled     := State;
300: end; {end procedure TCompGenDataForm.EnableCpFields (state : Boolean); }
301:
302: { procedure to edit non-aqueous (ionic) heat capacity record(s)
303:   to update existing record then flag update = True,
304:   else insert new record (if insert = False )
305: ********************************************************************************* }
306: procedure TCompGenDataForm.UpdateCpFields(Update : Boolean);
307: var
308:   SQLStr   : string;                              // eSQL for update
309:   CoeffA,
310:   CoeffB,
311:   CoeffC,
312:   CoeffD,
313:   CoeffE   : single;                              // coefficients
314: begin
315:   // if non-aqueous ( equivalent to non-ionic)
316:   if EBStateCode.Text <> 'aqI' then begin
317:     { a complication arises when trying to convert an empty string to a
318:       floating point value, which is necessary for the update and insert
319:       SQL strings, hence if the string is empty, then the value is 0 }
320:     if EBCoeffA.Text = '' then CoeffA := 0
321:     else CoeffA := StrToFloat(EBCoeffA.Text);
322:     if EBCoeffB.Text = '' then CoeffB := 0
323:     else CoeffB := StrToFloat(EBCoeffB.Text);
324:     if EBCoeffC.Text = '' then CoeffC := 0
325:     else CoeffC := StrToFloat(EBCoeffC.Text);
326:     if EBCoeffD.Text = '' then CoeffD := 0
327:     else CoeffD := StrToFloat(EBCoeffD.Text);
328:     if EBCoeffE.Text = '' then CoeffE := 0
329:     else CoeffE := StrToFloat(EBCoeffE.Text);
330:
331:     with CompGenDM.CpDataQy do begin
332:       // one cannot update a record which does not exist, hence
333:       if not Active then Open;
334:       Update := Locate('COMPNO', EBNumber.Text, LocateOptions);
335:
336:       if Active then Close;
337:       SQL.Clear;
338:       if RequestLive then RequestLive := False;
339:       if Update then begin // true is update
340:         SQLStr := 'UPDATE CP_DATA SET COEFF_A = %g,COEFF_B = %g,COEFF_C = %g,'+
341:           'COEFF_D = %g, COEFF_E = %g, REFCODE = "%s", ST_CODE = "%s" ' +
342:           'WHERE (COMPNO = %d);';
343:         SQL.Add(Format(SQLStr, [CoeffA, CoeffB, CoeffC, CoeffD, CoeffE,
344:           EBCpRefCode.Text, EBStateCode2.Text, StrToInt(EBNumber3.Text)]));
345:       end {end if Update then begin // true is update }
346:       else begin {else for if Update then begin // true is update }
347:         SQLStr :=
348:           'INSERT INTO CP_DATA (COMPNO, COEFF_A, COEFF_B, COEFF_C, COEFF_D, ' +
349:         'COEFF_E, REFCODE, ST_CODE) VALUES(%d, %g, %g, %g, %g, %g, "%s", "%s")';
350:         // CompNo, a, b, c, d, e, RefCode, St Code
351:         SQL.Add(Format(SQLStr,[StrToInt(EBNumber3.Text),
352:           CoeffA, CoeffB, CoeffC, CoeffD, CoeffE,
353:           EBCpRefCode.Text, EBStateCode2.Text]));
354:       end; {end else for if Update then begin // true is update }
355:       // carry out eSQL
356:       ExecSQL;
357:       SQL.Clear;
358:       SQL.Add ('SELECT * FROM CP_DATA ORDER BY COMPNO');
359:       if not RequestLive then RequestLive := True;
360:       if not Active then Open;
```

```
361:     end; {end with CompGenDM.CpDataQy do begin }
362:   end; {end if EBStateCode.Text <> 'aqI' then begin }
363: end; {end procedure TCompGenDataForm.UpdateCpFields(Update : Boolean);
364:
365: { procedure to edit thermodynamic record(s)
366:   to update existing record then flag update = True,
367:   else insert new record (if insert = False )
368: ********************************************************************** }
369: procedure TCompGenDataForm.UpdateThermosFields(Update : Boolean);
370: var
371:   SQLStr   : string;                          // eSQL for update
372:   EntropyS,
373:   EnthalpyF,
374:   GibbsF,
375:   HeatCap : single;                          // floating point values
376: begin
377:   { a complication arises when trying to convert an empty string to a
378:     floating point value, which is necessary for the update and insert
379:     SQL strings, hence if the string is empty, then the value is or NULL }
380:   if EBGibbsF.Text = '' then GibbsF := NULL
381:   else GibbsF := StrToFloat(EBGibbsF.Text);
382:   if EBEnthalpyF.Text = '' then EnthalpyF := NULL
383:   else EnthalpyF := StrToFloat(EBEnthalpyF.Text);
384:   if EBEntropyS.Text = '' then EntropyS := NULL
385:   else EntropyS := StrToFloat(EBEntropyS.Text);
386:   if EBCpS.Text = '' then HeatCap := NULL
387:   else HeatCap := StrToFloat(EBCpS.Text);
388:
389:   with CompGenDM.CompThermoQy do begin
390:     // one cannot update a record which does not exist, hence
391:     if not Active then Open;
392:     Update := Locate('COMPNO', EBNumber.Text, LocateOptions);
393:
394:     if Active then Close;
395:     SQL.Clear;
396:     if RequestLive then RequestLive := False;
397:     if Update then begin // true is update
398:       SQLStr:= 'UPDATE COMP_THERMO SET GIBBS_F = %g, ENTHALPY_F = %g, ' +
399:        ' ENTROPY = %g, HEATCAP = %g, REFCODE = "%s" WHERE (COMPNO = %d);';
400:       SQL.Add(Format(SQLStr, [GibbsF, EnthalpyF, EntropyS, HeatCap,
401:        EBThermosRefCode.Text, StrToInt(EBNumber2.Text)]));
402:     end {end if Update then begin // true is update }
403:     else begin {else for if Update then begin // true is update }
404:       SQLStr :=
405:       'INSERT INTO COMP_THERMO (COMPNO, GIBBS_F, ENTHALPY_F, ENTROPY, HEATCAP,'+
406:       ' REFCODE) VALUES(%d, %g, %g, %g, %g, "%s")';
407:       // CompNo, Gibbs F, Enthalpy F, Entropy, HeatCap, RefCode
408:       SQL.Add(Format(SQLStr,[StrToInt(EBNumber2.Text),
409:        GibbsF, EnthalpyF, EntropyS, HeatCap, EBThermosRefCode.Text]));
410:     end; {end else for if Update then begin // true is update }
411:     // carry out eSQL
412:     ExecSQL;
413:     SQL.Clear;
414:     SQL.Add ('SELECT * FROM COMP_THERMO ORDER BY COMPNO');
415:     if not RequestLive then RequestLive := True;
416:     if not Active then Open;
417:   end; {end with CompGenDM.CompThermoQy do begin }
418: end; {end procedure TCompGenDataForm.UpdateThermosFields(Update : Boolean); }
419:
420: { procedure to edit general record(s)
421:   to update existing record then flag update = True,
422:   else insert new record (if insert = False )
423: ********************************************************************** }
424: procedure TCompGenDataForm.UpdateGeneralFields(Update : Boolean);
425: var
426:   SQLStr   : string;                          // eSQL for update
427:   IonSize  : integer;                         // floating point values
428: begin
429:   { a complication arises when trying to convert an empty string to a
430:     floating point value, which is necessary for the update and insert
431:     SQL strings, hence if the string is empty, then the value is or NULL }
432:   if EBDHIonicSize.Text = '' then IonSize := -1
```

```
433:   else IonSize := StrToInt(EBDHIonicSize.Text);
434:
435:   { several fields may not be empty or NULL }
436:   if EBMolecularWeight.Text = '' then raise EDataError.Create(
437:     'Please insert a valid number in the molecular weight field.');
438:   if EBIonicCharge.Text = '' then raise EDataError.Create(
439:     'Please insert a valid integer for the ionic charge in the field.');
440:
441:   with CompGenDM do begin
442:   { further, St Code and Criss_Code are restricted by values already in the
443:     respective tables. }
444:     if not StateQy.Active then StateQy.Open;
445:     if not StateQy.Locate('ST_CODE', EBStateCode.Text, LocateOptions) then
446:       raise EDataError.Create(
447:        'Please select a state code already in the table, ' +
448:        'or define a new code then re-enter the data. ');
449:     // if the compound is aqI then
450:     if EBCrissCobbleCode.Text = 'aqI' then begin
451:       if not CrissDescrQy.Active then CrissDescrQy.Open;
452:       if not CrissDescrQy.Locate('CRISS_CODE', EBCrissCobbleCode.Text,
453:        LocateOptions) then
454:         raise EDataError.Create(
455:      'Please select a Criss Cobble code from those in CrissCodeDescr table.');
456:     end {end if EBCrissCobbleCode.Text = 'aqI' then begin }
457:     else {else for if EBCrissCobbleCode.Text = 'aqI' then begin }
458:     EBCrissCobbleCode.Text := ''; // to ensure no invalid entries
459:
460:   end; {end   with CompGenDM do begin }
461:   with CompGenDM.CompGenQy do begin
462:     if not Active then Open;
463:     { one cannot update a record which does not exist, hence, you need to check
464:       whether or not the formula already exists in CompGen table.  Formula as
465:       number may not have been alocated yet. }
466:     Update := Locate('CFORM', EBFormula.Text, LocateOptions);
467:     if Active then Close;
468:     SQL.Clear;
469:     if RequestLive then RequestLive := False;
470:     if Update then begin // true is update
471:       SQLStr:= 'UPDATE COMPGEN SET CFORM = "%s", CNAME = "%s", CHARGE = %d, ' +
472:        'MWEIGHT = %g, IONSIZE = %d, ST_CODE = "%s", CRISS_CODE = "%s" ' +
473:        'WHERE (COMPNO = %d);';
474:       SQL.Add(Format(SQLStr, [EBFormula.Text, EBName.Text,
475:        StrToInt (EBIonicCharge.Text), StrToFloat(EBMolecularWeight.Text),
476:        IonSize, EBStateCode.Text, EBCrissCobbleCode.Text,
     StrToInt(EBNumber.Text)]));
477:     end {end if Update then begin // true is update }
478:     else begin {else for if Update then begin // true is update }
479:       SQLStr :=
480:        'INSERT INTO COMPGEN (COMPNO, CFORM, CNAME, CHARGE, MWEIGHT, IONSIZE, '+
481:        ' ST_CODE, CRISS_CODE) VALUES("%s", "%s", "%s", %d, %g, %d, "%s", "%s")';
482:       // CompNo CForm, CName, Charge, MWeight, IonSize, St Code, Criss Code
483:       SQL.Add(Format(SQLStr,[EBNumber.Text, EBFormula.Text, EBName.Text,
484:       StrToInt (EBIonicCharge.Text), StrToFloat(EBMolecularWeight.Text),
485:       IonSize, EBStateCode.Text, EBCrissCobbleCode.Text]));
486:     end; {end else for if Update then begin // true is update }
487:     // carry out eSQL
488:     ExecSQL;
489:
490:     SQL.Clear;
491:     SQL.Add ('SELECT * FROM COMPGEN ORDER BY COMPNO');
492:     if not RequestLive then RequestLive := True;
493:     if not Active then Open;
494:     if not Update then begin          // place the compno in the other fields
495:       if Locate('CFORM', EBFormula.Text, LocateOptions) then
496:         EBNumber.Text := FieldByName('COMPNO').AsString
497:       else {else for if Locate('CFORM', EBFormula.Text, LocateOptions) then }
498:         raise EDataError.Create('Data not entered into CompGen table.');
499:     end; {end if not Update then begin   }
500:   end; {end with CompGenDM.CompGenQy do begin }
501: end; {end procedure TCompGenDataForm.UpdateGeneralFields(Update : Boolean); }
502:
503: { procedure to edit record(s)
```

```
504:   to insert then flag := 'insert'
505:   to update then flag := 'update'
506: ********************************************************************** }
507: procedure TCompGenDataForm.BtnUpdateClick(Sender: TObject);
508: var
509:   Update        : boolean;                          // true for update
510:   SomethingDone : boolean;                          // false initially
511: begin
512:   try
513:     SomethingDone := False;
514:     with CompGenDM.CompGenQy do begin
515:       // check whether an update or an insert is required
516:       if not Active then open;
517:       if Locate('CFORM', EBFormula.Text, LocateOptions) then Update := True;
518:       if CBGeneral.Checked = True then begin
519:         UpdateGeneralFields (Update);
520:         SomethingDone := True;
521:       end; {end if CBGeneral.Checked = True then  }
522:       if CBCp.Checked = True then begin
523:         UpdateCpFields (Update);
524:         SomethingDone := True;
525:       end; {end if CBCp.Checked = True then  }
526:       if CBThermos.Checked = True then begin
527:         UpdateThermosFields (Update);
528:         SomethingDone := True;
529:       end; {end if CBThermos.Checked = True then  }
530:       // return to previous position
531:       if not Active then Open;
532:       Locate('CFORM', EBFormula.Text, LocateOptions);
533:       if not SomethingDone then ShowMessage
534:        ('Nothing updated, check update options.');
535:     end; {end with CompGenDM.CompGenQy do begin}
536:
537:   except
538:     on E:EDataError do begin
539:      MessageDlg (E.Message, mtError, [mbOK], 0);
540:     end; {end on E:EDataError do begin }
541:   end; {end try ... except block }
542: end; {end procedure TCompGenDataForm.BtnUpdateClick(Sender: TObject);}
543:
544: { procedure to ensure that a similar state is maintained in all related tables
545: ********************************************************************** }
546: procedure TCompGenDataForm.EBStateCodeChange(Sender: TObject);
547: begin
548:   EBStateCode2.Text := EBStateCode.Text;
549: end; {end procedure TCompGenDataForm.EBStateCodeChange(Sender: TObject); }
550:
551: { procedure to ensure that a similar number is maintained in all related tables
552: ********************************************************************** }
553: procedure TCompGenDataForm.EBNumberChange(Sender: TObject);
554: begin
555:   EBNumber2.Text := EBNumber.Text;
556:   EBNumber3.Text := EbNumber.Text;
557: end; {end procedure TCompGenDataForm.EBNumberChange(Sender: TObject); }
558:
559: { procedure to blank out all the fields on CompGenDataForm create
560: ********************************************************************** }
561: procedure TCompGenDataForm.FormCreate(Sender: TObject);
562: begin
563:    // compound does not exist
564:   BlankGeneralFields;
565:   BlankThermosFields;
566:   BlankCpFields;
567: end; {end procedure TCompGenDataForm.FormCreate(Sender: TObject); }
568:
569: { procedure to move to next compound formula in CompGen table
570: ********************************************************************** }
571: procedure TCompGenDataForm.BtnNextClick(Sender: TObject);
572: begin
573:   with CompGenDM.CompGenQy do begin
574:     Next;
575:     EBFormula.Text := FieldByName('CFORM').AsString;
```

```
576:
577:    end; {end with CompGenDM.CompGenQy do begin }
578: end; {procedure TCompGenDataForm.SpeedButton2Click(Sender: TObject); }
579:
580: { procedure to move to the first compound formula in CompGen table
581: ********************************************************************** }
582: procedure TCompGenDataForm.BtnFirstClick(Sender: TObject);
583: begin
584:    with CompGenDM.CompGenQy do begin
585:      First;
586:      EBFormula.Text := FieldByName('CFORM').AsString;
587:    end; {end with CompGenDM.CompGenQy do begin }
588: end; {end procedure TCompGenDataForm.BtnFirstClick(Sender: TObject); }
589:
590: { procedure to move to the first compound formula in CompGen table
591: ********************************************************************** }
592: procedure TCompGenDataForm.BtnPreviousClick(Sender: TObject);
593: begin
594:    with CompGenDM.CompGenQy do begin
595:      Prior;
596:      EBFormula.Text := FieldByName('CFORM').AsString;
597:    end; {end with CompGenDM.CompGenQy do begin }
598: end; {end procedure TCompGenDataForm.BtnPreviousClick(Sender: TObject); }
599:
600: procedure TCompGenDataForm.EBNumberExit(Sender: TObject);
601: begin
602:    with CompGenDM.CompGenQy do begin
603:      if Locate ('COMPNO', StrToInt(EBNumber.Text),[]) then
604:        EBFormula.Text := FieldByName('CFORM').AsString
605:      else {else for  if Locate ('COMPNO', StrToInt(EBNumber.Text),[]) then }
606:      begin
607:        ShowMessage('Compound does not exist, please enter the formula first. ' +
608:          'Compound numbers automatically assigned.');
609:        EBNumber.Text := '';
610:      end; {end else for if Locate ('COMPNO', StrToInt(EBNumber.Text),[]) then }
611:    end; {end with CompGenQy do begin }
612: end;
613:
614: { procedure to add a compound into CompGen, essentially add a new compound
615: ********************************************************************** }
616: procedure TCompGenDataForm.BtnAddCompClick(Sender: TObject);
617: var
618:    LastCompNo : integer;
619:    NewCompNo  : integer;
620: begin
621:    with CompGenDM do begin
622:      with CompGenQy do begin
623:        if Active then Close;
624:        if RequestLive then RequestLive := False;
625:        SQL.Clear;
626:        SQL.Add('SELECT * FROM COMPGEN ORDER BY COMPNO');  // i.e. last will be last
627:        if not Active then Open;
628:        if not RequestLive then RequestLive := True;
629:        Last;                                              // move to last record
630:        LastCompNo := FieldByName('COMPNO').AsInteger;
631:        NewCompNo := LastCompNo + 1;
632:        EBFormula.Text :=   InputBox('SimBioSys', 'Input compound formula.',
633:         'New Compound');
634:        EBNumber.Text := IntToStr(NewCompNo);
635:
636:      end; {end with CompGenQy do begin}
637:    end; {end with CompGenDM do begin }
638: end; {end procedure TCompGenDataForm.BtnAddCompClick(Sender: TObject); }
639:
640: { procedure to cancel previous post or insert
641: ********************************************************************** }
642: procedure TCompGenDataForm.BtnCancelClick(Sender: TObject);
643: begin
644:    with CompGenDM.CompGenQy do begin
645:      Cancel;
646:      if Active then Close;
647:      if not RequestLive then RequestLive := True;
```

```
648:     SQL.Clear;
649:     SQL.Add('SELECT * FROM COMPGEN ORDER BY COMPNO');
650:     if not Active then Open;
651:     if not RequestLive then RequestLive:= True;
652:  end; {end with CompGenDM.CompGenQy do begin }
653: end; {end procedure TCompGenDataForm.BtnCancelClick(Sender: TObject); }
654:
655: initialization
656:   LocateOptions := [loCaseInsensitive];      // note whole word only
657: //end; {end initialization begin }
658:
659: end.
```

```
 1: unit CompGenDMUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   DB, DBTables;
 8:
 9: type
10:   TCompGenDM = class(TDataModule)
11:     CompGenDS: TDataSource;
12:     CompGenQy: TQuery;
13:     RxnDescrQy: TQuery;
14:     StateQy: TQuery;
15:     StateDS: TDataSource;
16:     RxnDescrDS: TDataSource;
17:     SpeciesDB: TDatabase;
18:     RxnStoichQy: TQuery;
19:     RxnStoichDS: TDataSource;
20:     CompThermoDS: TDataSource;
21:     RefDescrDS: TDataSource;
22:     CrissCobbleQy: TQuery;
23:     CrissCobbleDS: TDataSource;
24:     CrissDescrQy: TQuery;
25:     CrissDescrDS: TDataSource;
26:     CpDataDS: TDataSource;
27:     CpDataQy: TQuery;
28:     CompThermoQy: TQuery;
29:     RefDescrQy: TQuery;
30:   private
31:     { Private declarations }
32:   public
33:     { Public declarations }
34:     function GiveSelRxnEntries(var RsRxnEntries, LkRxnNo : Integer) : Boolean;
35:     procedure SetRxnNoFilter (SelRxnNo : Integer);
36:     function FindNextRxnStoich : Boolean;
37:     function FindFirstRxnStoich : Boolean;
38:     function GiveCompFormula(LkCompNo : Integer;  var Error : Boolean) : String;
39:     function GiveCompGibbsF(LkCompNo : Integer;  var Error : Boolean): String;
40:     function GiveCompEntropy(LkCompNo : Integer; var Error : Boolean): String;
41:     function GiveRxnDescr(LkRxnNo : Integer; var Error : Boolean) : String;
42:     procedure SelectAllRxnStoich;
43:
44:   end;
45:
46: var
47:   CompGenDM: TCompGenDM;
48:
49: implementation
50:
51: {$R *.DFM}
52:
53: uses
54:   Exceptns;
55:
56: { note the abbreviations:
57:       Lk                lookup variable
58:       Rs                result variable
59:   note the error conventions:
60:       true              if an error occurs
61:       false             if no error occurs            }
62:
63:
64: // function to pass the RxnEntries given the SelRxnNo
65: // *****************************************************************************
66: function TCompGenDM.GiveSelRxnEntries( var RsRxnEntries, LkRxnNo : Integer)
67:   : Boolean;
68: begin
69:   with RxnDescrQy do begin
70:     if Lookup('RxnNo', IntToStr(LkRxnNo), 'Entries') = NULL then begin
71:       MessageDlg ('No database entry for stoichiometric entries for reaction '
72:         + IntToStr(LkRxnNo) + '.  Please edit the data before continuing.',
```

```
73:         mtError, [mbOK],0);
74:       RsRxnEntries := -1;              // error code for calculation
75:       Result := True;
76:     end {end if Lookup... then }
77:     else begin
78:       RsRxnEntries := Lookup('RxnNo', IntToStr(LkRxnNo), 'Entries');
79:       Result := False;                 // no error
80:     end; {end else begin}
81:   end; {end with RxnDescrQy do }
82: end;
83:
84: // function to pass the component formula given its number
85: // if an error occurs in the lookup then 'Error' is passed and Error is 'true'
86: // *****************************************************************************
87: function TCompGenDM.GiveCompFormula(LkCompNo : Integer; var Error :Boolean )
88:    :String;
89: begin
90:   if CompGenQy.Lookup('CompNo',LkCompNo,'CForm') = NULL then begin
91:     MessageDlg ('No matching information for ' + IntToStr(LkCompNo) +
92:       ', please check component data.', mtError, [mbOK],0);
93:     Error := True;
94:     Result := 'Error';
95:   end {end if CompGenQy.Lookup('CompNo',LkCompNo,'CForm') = NULL }
96:   else begin
97:     // note that Error is not deemed false in case another error has occurred
98:     Result := CompGenQy.Lookup('CompNo',LkCompNo,'CForm');
99:   end; {end else for if CompGenQy.Lookup.... }
100: end; {end function TCompGenDM.GiveCompFormula(LkCompNo : Integer .... }
101:
102: // function to pass the reaction description given the number
103: // if an error occurs in the lookup then 'error' is passed and Error is 'true'
104: // *****************************************************************************
105: function TCompGenDM.GiveRxnDescr(LkRxnNo : Integer; var Error : Boolean)
106:    : String;
107: begin
108:   if RxnDescrQy.Lookup('RxnNo',LkRxnNo,'Descr') = NULL then begin
109:     MessageDlg ('No matching information for ' + IntToStr(LkRxnNo) +
110:       ', please reaction description data.', mtError, [mbOK],0);
111:     Error := True;
112:     Result := 'Error';
113:   end {end if RxnDescrQy.Lookup('RxnNo',LkRxnNo,'Descr') = NULL }
114:   else begin
115:     // note that Error is not deemed false in case another error has occurred
116:     Result := RxnDescrQy.Lookup('RxnNo',LkRxnNo,'Descr').AsString;
117:   end; {end else for if RxnDescrQy.Lookup('RxnNo',LkRxnNo,'Descr')... }
118: end; {end function TCompGenDM.GiveRxnDescr(LkRxnNo : Integer; }
119:
120: // function to pass the component Entropy (J/mol.K) given its number
121: // if an error occurs in the lookup then 'ERROR' is passed and Error is 'true'
122: // *****************************************************************************
123: function TCompGenDM.GiveCompEntropy(LkCompNo : Integer; var Error : Boolean)
124:    : String;
125: begin
126:   if (CompThermoQy.Lookup('CompNo',LkCompNo,'Entropy') = NULL) then begin
127:     MessageDlg ('No matching Entropy information for ' + IntToStr(LkCompNo) +
128:     ', check thermodynamic data.', mtError, [mbOK],0);
129:     Error := True;
130:     Result :='ERROR';
131:   end {end if (CompThermoTl.Lookup('CompNo',LkCompNo,'Entropy') = NULL) }
132:   else begin
133:     // note that Error is not deemed false in case another error has occurred
134:     Result := FloatToStrF(CompThermoQy.Lookup('CompNo',LkCompNo,'Entropy'),
135:      ffNumber,6,2);
136:   end; {end else for if (CompThermoTl.Lookup('CompNo',LkCompNo,'.... = NULL) }
137: end; {end function TCompGenDM.GiveCompEntropy(LkCompNo : Integer; var }
138:
139: // function to pass the component Gibbs F (kJ/mol) given its number
140: // if an error occurs in the lookup then 'error' is passed and Error is 'true'
141: // *****************************************************************************
142: function TCompGenDM.GiveCompGibbsF(LkCompNo : Integer; var Error : Boolean)
143:    : String;
144: begin
```

```
145:   if (CompThermoQy.Lookup('CompNo',LkCompNo,'Gibbs_F') = NULL) then begin
146:     MessageDlg ('No matching Gibbs Formation information for ' +
147:       IntToStr(LkCompNo) + ', check thermodynamic data.', mtError, [mbOK],0);
148:     Error := True;
149:     Result :='ERROR';
150:   end {end if (CompThermoTl.Lookup('CompNo',LkCompNo,'Gibbs_F') = NULL) }
151:   else begin
152:     // note that Error is not deemed false in case another error has occurred
153:     Result := FloatToStrF(CompThermoQy.Lookup('CompNo',LkCompNo,'Gibbs_F'),
154:      ffNumber,6,2);
155:   end; {end else for if (CompThermoTl.Lookup('CompNo',LkCompNo,'.... = NULL) }
156: end; {end function TCompGenDM.GiveCompGibbsF(LkCompNo : Integer; var }
157:
158: // procedure to set the RxnStoichQy.Filter to the selected reaction
159: // *************************************************************************
160: procedure TCompGenDM.SetRxnNoFilter (SelRxnNo : Integer);
161: var
162:   SearchOptions : TFilterOptions;
163: begin
164:   with CompGenDM.RxnStoichQy do begin
165:     SelectAllRxnStoich;
166:     // filter search options
167:     SearchOptions := [];    //foCaseInsensitive, foNoPartialCompare
168:     Filter := '[RxnNo] = '+IntToStr(SelRxnNo);
169:     FilterOptions := SearchOptions;
170:     Filtered := True;
171:   end; {end with CompGenDM.RxnStoichQy do }
172: end;  {end procedure TCompGenDM.SetRxnNoFilter (SelRxnNo : Integer); }
173:
174: // function to find first record where RxnNo matches filter
175: // need to have initialised RxnStoichQy.Filter with SetRxnNoFilter
176: // passes true if an error has occurred, calling function raises exception
177: // *************************************************************************
178: function TCompGenDM.FindFirstRxnStoich : Boolean;
179: begin
180:   if not RxnStoichQy.FindFirst then begin
181:     Result := True;
182:   end {end if not RxnStoichQy.FindFirst  }
183:   else  Result := False;
184: end; {end function TCompGenDM.FindFirstRxnStoich : Boolean; }
185:
186: // function to find next record where RxnNo matches filter
187: // need to have initialised RxnStoichQy.Filter with SetRxnNoFilter
188: // passes true if an error has occurred
189: // *************************************************************************
190: function TCompGenDM.FindNextRxnStoich : Boolean;
191: begin
192:   if not RxnStoichQy.FindNext then begin
193:     {MessageDlg ('No more stoichiometric records found.', mtInformation,
194:       [mbOK],0);  // redundant now because of the entries/counter comparison }
195:     Result := True;
196:   end {end if not RxnStoichQy.FindNext }
197:   else
198:     Result := False;
199: end; {end function TCompGenDM.FindNextRxnStoich : Boolean; }
200:
201: { procedure to select all records in RxnStoich table
202: ************************************************************************** }
203: procedure TCompGenDM.SelectAllRxnStoich;
204: begin
205:   with RxnStoichQy do begin
206:     Filtered := False;
207:     Close;
208:     SQL.Clear;
209:     SQL.Add('SELECT * FROM RXN_STOICH ORDER BY RXNNO');
210:     Open;
211:   end; {end with RxnStoichQy do }
212: end;  {end procedure TCompGenDM.SelectAllRxnStoich; }
213:
214: end.
```

```
 1: unit Constants;
 2:
 3: interface
 4:
 5: function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) : single;
 6:
 7: const
 8:   UnivGasConstant  : Single = 8.314;
 9:   CrissTRef        : Single = 298;                    // degrees K
10:   // minimum and maximum temperatures for Cp calculation]
11:   MinTemp          : Integer = 0;                     // °C
12:   MaxTemp          : Integer = 100;                   // °C
13:   TRef             : single = 298;                    // degrees K
14:
15: implementation
16:
17: { function to return the equilibrium constant at the conditions of the given
18:   Gibbs free energy (kJ/mol)
19: ************************************************************************** }
20: function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) : single;
21: begin
22:   Result := exp((-GibbsRxn * 1000 )/(UnivGasConstant * TemperatureK));
23: end; {end function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) ... }
24:
25:
26: end.
```

```
 1: unit Constants;
 2:
 3: interface
 4:
 5: function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) : single;
 6:
 7: const
 8:   UnivGasConstant  : Single = 8.314;
 9:   CrissTRef        : Single = 298;                    // degrees K
10:   // minimum and maximum temperatures for Cp calculation]
11:   MinTemp          : Integer = 0;                     // °C
12:   MaxTemp          : Integer = 100;                   // °C
13:   TRef             : single = 298;                    // degrees K
14:
15: implementation
16:
17: { function to return the equilibrium constant at the conditions of the given
18:   Gibbs free energy (kJ/mol)
19: ********************************************************************************* }
20: function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) : single;
21: begin
22:   Result := exp((-GibbsRxn * 1000 )/(UnivGasConstant * TemperatureK));
23: end; {end function ReturnKRxnforGibbsRxn (GibbsRxn, TemperatureK : Single) ... }
24:
25:
26: end.
```

```
unit DisplayRunningDataFormUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Grids, DBGrids, ComCtrls, ExtCtrls, Buttons;

type
  TDisplayRunningDataForm = class(TForm)
    GBRunning: TGroupBox;
    DBGridRunning: TDBGrid;
    Panel1: TPanel;
    Label2: TLabel;
    CBRunning: TComboBox;
    SBRunning: TStatusBar;
    BtnRead: TSpeedButton;
    BtnEdit: TSpeedButton;
    BtnDelete: TSpeedButton;
    BtnPost: TSpeedButton;
    BtnAdd: TSpeedButton;
    BtnClose: TSpeedButton;
    procedure CBRunningChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BtnCloseClick(Sender: TObject);
    procedure BtnReadClick(Sender: TObject);
    procedure BtnEditClick(Sender: TObject);
    procedure BtnDeleteClick(Sender: TObject);
    procedure BtnPostClick(Sender: TObject);
    procedure BtnAddClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  DisplayRunningDataForm: TDisplayRunningDataForm;

implementation

uses RunningDMUnit;

{$R *.DFM}
{ this procedure ensures that the DataGrid displays the appropriate information
  as indicated by the combo box text.
********************************************************************************* }
procedure TDisplayRunningDataForm.CBRunningChange(Sender: TObject);
begin
  // combobox reading indicated by ItemIndex property.
  with CBRunning do begin
    case ItemIndex of
      0 :  begin                  // Selected Compounds (CompConc)
        DBGridRunning.DataSource := RunningDM.CompConcDS;
        GBRunning.Caption := CBRunning.Items[ItemIndex];
      end; {end 0 : }
      1 :  begin                  // Selected Reactions (Rxn_Info)
        DBGridRunning.DataSource := RunningDM.RxnInfoDS;
        GBRunning.Caption := CBRunning.Items[ItemIndex];
      end; {end  1 : }
      2 : begin                   // Calculation Output (CalcOutput)
        DBGridRunning.DataSource := RunningDM.CalcOutputDS;
        GBRunning.Caption := CBRunning.Items[ItemIndex];
      end; {end 2 : }
    else begin
      ShowMessage (' Please select a table from the list. ' +
       'Display set to Reaction Information.');
      GBRunning.Caption := CBRunning.Items[1];;
      DBGridRunning.DataSource := RunningDM.RxnInfoDS;

    end; {end else for case ItemIndex of}
    end; {end case ItemIndex of }
```

```
  end; {end with CBRunning do }

end;  {end procedure TDisplayRunningDataForm.ComboBox1Change(Sender: TObject); }

{  Set DBGrid to default... RxnInfo Table
********************************************************************************* }
procedure TDisplayRunningDataForm.FormCreate(Sender: TObject);
begin
  with CBRunning do begin
    ItemIndex := 1;
    Text := Items[ItemIndex];
    GBRunning.Caption := Items[ItemIndex];;
  end; {end with CBRunning do }
  DBGridRunning.DataSource := RunningDM.RxnInfoDS;
end; {end procedure TDisplayRunningDataForm.FormCreate(Sender: TObject);}
{ close DisplayRunningDataForm
********************************************************************************* }
procedure TDisplayRunningDataForm.BtnCloseClick(Sender: TObject);
begin
  DisplayRunningDataForm.Hide;
end; {end procedure TDisplayRunningDataForm.BtnCloseClick(Sender: TObject); }

{ READ ONLY MODE Switch current data source to AutoEdit := False
********************************************************************************* }
procedure TDisplayRunningDataForm.BtnReadClick(Sender: TObject);
begin
  DBGridRunning.DataSource.AutoEdit := False;
  BtnRead.Enabled := False;
  BtnEdit.Enabled := True;
  SBRunning.Panels[1].Text:= 'Read only.';
end; {end procedure TDisplayRunningDataForm.BtnReadClick(Sender: TObject); }

{ EDIT MODE Switch current data source to AutoEdit := True
********************************************************************************* }
procedure TDisplayRunningDataForm.BtnEditClick(Sender: TObject);
begin
  DBGridRunning.DataSource.AutoEdit := True;
  BtnRead.Enabled := True;
  BtnEdit.Enabled := False;
  SBRunning.Panels[1].Text:= 'Edit.';
end; {end procedure TDisplayRunningDataForm.BtnEditClick(Sender: TObject); }

{ Delete the current record, with confirmation
  puts the datasource in read only mode afterwards
********************************************************************************* }
procedure TDisplayRunningDataForm.BtnDeleteClick(Sender: TObject);
begin
  // confirmation
  if MessageDlg ('Are you sure you want to delete the current record?',
   mtConfirmation, [mbYES, mbNO],0) = mrYES then begin
    SBRunning.Panels[1].Text:= 'Deleting current record.';
    // get the active query object from the datasource...
    with DBGridRunning.DataSource.DataSet do begin
      if not active then Open;
      Delete;
    end; {end with DBGridRunning.DataSource.DataSet do begin }
    // switch to read only mode...
    DBGridRunning.DataSource.AutoEdit := False;
    BtnRead.Enabled := False;
    BtnEdit.Enabled := True;
    SBRunning.Panels[1].Text:= 'Read only.';
  end; {end if ModalResult = mrYES then begin }
end; {end procedure TDisplayRunningDataForm.BtnDeleteClick(Sender: TObject); }

{ Post the changes to the dataset
  automatically puts the table into read only mode
********************************************************************************* }
procedure TDisplayRunningDataForm.BtnPostClick(Sender: TObject);
begin
  with DBGridRunning.DataSource do begin
    // first check if the dataset is in edit mode.
```

```
145:      if AutoEdit = True then begin
146:        with DataSet do begin
147:          Post;
148:          Close;
149:          Open;
150:        end; {end with DataSet do begin }
151:      end {end if Autoedit = True then begin }
152:      else {else for if Autoedit = True then begin }
153:        ShowMessage('Data set not in edit mode.');
154:    end; {end with DBGridRunning.DataSource.DataSet do begin }
155:    // put the datasource into read only mode
156:    DBGridRunning.DataSource.AutoEdit := False;
157:    BtnRead.Enabled := False;
158:    BtnEdit.Enabled := True;
159:    SBRunning.Panels[1].Text:= 'Read only.';
160: end;  {end procedure TDisplayRunningDataForm.BtnPostClick(Sender: TObject); }
161:
162: { add a record into the table currently displayed on the data grid
163:   automatically puts the grid\table in autoedit mode
164: ********************************************************************************* }
165: procedure TDisplayRunningDataForm.BtnAddClick(Sender: TObject);
166: begin
167:    // get the active query object from the datasource...
168:    with DBGridRunning.DataSource.DataSet do begin
169:      if not active then Open;
170:      Append;
171:    end; {end with DBGridRunning.DataSource.DataSet do begin }
172:    // switch to edit mode...
173:    DBGridRunning.DataSource.AutoEdit := True;
174:    BtnRead.Enabled := True;
175:    BtnEdit.Enabled := False;
176:    SBRunning.Panels[1].Text:= 'Edit.';
177: end; {end procedure TDisplayRunningDataForm.BtnAddClick(Sender: TObject); }
178:
179:
180: end.
```

```
 1: unit DisplaySpeciesDataFormUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   StdCtrls, Grids, DBGrids, ComCtrls, ExtCtrls, Buttons, DBTables;
 8:
 9: type
10:   TDisplaySpeciesDataForm = class(TForm)
11:     GBSpecies: TGroupBox;
12:     DBGridSpecies: TDBGrid;
13:     Panel1: TPanel;
14:     Label2: TLabel;
15:     CBSpecies: TComboBox;
16:     BtnClose: TSpeedButton;
17:     BtnAdd: TSpeedButton;
18:     BtnPost: TSpeedButton;
19:     BtnDelete: TSpeedButton;
20:     BtnEdit: TSpeedButton;
21:     SBSpecies: TStatusBar;
22:     BtnRead: TSpeedButton;
23:     procedure CBSpeciesChange(Sender: TObject);
24:     procedure FormCreate(Sender: TObject);
25:     procedure BtnCloseClick(Sender: TObject);
26:     procedure BtnEditClick(Sender: TObject);
27:     procedure BtnReadClick(Sender: TObject);
28:     procedure BtnAddClick(Sender: TObject);
29:     procedure BtnPostClick(Sender: TObject);
30:     procedure BtnDeleteClick(Sender: TObject);
31:   private
32:     { Private declarations }
33:   public
34:     { Public declarations }
35:   end;
36:
37: var
38:   DisplaySpeciesDataForm: TDisplaySpeciesDataForm;
39:
40: implementation
41:
42: uses CompGenDMUnit;
43:
44: {$R *.DFM}
45: { this procedure ensures that the DataGrid displays the appropriate information
46:   as indicated by the combo box text.
47: ********************************************************************************* }
48: procedure TDisplaySpeciesDataForm.CBSpeciesChange(Sender: TObject);
49: begin
50:   // combobox reading indicated by ItemIndex property.
51:   with CBSpecies do begin
52:     case ItemIndex of
53:       0 :  begin              // General Compound Information (CompGen)
54:         DBGridSpecies.DataSource := CompGenDM.CompGenDS;
55:         GBSpecies.Caption := CBSpecies.Items[ItemIndex];
56:       end; {end 0 : }
57:       1 :  begin              // Compound Thermodynamic Data (comp_thermo)
58:         DBGridSpecies.DataSource := CompGenDM.CompThermoDS;
59:         GBSpecies.Caption := CBSpecies.Items[ItemIndex];
60:       end; {end  1 : }
61:       2 :  begin              // Heat Capacity Data (cp_data)
62:         DBGridSpecies.DataSource := CompGenDM.CpDataDS;
63:         GBSpecies.Caption := CBSpecies.Items[ItemIndex];
64:       end; {end 2 : }
65:       3 :  begin              //  State Description (state_descr)
66:         DBGridSpecies.DataSource := CompGenDM.StateDS;
67:         GBSpecies.Caption := CBSpecies.Items[ItemIndex];
68:       end; {end 3 : }
69:       4 :  begin              // Reference Description (ref_descr)
70:         DBGridSpecies.DataSource := CompGenDM.RefDescrDS;
71:         GBSpecies.Caption := CBSpecies.Items[ItemIndex];
72:       end; {end 4 : }
```

```
      5 :  begin                  // Reaction Stoichiometry (rxn_stoich)
        DBGridSpecies.DataSource := CompGenDM.RxnStoichDS;
        GBSpecies.Caption := CBSpecies.Items[ItemIndex];
      end; {end 5 : }
      6 :  begin                  //  Reaction Description (rxn_descr)
        DBGridSpecies.DataSource := CompGenDM.RxnDescrDS;
        GBSpecies.Caption := CBSpecies.Items[ItemIndex];
      end; {end 6 : }
      7 :  begin                  // Criss Cobble Code Description
        DBGridSpecies.DataSource := CompGenDM.CrissDescrDS;
        GBSpecies.Caption := CBSpecies.Items[ItemIndex];
      end; {end 7 : }
      8 :  begin                  // Criss Cobble Constants
        DBGridSpecies.DataSource := CompGenDM.CrissCobbleDS;
        GBSpecies.Caption := CBSpecies.Items[ItemIndex];
      end; {end 8 : }
    else begin
      ShowMessage (' Please select a table from the list. ' +
       'Display set to general compounds.');
      GBSpecies.Caption := CBSpecies.Items[0];;
      DBGridSpecies.DataSource := CompGenDM.CompGenDS;

    end; {end else for case ItemIndex of}
    end; {end case ItemIndex of }


  end; {end with CBSpecies do }


end;  {end procedure TDisplaySpeciesDataForm.ComboBox1Change(Sender: TObject); }


{  Set DBGrid to default... CompGen Table
********************************************************************************* }
procedure TDisplaySpeciesDataForm.FormCreate(Sender: TObject);
begin
  with CBSpecies do begin
    ItemIndex := 0;
    Text := Items[ItemIndex];
    GBSpecies.Caption := Items[ItemIndex];;
  end; {end with CBSpecies do }
  DBGridSpecies.DataSource := CompGenDM.CompGenDS;
end; {end procedure TDisplaySpeciesDataForm.FormCreate(Sender: TObject);}

{ close DisplaySpeciesDataForm
********************************************************************************* }
procedure TDisplaySpeciesDataForm.BtnCloseClick(Sender: TObject);
begin
  DisplaySpeciesDataForm.Hide;    // Main form gets focus
end; {end procedure TDisplaySpeciesDataForm.BtnCloseClick(Sender: TObject); }

{ EDIT MODE Switch current data source to AutoEdit := True
********************************************************************************* }
procedure TDisplaySpeciesDataForm.BtnEditClick(Sender: TObject);
begin
  DBGridSpecies.DataSource.AutoEdit := True;
  BtnRead.Enabled := True;
  BtnEdit.Enabled := False;
  SBSpecies.Panels[1].Text:= 'Edit.';
end; {end procedure TDisplaySpeciesDataForm.BtnEditClick(Sender: TObject); }

{ READ ONLY MODE Switch current data source to AutoEdit := False
********************************************************************************* }
procedure TDisplaySpeciesDataForm.BtnReadClick(Sender: TObject);
begin
  DBGridSpecies.DataSource.AutoEdit := False;
  BtnRead.Enabled := False;
  BtnEdit.Enabled := True;
  SBSpecies.Panels[1].Text:= 'Read only.';
end; {end procedure TDisplaySpeciesDataForm.BtnReadClick(Sender: TObject); }

{ add a record into the table currently displayed on the data grid
```

```
145:   automatically puts the grid\table in autoedit mode
146: ********************************************************************************* }
147: procedure TDisplaySpeciesDataForm.BtnAddClick(Sender: TObject);
148: begin
149:   // get the active query object from the datasource...
150:   with DBGridSpecies.DataSource.DataSet do begin
151:     if not active then Open;
152:     Append;
153:   end; {end with DBGridSpecies.DataSource.DataSet do begin }
154:   // switch to edit mode...
155:   DBGridSpecies.DataSource.AutoEdit := True;
156:   BtnRead.Enabled := True;
157:   BtnEdit.Enabled := False;
158:   SBSpecies.Panels[1].Text:= 'Edit.';
159: end; {end procedure TDisplaySpeciesDataForm.BtnAddClick(Sender: TObject); }
160:
161: { Post the changes to the dataset
162:   automatically puts the table into read only mode
163: ********************************************************************************* }
164: procedure TDisplaySpeciesDataForm.BtnPostClick(Sender: TObject);
165: begin
166:   with DBGridSpecies.DataSource do begin
167:     // first check if the dataset is in edit mode.
168:     if AutoEdit = True then begin
169:       with DataSet as TQuery do
170:         if not RequestLive then RequestLive:= True;
171:       with DataSet do begin
172:         Post;
173:         Close;
174:         Open;
175:       end; {end with DataSet do begin }
176:     end {end if Autoedit = True then begin }
177:     else {else for if Autoedit = True then begin }
178:       ShowMessage('Data set not in edit mode.');
179:   end; {end with DBGridSpecies.DataSource.DataSet do begin }
180:   // put the datasource into read only mode
181:   DBGridSpecies.DataSource.AutoEdit := False;
182:   BtnRead.Enabled := False;
183:   BtnEdit.Enabled := True;
184:   SBSpecies.Panels[1].Text:= 'Read only.';
185: end;  {end procedure TDisplaySpeciesDataForm.BtnPostClick(Sender: TObject); }
186:
187: { Delete the current record, with confirmation
188:   puts the datasource in read only mode afterwards
189: ********************************************************************************* }
190: procedure TDisplaySpeciesDataForm.BtnDeleteClick(Sender: TObject);
191: begin
192:   // confirmation
193:   if MessageDlg ('Are you sure you want to delete the current record?',
194:    mtConfirmation, [mbYES, mbNO],0) = mrYES then begin
195:     SBSpecies.Panels[1].Text:= 'Deleting current record.';
196:     // get the active query object from the datasource...
197:     with DBGridSpecies.DataSource.DataSet do begin
198:       if not active then Open;
199:       Delete;
200:     end; {end with DBGridSpecies.DataSource.DataSet do begin }
201:     // switch to read only mode...
202:     DBGridSpecies.DataSource.AutoEdit := False;
203:     BtnRead.Enabled := False;
204:     BtnEdit.Enabled := True;
205:     SBSpecies.Panels[1].Text:= 'Read only.';
206:   end; {end if ModalResult = mrYES then begin }
207: end; {end procedure TDisplaySpeciesDataForm.BtnDeleteClick(Sender: TObject); }
208:
209: end.
```

```
 1: unit Exceptns;
 2:
 3: {unit containing all additional exception information }
 4:
 5: interface
 6: uses
 7:   SysUtils;
 8:
 9: type
10:   EDataError          = class (Exception);
11:   ECompFormNotFound   = class (EDataError);
12:   ECompNoInvalid      = class (EDataError);
13:   EDataNotExist       = class (EDataError);
14:   EStoichDataError    = class (EDataError);
15:   ECrissDataError     = class (EDataError);
16:
17:   ERxnError           = class (EDataError);
18:   ERxnNoInvalid       = class (ERxnError);
19:   ERxnNotDescribed    = class (ErxnError);
20:   ERxnNoStoichData    = class (ERxnError);
21:
22:   ECrissError         = class (EDataError);
23:   ECpError            = class (EDataError);
24:   ENotIonicError      = class (ECrissError);
25:
26:   EChartError         = class (EDataError);
27:
28:
29: implementation
30:
31:
32: end.
```

```
 1: unit GibbsCalcClassUnit;
 2:
 3: interface
 4: uses
 5:   CompClassUnit, RxnClassUnit, GibbsCalcFormUnit;
 6:
 7: type
 8:   { class which controls the Gibbs free energy calculation process ... }
 9:   TGibbsCalcClass = class (TDescription)
10:   private
11:     TemperatureK        : single;              // container for the temperature
12:     TempCounter         : integer;             // counter for the temperature
13:     InitTempC,
14:     FinalTempC          : single;
15:     NumberCalcs         : integer;
16:     Interval            : single;
17:     SelRxnNo            : integer;
18:     RxnDetails          : TRxnDetails;
19:     CompoundDetails     : TCompDetails;
20:     GibbsRxnT           : single;
21:     KRxnT               : double;
22:
23:     procedure GetRxnDetails;
24:
25:   public
26:     constructor Create ( text : string);
27:     destructor Destroy;
28:     procedure PassTemperatures (Initial, Final : single; Points : integer);
29:     procedure GetCurrentRxnNo(SelRxnNo : integer);
30:     procedure CalcGibbsGivenRxnNo (SelRxnNo : integer);
31:
32:   end; {end TGibbsCalcClass = class (TObject) }
33:
34: implementation
35:
36: uses
37:    Dialogs, Controls, Exceptns, Windows, Messages, SysUtils, Classes, Graphics,
38:     Forms, Menus, ComCtrls, Math,
39:    RunningDMUnit, CompGenDMUnit, SelectionFormUnit, Constants,
40:     GibbsCalcProgressUnit;
41:
42: { constructor for TGibbsCalcClass
43: ********************************************************************************* }
44: constructor TGibbsCalcClass.Create (text : string);
45: begin
46:   // instantiate and initialise TDecription object (ancestor)
47:   TDescription.Create (text);
48:   // initialise variables and constant
49:   InitTempC    := 0;
50:   FinalTempC   := 0;
51:   Interval     := 0;
52:   NumberCalcs  := 0;
53:   TempCounter  := 0;
54:  end; {end constructor TGibbsCalcClass.Create }
55:
56: { procedure to calculate the Gibbs free energy for the given reaction and at
57:   the given temperature
58: ********************************************************************************* }
59: procedure TGibbsCalcClass.CalcGibbsGivenRxnNo(SelRxnNo : integer);
60: var
61:   CpMeanT          : single;                  // mean heat capacity
62:   CpMeanTRxn       : single;                  // mean heat capacity for the rxn
63:   GibbsRxnStd      : single;                  // std Gibbs change for Rxn (298K)
64:   EntropyRxnStd    : single;                  // std Entropy change for Rxn
65:   logKRxnT         : single;                  // log of equilibrium constant
66:
67: begin
68:     // get reaction number
69:     GetCurrentRxnNo (SelRxnNo);              // note 'RxnDetails' is created in this
    loop
70:     // instantiate compound information container
71:     CompoundDetails := TCompDetails.Create;
```

Classes\GibbsCalcClassUnit.pas

```
72:     TempCounter := 0;                                   // zero counter to be certain
73:     if GibbsCalcForm.PointSeries = 4 then
74:       GibbsCalcForm.PointSeries := 0;                   // reset Point
      Series
75:     inc(GibbsCalcForm.PointSeries,1);
76:
77:     // initialise GibbsCalcForm.PointSeries1 (better implemented with a pointer)
78:     with GibbsCalcForm do begin
79:       case PointSeries of
80:         1: begin
81:           PointOne.Clear;
82:           PointOne.XValues.DateTime := False;
83:           PointOne.Title := 'Reaction: ' +IntToStr(SelRxnNo);
84:         end; {end 1: }
85:         2: begin
86:           PointTwo.Clear;
87:           PointTwo.XValues.DateTime := False;
88:           PointTwo.Title := 'Reaction: ' +IntToStr(SelRxnNo);
89:         end; {end 2: }
90:         3: begin
91:           PointThree.Clear;
92:           PointThree.XValues.DateTime := False;
93:           PointThree.Title := 'Reaction: ' +IntToStr(SelRxnNo);
94:         end; {end 3: }
95:         4: begin
96:           PointFour.Clear;
97:           PointFour.XValues.DateTime := False;
98:           PointFour.Title := 'Reaction: ' +IntToStr(SelRxnNo);
99:         end; {end 4: }
100:        else raise EChartError.Create ('Please reset the application.');
101:      end; {end case PointSeries of }
102:    end; {end with GibbsCalcForm do begin }
103:
104:    while TempCounter < (NumberCalcs-1) do begin       // temperature loop
105:      // get temperature from interval, counter and initial temperature
106:      TemperatureK := (InitTempC + 273.15) + TempCounter*Interval;
107:      RxnDetails.Zeroicomp;          // reset the RxnDetails counter
108:      CpMeanTRxn      := RxnDetails.PassCpMeanTRxn (TemperatureK);
109:      GibbsRxnStd     := StrToFloat(RxnDetails.PassGibbsRxnStdStr);
110:      EntropyRxnStd   := StrToFloat(RxnDetails.PassEntropyRxnStdStr);
111:      // calculate Gibbs free energy change for given reaction at T (kJ/mol)
112:      GibbsRxnT       := (GibbsRxnStd*1000 - (TemperatureK-TRef)*EntropyRxnStd+
113:                          (TemperatureK-TRef)*CpMeanTRxn-
114:                          TemperatureK*ln(TemperatureK/TRef)* CpMeanTRxn)/1000;
115:      // get equilibrium constant given GibbsRxnT
116:      KRxnT           := ReturnKRxnforGibbsRxn (GibbsRxnT, TemperatureK);
117:      logKRxnT        := log10(KRxnT);
118:      // store mean heat capacity for the reaction
119:      { procedure TRunningDM.StoreCalcOutput(RxnNo : integer; TempK : single;
120:           CpMeanT, GibbsRxnT, logKRxnT : single; KRxnT : Double); }
121:      RunningDM.StoreCalcOutput(
122:       SelRxnNo, TemperatureK, CpMeanTRxn, GibbsRxnT, logKRxnT, KRxnT);
123:
124:      // output to graph - PointSeries decided by 'PointSeries'
125:      with GibbsCalcForm do begin
126:        case PointSeries of
127:          1: PointOne.AddXY(TemperatureK-273.15, logKRxnT,'',clTeeColor);
128:          2: PointTwo.AddXY(TemperatureK-273.15, logKRxnT,'',clTeeColor);
129:          3: PointThree.AddXY(TemperatureK-273.15, logKRxnT,'',clTeeColor);
130:          4: begin
131:            PointFour.AddXY(TemperatureK-273.15, logKRxnT,'',clTeeColor);
132:          end; {end 4: }
133:        end; {end case PointSeries of }
134:      end; {end with GibbsCalcForm do begin }
135:
136:      inc(TempCounter,1);    // increment counter by 1
137:      GibbsCalcProgress.StepTempProgressBar(Round((TempCounter/NumberCalcs))*100);
138:    end; {end   while counter < NumberCalcs do begin }
139: end; {end procedure TGibbsCalcClass.CalcGibbsGivenTandRxnNo; }
140:
141: { procedure to pass the initial and final temperature, and the number of
142:   calculations which the enduser wishes to process
```

```
143: ******************************************************************************* }
144: procedure TGibbsCalcClass.PassTemperatures (Initial, Final : single; Points :
145:  integer);
146: begin
147:   // assign values to private variables
148:   InitTempC   := Initial;
149:   FinalTempC  := Final;
150:   NumberCalcs := Points;
151:   Interval    := (FinalTempC-InitTempC)/NumberCalcs;
152: end; {end procedure TGibbsCalcClass.PassTemperatures (Initial, Final : ... }
153:
154: { procedure to retrieve current reaction number from current position in RxnInfo
155:   instantiates RxnDetails object
156: ******************************************************************************* }
157: procedure TGibbsCalcClass.GetCurrentRxnNo(SelRxnNo : integer);
158: var
159:   SQLStr : string;                                    // eSQL container string
160: begin
161:   SQLStr := 'SELECT * FROM CALCOUTPUT WHERE RXNNO = "%s";';
162:   // check whether data exists in CalcOutputQy for this reaction..
163:   with RunningDM.CalcOutputQy do begin
164:     SQL.Clear;
165:     SQL.Add ('SELECT * FROM CALCOUTPUT');  // need to 'reset' the SQL set
166:     if not Active then Open;
167:     if Locate('RXNNO', SelRxnNo,[loCaseInsensitive]) then
168:       if MessageDlg ('Calculation information for reaction ' +
169:        CompGenDM.RxnDescrQy.Lookup('RXNNO', SelRxnNo, 'DESCR') +
170:        'already exists.  Recalculate or abort calculation? ', mtConfirmation,
171:        [mbOK, mbCancel], 0) = mrOK then
172:        begin
173:          ShowMessage ('Deleting previous CalcOutput for this reaction.');
174:          with RunningDM.CalcOutputQy do begin
175:            if Active then Close;
176:            if RequestLive then RequestLive := False;
177:            SQL.Clear;
178:            SQL.Add (Format(SQLStr, [ IntToStr(SelRxnNo)]));
179:            ExecSQL;
180:            SQL.Clear;
181:            SQL.Add ('SELECT * FROM CALCOUTPUT ORDER BY TEMPK');
182:            if not Active then Open;
183:            if not RequestLive then RequestLive := True;
184:          end; {end with RunningDM.CalcOutputQy do begin}
185:        end {end if MessageDlg ('Calculation information for react'... = mrOK}
186:        else {else for if MessageDlg ('Calculation information for '... = mrOK}
187:          raise ERxnError.Create('Calculation aborted by user.');
188:     end; {end   with RunningDM.CalcOutputQy do begin }
189:     // refresh the GibbsCalcProgress information dialog
190:     GibbsCalcProgress.Refresh;
191:     // instantiate TRxnDetails object
192:     RxnDetails := TRxnDetails.Create (SelRxnNo);
193: end; {end procedure TGibbsCalcClass.GetCurrentRxnNo; }
194:
195: { procedure to instantiate the TRxnDetails object, and retrieve comp information
196: ******************************************************************************* }
197: procedure TGibbsCalcClass.GetRxnDetails;
198: begin
199:   // instantiate RxnDetails object
200:   // TCompDetails.Create (DescripStr, UserID :string; CompNo : integer);
201:   RxnDetails := TRxnDetails.Create (SelRxnNo);
202:
203: end; {end procedure TGibbsCalcClass.GetRxnDetails; }
204:
205: { destructor for TGibbsCalcClass
206: ******************************************************************************* }
207: destructor TGibbsCalcClass.Destroy;
208: begin
209:   RxnDetails.Free;
210:   CompoundDetails.Free;
211: end; {end destructor TGibbsCalcClass.Destroy; }
212:
213: end.
```

GibbsCalcFormUnit.pas

```
unit GibbsCalcFormUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
   StdCtrls, ComCtrls, Teengine, Chart, Buttons, ExtCtrls, Grids, DBGrids,
  RxnClassUnit, Series;

type
  TGibbsCalcForm = class(TForm)
    SBGibbsCalcForm: TStatusBar;
    PlGibbsCalcForm: TPanel;
    BtnClose: TSpeedButton;
    PlGibbsChart: TPanel;
    BtnRxnSelect: TSpeedButton;
    PlSelection: TPanel;
    GBSharedVariables: TGroupBox;
    Label3: TLabel;
    Label4: TLabel;
    EBTempInitial: TEdit;
    EBTempFinal: TEdit;
    EBPoints: TEdit;
    PCGibbsCalcForm: TPageControl;
    TSReactions: TTabSheet;
    TSCompound: TTabSheet;
    GroupBox1: TGroupBox;
    RGReactions: TRadioGroup;
    BtnGo: TSpeedButton;
    PCChartMemo: TPageControl;
    TSChart: TTabSheet;
    GibbsChart: TChart;
    TSMemo: TTabSheet;
    MmGibbsCalcForm: TMemo;
    DBGridGibbsCalcForm: TDBGrid;
    BtnClearMemo: TSpeedButton;
    BtnSaveMemo: TSpeedButton;
    SaveDialog1: TSaveDialog;
    BtnOpenMemo: TSpeedButton;
    OpenDialog1: TOpenDialog;
    TSCalcOutput: TTabSheet;
    BtnClear: TSpeedButton;
    PointOne: TPointSeries;
    PointTwo: TPointSeries;
    PointThree: TPointSeries;
    PointFour: TPointSeries;
    Label1: TLabel;
    DBGridCalcOutput: TDBGrid;
    PrintDialog1: TPrintDialog;
    PrinterSetupDialog1: TPrinterSetupDialog;
    BtnPrintMemo: TSpeedButton;
    procedure BtnCloseClick(Sender: TObject);
    procedure BtnRxnSelectClick(Sender: TObject);
    procedure BtnGoClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BtnSaveMemoClick(Sender: TObject);
    procedure BtnClearMemoClick(Sender: TObject);
    procedure BtnOpenMemoClick(Sender: TObject);
    procedure PCChartMemoChange(Sender: TObject);
    procedure BtnClearClick(Sender: TObject);
    procedure BtnPrintMemoClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    PointSeries : integer;

    procedure OutputToMemo  (const RxnDetails : TRxnDetails);
  end;

var
  GibbsCalcForm: TGibbsCalcForm;
```

```
 73:
 74: implementation
 75:
 76: uses
 77:   SelectionFormUnit, RunningDMUnit, GibbsCalcClassUnit, UnitsCheckUnit,
 78:    MainFormUnit,GibbsCalcProgressUnit, CompGenDMUnit, Exceptns,
 79:   Math, DBTables, Printers ;
 80:
 81: {$R *.DFM}
 82:
 83: { procedure to pass salient information to the Memo MmGibbsCalcForm
 84: ********************************************************************** }
 85: procedure TGibbsCalcForm.OutputToMemo (const RxnDetails : TRxnDetails);
 86: var
 87:   SQLStr    : string;                  // eSQL container string
 88:   TitleStr,
 89:   UnitStr,
 90:   OutputStr : string;                  // output to memo format string
 91:   TempK,
 92:   GibbsRxnT,
 93:   CpMeanT,
 94:   LogKRxnT  : single;
 95:   KRxnT     : double;
 96:
 97: begin
 98:   with MmGibbsCalcForm do begin
 99:     Lines.Add ('User name:                ' + MainForm.User.PassUserID);
100:     Lines.Add ('Date of calculation:      ' + DateToStr(Date));
101:     Lines.Add ('Time of calculation:      ' + TimeToStr(Time));
102:     Lines.Add ('Reaction number:          ' + RxnDetails.PassRxnNoStr);
103:     Lines.Add ('Reaction description:     ' + RxnDetails.PassRxnDescrStr);
104:     Lines.Add ('Std Entropy:              ' + RxnDetails.PassEntropyRxnStdStr);
105:     Lines.Add ('Std Gibbs free energy :   ' + RxnDetails.PassGibbsRxnStdStr);
106:     Lines.Add ('Equilibrium const. (Std): ' + RxnDetails.PassKGibbsRxnStdStr);
107:     Lines.Add ('log K (above):            ' + FloatToStrF(
108:      log10 (StrToFloat(RxnDetails.PassKGibbsRxnStdStr)), ffGeneral, 5, 7));
109:     Lines.Add ('*************************************************************');
110:   end; {end withMmGibbsCalcForm do begin }
111:
112:   // grab the calc output from the database table, prepare string formats
113:   TitleStr  := '     T    | Cp Rxn   |   dG Rxn |       K Rxn  |  log K  |';
114:   UnitStr   := '     K    | J/mol·K  |   kJ/mol |              |         ';
115:   {            "1234567890|12345678901|1234567890|123456789012345|123456789|'  }
116:   OutputStr := '   %6.2f | %9.3f | %8.3f | %12.3g | %8.3f ';
117:   //            123456789|123456789|123456789|123456789|123456789|
118:   // firstly select the necessary records from the database
119:   with RunningDM.CalcOutputQy do begin
120:     if Active then Close;
121:     if RequestLive then RequestLive := False;
122:     SQLStr := 'SELECT * FROM CALCOUTPUT WHERE RXNNO = "%s" ORDER BY TEMPK';
123:     SQL.Clear;
124:     SQL.Add(Format(SQLStr, [RxnDetails.PassRxnNoStr]));
125:     Open;
126:     if not Active then Open;
127:     if not RequestLive then RequestLive := True;
128:   end; {end with RunningDM.CalcOutputQy do begin }
129:
130:   // now run through the database records and add them to the memo
131:   with RunningDM do begin
132:     // move to the first record
133:     if not CalcOutputQy.FindFirst then
134:       raise EDataError.Create
135:        ('Calculation output data not found, rerun calculation.');
136:     with MmGibbsCalcForm.Lines do begin
137:       Add ('Calculation Output for reaction '+ RxnDetails.PassRxnNoStr+'.');
138:       Add ('');
139:       Add (TitleStr);
140:       Add (UnitStr);
141:       Add ('_____________________________________________________________');
142:     end; {end     with MmGibbsCalcForm.Lines do begin }
143:
144:     // retrieve the first records
```

```
145:     RetrieveCurrentCalcOutput( TempK, CpMeanT, GibbsRxnT, logKRxnT, KRxnT );
146:
147:     MmGibbsCalcForm.Lines.Add (Format (OutputStr,[TempK, CpMeanT, GibbsRxnT,
148:      KRxnT, logKRxnT]));
149:
150:     {
151:        TitleStr  := '    T    | Cp Rxn  |  dG Rxn |  K Rxn  |  log K  |';
152:        UnitStr   := '    K    | J/mol·K |  kJ/mol |         |         |'; }
153:
154:     // move through the rest of the records
155:     while not CalcOutputQy.EOF do begin                    // EOF = end of file
156:       // move to next record
157:       CalcOutputQy.Next;
158:       RetrieveCurrentCalcOutput( TempK, CpMeanT, GibbsRxnT, logKRxnT, KRxnT );
159:       MmGibbsCalcForm.Lines.Add (Format (OutputStr,[TempK, CpMeanT, GibbsRxnT,
160:        KRxnT, logKRxnT]));
161:     end; {end while not EOF do begin }
162:   end; {end with RunningDM do begin }
163:
164:   MmGibbsCalcForm.Lines.Add
165:    ('************************************************************');
166:
167: end; {end procedure TGibbsCalcForm.OutputToMemo }
168:
169: { close the window and pass active focus to the main form
170: ************************************************************************** }
171: procedure TGibbsCalcForm.BtnCloseClick(Sender: TObject);
172: begin
173:   GibbsCalcForm.Hide;
174: end; {end procedure TGibbsCalcForm.BtnCloseClick(Sender: TObject); }
175:
176: { procedure to show SelectionForm - in order to work with Rxn Info and CompConc
177: ************************************************************************** }
178: procedure TGibbsCalcForm.BtnRxnSelectClick(Sender: TObject);
179: begin
180:   SelectionForm.Show;
181: end; {end procedure TGibbsCalcForm.SpeedButton1Click(Sender: TObject); }
182:
183: { procedure to start Gibbs calculation
184: ************************************************************************** }
185: procedure TGibbsCalcForm.BtnGoClick(Sender: TObject);
186: var
187:   InitTemp    : Single;                  // initial temperature
188:   FinalTemp   : Single;                  // final temperature
189:   NumberCalcs : Integer;                 // number of calculations \ points
190:   GibbsCalc   : TGibbsCalcClass;         // class containing calc methods
191:   Check       : TCheckUnits;             // class containing methods to
192:                                          // check units
193:   Rxn         :TRxnDetails;              // reaction information container
194:   SelRxnNo    : integer;                 // current reaction being evaluated
195:
196: begin
197:   try {try .. finally block}
198:   try {try .. exception block}
199:     Check := TCheckUnits.Create;
200:     // get temperatures
201:     // include some range checking on the temperatures... particularly units!
202:     if (Check.Celsius(StrToFloat(EbTempInitial.Text))) then
203:       InitTemp := StrToFloat(EbTempInitial.Text);
204:     if (Check.Celsius(StrToFloat(EbTempFinal.Text))) then
205:       FinalTemp   := StrToFloat(EbTempFinal.Text);
206:     NumberCalcs := StrToInt(EbPoints.Text);
207:     // instantiate calculation class
208:     GibbsCalc   := TGibbsCalcClass.Create ('Gibbs Calculation');
209:     // PassTemperatures (Initial, Final : single; Points : integer)
210:     GibbsCalc.PassTemperatures (InitTemp, FinalTemp, NumberCalcs);
211:
212:     // is the calculation for one reaction or a set or reactions?
213:     case RGReactions.ItemIndex of
214:       0: begin // current reactions
215:         SelRxnNo := RunningDM.RxnInfoQy.FieldByName('RxnNo').AsInteger;
216:         Rxn:=TRxnDetails.Create (SelRxnNo);
```

```
217:         with GibbsCalcProgress do begin
218:           // show a form which will help the user know what is happening
219:           GenDescr.Caption := 'Calculating Gibbs free energy for reaction:';
220:           RxnDescr.Caption :=
221:            CompGenDM.RxnDescrQy.Lookup('RXNNO',SelRxnNo, 'DESCR');
222:           Refresh;
223:           Show;
224:         end; {end with GibbsCalcProgress do begin }
225:         GibbsCalc.CalcGibbsGivenRxnNo(SelRxnNo);
226:         // Output results to memo.
227:         OutputToMemo (Rxn);
228:         GibbsCalcProgress.Hide;
229:       end; {end case 0: }
230:
231:       1: begin // set of reactions
232:         // same as for current reaction for now
233:         Rxn:=TRxnDetails.Create
234:          (RunningDM.RxnInfoQy.FieldByName('RxnNo').AsInteger);
235:         //CalcGibbsForT;
236:         // Output results to memo.
237:         OutputToMemo (Rxn);
238:       end; {end case 1: }
239:     end; {end case RGReactions.ItemIndex of }
240:     with RunningDM.CalcOutputQy do begin
241:       if Active then close;
242:       if RequestLive then RequestLive := False;
243:       SQL.Clear;
244:       SQL.Add ('SELECT * FROM CALCOUTPUT ORDER BY TEMPK;');
245:       if not Active then Open;
246:       if not RequestLive then RequestLive := True;
247:     end; {end with CalcOutputQy do begin }
248:
249:   except
250:     on E:EDataError do begin
251:        MessageDlg(E.Message, mtError, [mbOK], 0);
252:     end; {end on E:EDataError do begin }
253:
254:   end; {end try .. except loop}
255:   finally
256:      GibbsCalc.Free;
257:      GibbsCalcProgress.Hide;
258:   end;  {end try .. finally loop}
259:
260:
261:
262: end; {end procedure TGibbsCalcForm.BtnGoClick(Sender: TObject); }
263:
264: { procedure to initialise data grid to display reaction information.
265: ************************************************************************** }
266: procedure TGibbsCalcForm.FormCreate(Sender: TObject);
267: begin
268:   DBGridGibbsCalcForm.DataSource := RunningDM.RxnInfoDS;
269:   // initialise MmGibbsCalcForm and associated speed buttons
270:   PCChartMemo.ActivePage := TSChart;
271:   BtnSaveMemo.Enabled    := False;
272:   BtnOpenMemo.Enabled    := False;
273:   BtnClearMemo.Enabled   := False;
274:   BtnPrintMemo.Enabled   := False;
275:
276:   BtnClear.Enabled       := False;
277:
278:   PointSeries            := 0;
279: end; {end procedure TGibbsCalcForm.FormCreate(Sender: TObject); }
280:
281: { procedure to save the contents of MmGibbsCalcForm
282: ************************************************************************** }
283: procedure TGibbsCalcForm.BtnSaveMemoClick(Sender: TObject);
284: begin
285:   with SaveDialog1 do begin
286:     if Execute then begin
287:       MmGibbsCalcForm.Lines.SaveToFile(Filename);
288:       SBGibbsCalcForm.Panels[3].Text:= 'Output: '+ ExtractFilename(Filename);
```

```
289:       OpenDialog1.Filename := ExtractFilename(Filename);
290:     end ; {end if Execute then begin }
291:   end; {end with SaveDialog1 do begin }
292: end; {end procedure TGibbsCalcForm.BtnSaveMemo(Sender: TObject); }
293:
294: { procedure to erase or clear the contents of MmGibbsCalcForm
295: ********************************************************************** }
296: procedure TGibbsCalcForm.BtnClearMemoClick(Sender: TObject);
297: begin
298:   MmGibbsCalcForm.Lines.Clear;
299: end; {end procedure TGibbsCalcForm.BtnClearMemo(Sender: TObject); }
300:
301: { procedure to load the contents of a file into MmGibbsCalcForm
302: ********************************************************************** }
303: procedure TGibbsCalcForm.BtnOpenMemoClick(Sender: TObject);
304: begin
305:   with OpenDialog1 do begin
306:     if Execute then begin
307:       MmGibbsCalcForm.Lines.LoadfromFile(OpenDialog1.FileName);
308:       SaveDialog1.Filename := OpenDialog1.FileName;
309:       SBGibbsCalcForm.Panels[3].Text:= 'Output: '+ OpenDialog1.Filename;
310:     end; {end if Execute then begin }
311:   end; {end with OpenDialog1 do begin }
312: end; {end procedure TGibbsCalcForm.BtnOpenMemo(Sender: TObject); }
313:
314: { procedure to disable memo specific speed buttons depending on PCChartMemo
315: ********************************************************************** }
316: procedure TGibbsCalcForm.PCChartMemoChange(Sender: TObject);
317: begin
318:   if PCChartMemo.ActivePage = TSMemo then begin
319:     BtnSaveMemo.Enabled     := True;
320:     BtnOpenMemo.Enabled     := True;
321:     BtnClearMemo.Enabled    := True;
322:     BtnPrintMemo.Enabled    := True;
323:     BtnClear.Enabled        := False;
324:   end {end if PCChartMemo.ActivePage = TSMemo then begin }
325:   else if PCChartMemo.ActivePage = TSChart then begin
326:     BtnSaveMemo.Enabled     := False;
327:     BtnOpenMemo.Enabled     := False;
328:     BtnClearMemo.Enabled    := False;
329:     BtnPrintMemo.Enabled    := False;
330:     BtnClear.Enabled        := False;
331:   end {end else if PCChartMemo.ActivePage = TSChart then begin}
332:   else if PCChartMemo.ActivePage = TSCalcOutput then begin
333:     BtnSaveMemo.Enabled     := False;
334:     BtnOpenMemo.Enabled     := False;
335:     BtnClearMemo.Enabled    := False;
336:     BtnPrintMemo.Enabled    := False;
337:     BtnClear.Enabled        := True;
338:   end; {end else if PCChartMemo.ActivePage = TSCalcOutput then begin }
339:
340: end; {end procedure TGibbsCalcForm.PCChartMemoChange(Sender: TObject); }
341:
342: { routine to clear the CalcOutputQy database
343: ********************************************************************** }
344: procedure TGibbsCalcForm.BtnClearClick(Sender: TObject);
345: begin
346:   // clear CalcOutput database
347:   // first of all check whether TSCalcOutput is the active page
348:   // although the button should not be enabled if this is not the case
349:   if PCChartMemo.ActivePage = TSCalcOutput then begin
350:     // TSRunning is active...
351:     // confirmation
352:     if MessageDlg ('Are you certain you want to clear '+
353:      DBGridCalcOutput.DataSource.DataSet.Name + '?', mtConfirmation,
354:      [mbYEs, mbNO], 0) = mrYes then begin
355:       with DBGridCalcOutput.DataSource.DataSet as TQuery do begin
356:         if Active then Close;
357:         if RequestLive then RequestLive := False;
358:         SQL.Clear;
359:         SQL.Add ('delete from CalcOutput;');
360:         ExecSQL;
```

```
361:          SQL.Clear;
362:          if Active then Close;              // don't leave an empty table open.
363:          ShowMessage ('Dataset '+ Name+ ' records deleted.');
364:          if not RequestLive then RequestLive := True;
365:        end; {end with DBGridCalcOutput.DataSource.DataSet as TQuery do begin}
366:      end; { if MessageDlg ('Are you certain you want to clear '+ }
367:    end; {end if PCChartMemo.ActivePage = TSCalcOutput then begin}
368:
369: end; {end procedure TGibbsCalcForm.BtnClearClick(Sender: TObject); }
370:
371: { procedure to copy the contents of the memo to the printer
372: ******************************************************************************* }
373: procedure TGibbsCalcForm.BtnPrintMemoClick(Sender: TObject);
374: var
375:    POutput : TextFile;
376:    N       : LongInt;
377: begin
378:    if PrintDialog1.Execute then  begin
379:      AssignPrn(POutput);
380:      Rewrite(POutput);
381:      Printer.Canvas.Font := MmGibbsCalcForm.Font;
382:      for N := 0 to MmGibbsCalcForm.Lines.Count -1 do
383:        Writeln(POutput, MmGibbsCalcForm.Lines [N]);
384:      CloseFile(POutput);
385:    end;
386: end; {end procedure TGibbsCalcForm.BtnPrintMemoClick(Sender: TObject); }
387:
388: end.
```

```
 1: unit GibbsCalcProgressUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   StdCtrls, ExtCtrls, ComCtrls;
 8:
 9: type
10:   TGibbsCalcProgress = class(TForm)
11:     PlMiddle: TPanel;
12:     GenDescr: TLabel;
13:     RxnDescr: TLabel;
14:     TempProgressBar: TProgressBar;
15:   private
16:     { Private declarations }
17:   public
18:     { Public declarations }
19:     procedure StepTempProgressBar(Percentage : integer);
20:
21:   end;
22:
23: var
24:   GibbsCalcProgress: TGibbsCalcProgress;
25:
26: implementation
27:
28: {$R *.DFM}
29:
30: { procedure to increment the progress bar to display calculation progress
31: ************************************************************************** }
32: procedure TGibbsCalcProgress.StepTempProgressBar(Percentage : integer);
33: begin
34:   with TempProgressBar do begin
35:     // first of all set the step by amount property
36:     Step := Percentage - Position;
37:     Stepit;
38:   end; {end  with TempProgressBar do begin }
39:
40: end; {end procedure TGibbsCalcProgress.StepTempProgressBar(percentage ...; }
41: end.
```

```
 1: unit LogInOutFormUnit;
 2:
 3: interface
 4:
 5: uses
 6:   SysUtils, Windows, Messages, Classes, Graphics, Controls,
 7:   StdCtrls, ExtCtrls, Forms, ComCtrls, PasswordDialog;
 8:
 9: type
10:   { TLogInOurForm class
11:   ************************************************************************** }
12:   TLogInOutForm = class(TForm)
13:     ProgressBar1: TProgressBar;
14:     Panel1: TPanel;
15:     LlInformation: TLabel;
16:     PlUserName: TPanel;
17:     LlUserName: TLabel;
18:     PasswordDialog1: TPasswordDialog;
19:     LlComment: TLabel;
20:
21:   public
22:     OutText: String;
23:     UserName: string;
24:     function Login : string;
25:     procedure VerifyPassword;
26:     procedure InitDB;
27:
28:   end; {end TLogInOutForm = class(TForm) }
29:
30:
31:
32: var
33:   LogInOutForm: TLogInOutForm;
34:
35: implementation
36:
37: {$R *.DFM}
38:
39: uses
40:   Dialogs, CompGenDMUnit, RunningDMUnit, MainFormUnit;
41:
42: { function to control login process
43:   returns string for MainForm.SBMainForm
44: ************************************************************************** }
45: function TLogInOutForm.Login : string;
46: begin
47:     LlInformation.Caption := 'Attempting to log into SimBioSys databases.';
48:     LogInOutForm.Show;
49:     VerifyPassword;
50:     LoginOutForm.Hide;
51:     Result := OutText;
52: end; {end function TLogInOutForm.Login : string; }
53:
54: { function to login a user, returns status panel string
55:   calls InitDB if password navigated successfully
56: ************************************************************************** }
57: procedure TLogInOutForm.VerifyPassword;
58: begin
59:   // compare password given in edit box with password in component property
60:   if PasswordDialog1.Execute then begin
61:     // correct password, proceed with login
62:     UserName := PasswordDialog1.GiveUsername;
63:     MainForm.User.StoreUserID (UserName);
64:     PlUserName.Caption := UserName;
65:     InitDB;
66:   end {end  if PasswordDialog1.Execute then begin }
67:   else
68:     // wrong password, abort login
69:     ShowMessage('Sorry, password not valid.');
70: end; {end procedure TLogInOutForm.VerifyPassword; }
71:
72: // Mainform OnCreate event.... call the initialisation procedures
```

```
 73: // procedure to initialise the databases and the tables
 74: // for security and data integrity all tables should be 'closed'.
 75: // for now the database is opened automatically by 'Karen'
 76: procedure TLogInOutForm.InitDB();
 77: begin
 78:     // show LogInOutForm 'dialog box'
 79:     LogInOutForm.Refresh;
 80:     try
 81:       // to avoid an EAccessViolation ensure components are created
 82:       with CompGenDM do begin
 83:         LlComment.Caption := 'Opening SpeciesDB';
 84:         LogInOutForm.Refresh;
 85:         ProgressBar1.Stepit;              // 7%
 86:         SpeciesDB.Open;
 87:         LlComment.Caption := 'Opening SpeciesDB.CompGenQy';
 88:         LogInOutForm.Refresh;
 89:         ProgressBar1.Stepit;              // 14%
 90:         CompGenQy.Open;
 91:         LlComment.Caption := 'Opening SpeciesDB.StateQy';
 92:         LogInOutForm.Refresh;
 93:         ProgressBar1.Stepit;              // 21%
 94:         StateQy.Open;
 95:         LlComment.Caption := 'Opening SpeciesDB.RxnDescrQy';
 96:         LogInOutForm.Refresh;
 97:         ProgressBar1.Stepit;              // 28%
 98:         RxnDescrQy.Open;
 99:         LlComment.Caption := 'Opening SpeciesDB.RxnStoichQy';
100:         LogInOutForm.Refresh;
101:         ProgressBar1.Stepit;              // 35%
102:         RxnStoichQy.Open;
103:         LlComment.Caption := 'Opening SpeciesDB.ThermoQy';
104:         LogInOutForm.Refresh;
105:         ProgressBar1.Stepit;              // 42%
106:         CompThermoQy.Open;
107:         LlComment.Caption := 'Opening SpeciesDB.CpDataQy';
108:         LogInOutForm.Refresh;
109:         ProgressBar1.Stepit;              // 49%
110:         CpDataQy.Open;
111:         LlComment.Caption := 'Opening SpeciesDB.RefDescrQy';
112:         LogInOutForm.Refresh;
113:         ProgressBar1.Stepit;              // 56%
114:         RefDescrQy.Open;
115:         LlComment.Caption := 'Opening SpeciesDB.CrissDescrQy';
116:         LogInOutForm.Refresh;
117:         ProgressBar1.Stepit;              // 63%
118:         CrissDescrQy.Open;
119:         ProgressBar1.Stepit;              // 70%
120:         LlComment.Caption := 'Opening SpeciesDB.CrissCobbleQy';
121:         LogInOutForm.Refresh;
122:         CrissCobbleQy.Open;
123:       end; { with CompGenDM do begin }
124:
125:       // now repeat that with the running database
126:       with RunningDM do begin
127:         LlComment.Caption := 'Opening RunningDB';
128:         LogInOutForm.Refresh;
129:         RunningDB.Open;
130:         ProgressBar1.Stepit;              // 77%
131:         LlComment.Caption := 'Opening RunningDB.CompConcQy';
132:         LogInOutForm.Refresh;
133:         CompConcQy.Open;
134:         ProgressBar1.Stepit;              // 84%
135:         LlComment.Caption := 'Opening RunningDB.RxnInfoQy';
136:         LogInOutForm.Refresh;
137:         RxnInfoQy.Open;
138:         ProgressBar1.Stepit;              // 91%
139:         LlComment.Caption := 'Opening RunningDB.MeanCpQy';
140:         LogInOutForm.Refresh;
141:         MeanCpQy.Open;
142:         LlComment.Caption := 'Opening RunningDB.CalcOutputQy';
143:         LogInOutForm.Refresh;
144:         CalcOutputQy.Open;
```

```
145:         ProgressBar1.Stepit;              // 98%
146:         L1Comment.Caption := '';
147:         LogInOutForm.Refresh;
148:       end; {end with RunningDM}
149:
150:       // logged in successfully.. display user name..
151:       OutText := UserName + ' logged in.';
152:     except
153:       // If user not able to login, display reason
154:       // later implement a logged out page
155:       on E:EDBEngineError do begin
156:         MessageDlg(E.Message, mtError, [mbOK], 0);
157:         OutText:=UserName + ' not logged in.';
158:         // maybe put up a page
159:       end; {end on E:EDBEngineError}
160:
161:       on EDatabaseError do
162:         MessageDlg('Database Error', mtError, [mbOK],0);
163:
164:       on EAccessViolation do
165:         MessageDlg('AccessViolation', mtError, [mbOK], 0);
166:     end; {end except ... }
167: end; {end procedure TLoginInOutForm.InitDB(); }
168:
169:
170: end.
171:
172:
```

```
 1: unit MainFormUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   ComCtrls, Menus, StdCtrls,
 8:   UserClassUnit;
 9:
10: type
11:   TMainForm = class(TForm)
12:     MainMenu1: TMainMenu;
13:     File1: TMenuItem;
14:     Login1: TMenuItem;
15:     Logout1: TMenuItem;
16:     Exit1: TMenuItem;
17:     Exit2: TMenuItem;
18:     SBMainForm: TStatusBar;
19:     Data1: TMenuItem;
20:     DisplayData1: TMenuItem;
21:     Species1: TMenuItem;
22:     Running1: TMenuItem;
23:     SelectReactions1: TMenuItem;
24:     Calculations1: TMenuItem;
25:     Gibbs1: TMenuItem;
26:     Reactions1: TMenuItem;
27:     Compounds1: TMenuItem;
28:     CompoundData2: TMenuItem;
29:     N1: TMenuItem;
30:     procedure Exit2Click(Sender: TObject);
31:     procedure Login1Click(Sender: TObject);
32:     procedure Species1Click(Sender: TObject);
33:     procedure Running1Click(Sender: TObject);
34:     procedure SelectReactions1Click(Sender: TObject);
35:     procedure Reactions1Click(Sender: TObject);
36:     procedure CompoundData2Click(Sender: TObject);
37:   private
38:     { Private declarations }
39:   public
40:     User : TUser;
41:     RecalcFlag   : boolean;
42:     { Public declarations }
43:   end;
44:
45: var
46:   MainForm: TMainForm;
47:
48: implementation
49:
50: uses DisplaySpeciesDataFormUnit,   DisplayRunningDataFormUnit,
51:  SelectionFormUnit, GibbsCalcFormUnit, LogInOutFormUnit,
52:   CompGenDataFormUnit, GibbsCalcProgressUnit;
53:
54: {$R *.DFM}
55:
56: { Procedure to Exit SimBioSys
57: ********************************************************************************* }
58: procedure TMainForm.Exit2Click(Sender: TObject);
59: begin
60:    //DbClose;                          // close down the database
61:    Close;
62: end;       {end procedure TMainForm.Exit2Click(Sender: TObject);}
63:
64:
65: { Procedure to log user into SimBioSys
66:   Note password is set at design time to shrapnel, irrespective of username
67: ********************************************************************************* }
68: procedure TMainForm.Login1Click(Sender: TObject);
69: begin
70:   // instantiate the TUser object
71:   User := TUser.Create ('none');
72:   SBMainForm.Panels[1].Text :=  LogInOutForm.Login;
```

```
 73:
 74:
 75: end; {endprocedure TMainForm.Login1Click(Sender: TObject); }
 76:
 77:
 78: { display data from tables of speciesDB
 79: ******************************************************************************* }
 80: procedure TMainForm.Species1Click(Sender: TObject);
 81: begin
 82:    DisplaySpeciesDataForm.Show;
 83: end; {end procedure TMainForm.Species1Click(Sender: TObject); }
 84:
 85: { display data from tables of runningDB
 86: ******************************************************************************* }
 87: procedure TMainForm.Running1Click(Sender: TObject);
 88: begin
 89:   DisplayRunningDataForm.Show;
 90: end; {end procedure TMainForm.Running1Click(Sender: TObject); }
 91: { show SelectionForm
 92: ******************************************************************************* }
 93: procedure TMainForm.SelectReactions1Click(Sender: TObject);
 94: begin
 95:   SelectionForm.Show
 96: end; {end procedure TMainForm.SelectReactions1Click(Sender: TObject); }
 97:
 98: { procedure to display Gibbs Calculation form
 99:   page on PCGibbsCalcForm is determined by the menu option chosen
100: ******************************************************************************* }
101: procedure TMainForm.Reactions1Click(Sender: TObject);
102: begin
103:   with GibbsCalcForm do begin
104:     PCGibbsCalcForm.ActivePage := TSReactions;
105:     SBGibbsCalcForm.Panels[2].Text := 'Reactions.';
106:     Show;
107:   end; {end   with GibbsCalcForm do begin }
108: end; {end procedure TMainForm.Reactions1Click(Sender: TObject); }
109:
110:
111: { procedure to display Compound data insert, view or update
112:   tables CompGen, Comp Thermo and Cp Data
113: ******************************************************************************* }
114: procedure TMainForm.CompoundData2Click(Sender: TObject);
115: begin
116:   CompGenDataForm.Show;
117: end; {end procedure TMainForm.CompoundData2Click(Sender: TObject); }
118:
119: end.
```

```
unit RxnClassUnit;

interface

uses
  DB,
  CompClassUnit;

type
  { Class to contain information about current reaction and control selection
    of compounds and stoichiometry for each reaction. Descendant of TDescription
    hence it is time, date and user stamped ... }
  TRxnDetails = class (TDescription)
  private
    LocateOptions   : TLocateOptions;        // search options for 'Locate'
    icomp           : integer;               // compound counter
    StoichEntries   : integer;               // total stoich. records
    RxnNoInt        : integer;               // reaction no, 1° key

    CompStoichInt   : integer;               // compound stoichiometry
    integrity       : boolean;               // flag to pass variables
    GibbsRxnStd     : single;                // standard Gibbs for the reaction
    KGibbsRxnStd    : single;                // K corresponding to GibbsRxnStd
    EntropyRxnStd   : single;                // standard entropy change for rxn
    CompNoInt       : integer;               // pass to TCompDetails
    CpMeanTRxn      : single;                // mean heat capacity at T
    TemperatureK    : single;                // temperature (K)

    procedure FilterRxnStoich;               // filter RxnStoich table
    procedure StoreGibbsRxnStd;              // store GibbsRxnStd in RxnInfo
    procedure StoreEntropyRxnStd;            // store EntropyRxnStd in RxnInfo
    procedure StoreKGibbsRxnStd;             // store KGibbsRxnStd in RxnInfo
    procedure CalcCpMeanTRxn(TempK : single);

  public
    constructor Create (RxnNo : integer);
    procedure GetFirstCompInfo;              // use CompGenDM.FindFirstRxnStoich
    function GetNextCompInfo : boolean;      // use CompGenDM.FindNextRxnStoich
    procedure GetCompInfo (var CompNo : integer; var CompStoich : integer);
    procedure CalcThermosRxnStd;
    function PassRxnNoStr : string;
    function PassGibbsRxnStdStr : string;
    function PassEntropyRxnStdStr : string;
    function PassRxnDescrStr : string;
    function PassKGibbsRxnStdStr : string;
    procedure Zeroicomp;
    procedure Incicomp;
    function PassCompNoInt :integer;
    function PassStoichEntries :integer;
    function PassCompStoichInt : integer;
    function PassCpMeanTRxn (TempK : single) : single;


  end;

implementation

uses
  Dialogs,
  Exceptns, CompGenDMUnit, RunningDMUnit, MainFormUnit, Constants;


{ constructor for TRxnDetails, calls TDescription.Create (Descrip : string)
  checks whether RxnNo exists in both RxnDescrQy and RxnStoichQy
********************************************************************************* }
constructor TRxnDetails.Create (RxnNo : integer);
begin
  // initialise search options
  LocateOptions := [loCaseInsensitive];      // note whole word only
  // set TDescription details
  TDescription.Create ('RxnDetails');
  RxnNoInt := RxnNo;
```

```
  73:   // check compound number is valid
  74:   if RxnNoInt < 0 then
  75:     raise ERxnNoInvalid.Create('Reaction number is invalid.')
  76:   else {else for if RxnNoInt < 0 then }
  77:     if not (CompGenDM.RxnDescrQy.Locate('RxnNo', RxnNoInt, LocateOptions)) then
  78:       raise ERxnNotDescribed.Create
  79:        ('Reaction is not described in description table.')
  80:   else {else for if RxnNoInt < 0 then, if not (CompGenDM.RxnDescrQy.Locate ... }
  81:     if not (CompGenDM.RxnStoichQy.Locate('RxnNo', RxnNoInt, LocateOptions)) then
  82:       raise ERxnNoStoichData.Create
  83:        ('Reaction stoichiometry is not described in stoichiometry table.');
  84:
  85:   // initialise icomp counter - only occurs when instantiated.
  86:   icomp      := 0;     // i.e. next compound considered is the first compound
  87:   integrity  := true;  // default is to pass the varaibles
  88:   CpMeanTRxn := -1;    // i.e. not calculated yet (requires temp (K))
  89:
  90:    // get the number of stoich entries for the reaction
  91:   StoichEntries := CompGenDM.RxnDescrQy.Lookup ('RXNNO', RxnNoInt, 'ENTRIES');
  92:
  93:   // retrieve or calculate thermodynamics information for the reaction
  94:   CalcThermosRxnStd;
  95:
  96: end; {end constructor TRxnDetails.Create (RxnNo : integer; }
  97:
  98: { procedure to calculate the mean heat capacity for the reaction given the
  99:   reaction number, and pass the number to the calling function
 100: ******************************************************************************* }
 101: procedure TRxnDetails.CalcCpMeanTRxn (TempK : single);
 102: var
 103:   CompoundCounter : integer;
 104:   CompoundDetails : TCompDetails;
 105:   CpMeanT         : single;
 106: begin
 107: try  // try .. finally
 108:   // initialisation
 109:   CompoundDetails := TCompDetails.Create;
 110:   CompoundCounter := 0;
 111:   CpMeanT         := 0;
 112:   CpMeanTRxn      := 0;
 113:   TemperatureK    := TempK;
 114:
 115:   // start looping through compounds to calculate the mean Cp for the reaction
 116:   while CompoundCounter <> -1 do begin
 117:     // get compound number, thermodynamic and Criss Cobble info
 118:     if CompoundCounter = 0 then begin
 119:       GetFirstCompInfo;
 120:     end {end if icomp = 0 then begin }
 121:     else {else for if icomp = 0 then begin }
 122:       GetNextCompInfo;
 123:     // now the values of CompNoInt and CompNoStoich are set
 124:     // retrieve Gibbs and Entropy information by instantiating compound
 125:     // by taking this approach the existence of data is checked
 126:     CompoundDetails.NewCompound (CompNoInt );
 127:
 128:     CpMeanT := CompoundDetails.CalcCpMeanT(TemperatureK );
 129:     CpMeanTRxn := CpMeanTRxn + CpMeanT*CompStoichInt;
 130:
 131:     inc(CompoundCounter,1);
 132:
 133:     if CompoundCounter = StoichEntries
 134:       then CompoundCounter := -1;
 135:   end; {end while icomp <> -1 do begin }
 136: finally
 137:   CompoundDetails.Free;
 138: end; {end try .. finally block}
 139: end; {end procedure TRxnDetails.CalcCpMeanTRxn;
 140:
 141:
 142: { function to pass the mean heat capacity for the reaction to the calling
 143:   function, checks whether the value has been calculated.
 144: ******************************************************************************* }
```

```
145: function TRxnDetails.PassCpMeanTRxn (TempK : single) : single;
146: begin
147:   // if another temperature, then recalculate CpMeanTRxn
148:   if TempK <> TemperatureK then CpMeanTRxn := -1;
149:   if CpMeanTRxn = -1 then
150:       CalcCpMeanTRxn (TempK);
151:   Result := CpMeanTRxn;
152: end; {end function TRxnDetails.PassCpMeanTRxn (TempK : single) : single;}
153:
154: { procedure to pass next (or first) compound information
155:   note that if RxnStoich position is altered, then this procedure will need to
156:   be able to retrace to get to the right position, using icomp as index
157: ********************************************************************************* }
158: procedure TRxnDetails.GetCompInfo (var CompNo : integer;
159:  var CompStoich : integer);
160: begin
161:   // check whether or not this is the first compound?
162:   if icomp = 0 then begin
163:     integrity := true;
164:     Description := 'First Compound';
165:     FilterRxnStoich;
166:     GetFirstCompInfo;
167:   end {end if icomp = 0 then }
168:   else begin
169:     integrity := GetNextCompInfo;
170:     Description := ('Compound ' + IntToStr(icomp));
171:   end;  {end else for if icomp = 0 }
172:
173:
174:   if integrity then begin
175:     // pass information to referenced variables
176:     CompNo     := CompNoInt;
177:     CompStoich := CompStoichInt;
178:   end {end if integrity then }
179:   else begin
180:     CompNo     := 0;
181:     CompStoich := 0;
182:   end; {end else for if not integrity then }
183: end; {end procedure TRxnDetails.GetNextCompInfo (var CompNo : integer, ... }
184:
185: { procedure to call CompGenDM.SetRxnNoFilter, i.e. to filter RxnStoich
186: ********************************************************************************* }
187: procedure TRxnDetails.FilterRxnStoich;
188: begin
189:   //procedure TCompGenDM.SetRxnNoFilter (SelRxnNo : Integer);
190:   CompGenDM.SetRxnNoFilter (RxnNoInt);
191: end;  {end procedure TRxnDetails.FilterRxnStoich; }
192:
193: { procedure to call TCompGenDM.FindFirstRxnStoich;  passes true if an error has
194:   occurred - then raise an ENoRxnStoichData exception (with RxnNo information)
195: ********************************************************************************* }
196: procedure TRxnDetails.GetFirstCompInfo;
197: begin
198:   //function TCompGenDM.FindFirstRxnStoich : Boolean;
199:   if CompGenDM.FindFirstRxnStoich then raise EStoichDataError.Create
200:    ('Check stoichiometry data for reaction ' + IntToStr(RxnNoInt) + '. ');
201:   // retrieve compound information from RxnStoich (current position
202:   CompNoInt     := CompGenDM.RxnStoichQy.FieldByName ('COMPNO').AsInteger;
203:   CompStoichInt := CompGenDM.RxnStoichQy.FieldByName ('STOICH_COEFF').AsInteger;
204: end; {end procedure TRxnDetails.GetFirstCompInfo; }
205:
206: { function to call TCompGenDM.FindNextRxnStoich;  passes true if an error has
207:   occurred - then raise an ENoRxnStoichData exception (with RxnNo information)
208:   passes false if values must not be passed to the calling routine
209: ********************************************************************************* }
210: function TRxnDetails.GetNextCompInfo : boolean;
211: begin
212:   //function TCompGenDM.FindNextRxnStoich : Boolean;
213:   if CompGenDM.FindNextRxnStoich then begin
214:     if icomp <> (StoichEntries) then raise EStoichDataError.Create
215:      ('Check stoichiometry data for reaction ' + IntToStr(RxnNoInt) + '. ')
216:     else // - drawn to a natural conclusion, i.e. no errors.
```

```
217:       Result      := False;  // i.e. no integrity, do not pass values on
218:    end {end if CompGenDM.FindNextRxnStoich then begin }
219:                                  // i.e. next stoichiometric record found
220:    else begin
221:      // retrieve compound information from RxnStoich (current position
222:      CompNoInt     := CompGenDM.RxnStoichQy.FieldByName ('COMPNO').AsInteger;
223:      // instantiate compound
224:      CompStoichInt :=CompGenDM.RxnStoichQy.FieldByName('STOICH_COEFF').AsInteger;
225:      Result        := True;
226:    end; {end else for if CompGenDM.FindNextRxnStoich then begin }
227: end; {end procedure TRxnDetails.GetNextCompInfo; }
228:
229: { procedure to store GibbsRxnStd in RxnInfo
230: ********************************************************************************* }
231: procedure TRxnDetails. StoreGibbsRxnStd;
232: var
233:    SQLStrUpd : string;                    // string for update e-SQL
234: begin
235:    // initialise update string for e-SQL
236:    SQLStrUpd :=('UPDATE RXN_INFO SET STD_GIBBS_RXN  = "%s" WHERE (RXNNO = "%s")');
237:    with RunningDM.RxnInfoQy do begin
238:      if Active then Close;
239:      SQL.Clear;
240:      SQL.Add(Format (SQLStrUpd, [FloatToStr (GibbsRxnStd), IntToStr(RxnNoInt)]));
241:      ExecSQL;
242:      SQL.Clear;
243:      SQL.Add ('SELECT * FROM RXN_INFO ORDER BY RXNNO;');
244:      if not Active then Open;
245:    end; { end   with RunningDM.RxnInfoQy do begin }
246: end; {end procedure TRxnDetails. StoreGibbsRxnStd; }
247:
248: { procedure to store KGibbsRxnStd in RxnInfo
249: ********************************************************************************* }
250: procedure TRxnDetails. StoreKGibbsRxnStd;
251: var
252:    SQLStrUpd : string;                    // string for update e-SQL
253: begin
254:    // initialise update string for e-SQL
255:    SQLStrUpd :=('UPDATE RXN_INFO SET STD_K_RXN  = "%s" WHERE (RXNNO = "%s")');
256:    with RunningDM.RxnInfoQy do begin
257:      if Active then Close;
258:      SQL.Clear;
259:      SQL.Add(Format (SQLStrUpd, [FloatToStr(KGibbsRxnStd), IntToStr(RxnNoInt)]));
260:      ExecSQL;
261:      SQL.Clear;
262:      SQL.Add ('SELECT * FROM RXN_INFO ORDER BY RXNNO;');
263:      if not Active then Open;
264:    end; { end   with RunningDM.RxnInfoQy do begin }
265: end; {end procedure TRxnDetails. StoreGibbsRxnStd; }
266:
267: { procedure to store EntropyRxnStd in RxnInfo
268: ********************************************************************************* }
269: procedure TRxnDetails.StoreEntropyRxnStd;
270: var
271:    SQLStrUpd : string;                    // string for update e-SQL
272: begin
273:    // initialise update string for e-SQL
274:    SQLStrUpd:=('UPDATE RXN_INFO SET STD_ENTROPY_RXN = "%s" WHERE (RXNNO = "%s")');
275:    with RunningDM.RxnInfoQy do begin
276:      if Active then Close;
277:      SQL.Clear;
278:      SQL.Add(Format(SQLStrUpd, [FloatToStr(EntropyRxnStd), IntToStr(RxnNoInt)]));
279:      ExecSQL;
280:      SQL.Clear;
281:      SQL.Add ('SELECT * FROM RXN_INFO ORDER BY RXNNO;');
282:      if not Active then Open;
283:    end; { end   with RunningDM.RxnInfoQy do begin }
284: end; {end procedure TRxnDetails.StoreEntropyRxnStd; }
285:
286: { procedure to calculate the standard Gibbs free energy for the selection rxn
287: ********************************************************************************* }
288: procedure TRxnDetails.CalcThermosRxnStd;
289: var
```

```
289:   Compound           : TCompDetails;
290:   GibbsCompStdStr    : string;
291:   EntropyCompStdStr  : string;
292:
293: begin
294:   // initialise calculation variables
295:   GibbsRxnStd   := 0;
296:   EntropyRxnStd := 0;
297:   // note keep exception handling for main calling routine
298:   try                                                          // finally
299:     Compound:= TCompDetails.Create(); // instantiate compound
300:     // icomp = 0 then it is the first compound of the reaction
301:     while icomp <> -1 do begin
302:       // get compound number and thermos info of 'next' compound
303:       if icomp = 0 then begin
304:         GetFirstCompInfo;
305:         // Compound := TCompDetails.Create (CompNoInt);
306:       end {end if icomp = 0 then begin }
307:       else begin
308:         GetNextCompInfo;
309:       end; {end else for if icomp = 0 then begin }
310:
311:       // now the values of CompNoInt and CompNoStoich are set
312:       // retrieve Gibbs and Entropy information by instantiating compound
313:       // by taking this approach the existence of data is checked
314:      Compound.NewCompound ( CompNoInt);
315:       GibbsRxnStd := GibbsRxnStd + (CompStoichInt *
316:        (StrtoFloat(Compound.GiveGibbsStd)));
317:       EntropyRxnStd := EntropyRxnStd + (CompStoichInt *
318:        (StrtoFloat(Compound.GiveEntropyStd)));
319:       // increment icomp (compound number)
320:       icomp := icomp + 1;
321:       // check whether that was the last compound?
322:       if icomp = StoichEntries  then icomp := -1;
323:     end; {end while icomp <> -1 do begin }
324:
325:     // calculate the equilibrium constant corresponding to GibbsRxnStd
326:     // std. conditions, hence T := 298K
327:     KGibbsRxnStd := ReturnKRxnforGibbsRxn ( GibbsRxnStd, 298.15);
328:     // store reaction calculations in RunningDM.RxnInfoQy
329:     StoreGibbsRxnStd;
330:     StoreEntropyRxnStd;
331:     StoreKGibbsRxnStd;
332:
333:     finally
334:        Compound.Free;   // clear up memory
335:    end; {end try... }
336:
337: end; { end procedure TRxnDetails.CalcThermosRxnStd;}
338:
339:
340: { procedure to pass the reaction number
341: ********************************************************************************* }
342: function TRxnDetails.PassRxnNoStr : string;
343: begin
344:   Result := IntToStr(RxnNoInt);
345: end; { end procedure TRxnDetails.PassRxnNoStr : string; }
346:
347: { procedure to pass the reaction number
348: ********************************************************************************* }
349: function TRxnDetails.PassGibbsRxnStdStr : string;
350: begin
351:   Result := FloatToStrF(GibbsRxnStd, ffGeneral, 7, 8);
352: end; {end procedure TRxnDetails.PassGibbsRxnStdStr : string; }
353:
354: { procedure to pass the equilibrium constant for standard conditions
355: ********************************************************************************* }
356: function TRxnDetails.PassKGibbsRxnStdStr : string;
357: begin
358:   Result := FloatToStrF(KGibbsRxnStd, ffGeneral, 7, 8);
359: end; {end function TRxnDetails.PassKStdGibbsRxnStr : string; }
360:
```

```
361:
362: { procedure to pass the reaction number
363: ********************************************************************************* }
364: function TRxnDetails.PassEntropyRxnStdStr : string;
365: begin
366:   Result := FloatToStrF(EntropyRxnStd, ffGeneral, 7, 8);
367: end; {end procedure TRxnDetails.PassEntropyRxnStdStr : string; }
368:
369: { procedure to pass the reaction description
370: ********************************************************************************* }
371: function TRxnDetails.PassRxnDescrStr : string;
372: begin
373:   Result := CompGenDM.RxnDescrQy.Lookup('RXNNO', IntToStr(RxnNoInt), 'DESCR');
374: end; {end function TRxnDetails.PassRxnDescrStr : string; }
375:
376: { reset the counter icomp
377: ********************************************************************************* }
378: procedure TRxnDetails.Zeroicomp;
379: begin
380:   icomp := 0;
381: end; {end procedure TRxnDetails.Zeroicomp; }
382:
383: { increment the counter icomp
384: ********************************************************************************* }
385: procedure TRxnDetails.Incicomp;
386: begin
387:   inc(icomp,1);
388: end; {end procedure TRxnDetails.Incicomp; }
389:
390: { procedure to pass the stored CompNoInt to the calling routine
391: ********************************************************************************* }
392: function TRxnDetails.PassCompNoInt :integer;
393: begin
394:   Result := CompNoInt;
395: end; {end function TDescription.PassCompNoInt : integer; }
396:
397: { procedure to pass the stored CompStoichInt to the calling routine
398: ********************************************************************************* }
399: function TRxnDetails.PassCompStoichInt : integer;
400: begin
401:   Result := CompStoichInt;
402: end; {end function TDescription.PassCompStoichInt : integer; }
403:
404: { procedure to pass the stored StoichEntries to the calling routine
405: ********************************************************************************* }
406: function TRxnDetails.PassStoichEntries :integer;
407: begin
408:   Result := StoichEntries;
409: end; {end function TDescription.PassStoichEntries :string; }
410:
411: end.
```

```
 1: unit SelectionFormUnit;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   ComCtrls, Buttons, ExtCtrls, Grids, DBGrids, StdCtrls, DBTables;
 8:
 9: type
10:   TSelectionForm = class(TForm)
11:     SBSelectionForm: TStatusBar;
12:     PCSelectionForm: TPageControl;
13:     TSSpecies: TTabSheet;
14:     DBGridSpecies: TDBGrid;
15:     TSRunning: TTabSheet;
16:     DBGridRunning: TDBGrid;
17:     PlSelectionForm: TPanel;
18:     BtnClose: TSpeedButton;
19:     BtnClear: TSpeedButton;
20:     RGRunning: TRadioGroup;
21:     RGSpecies: TRadioGroup;
22:     BtnAdd: TSpeedButton;
23:     BtnCheckCompounds: TSpeedButton;
24:     BtnDelete: TSpeedButton;
25:     TSCalcOutput: TTabSheet;
26:     PlCalcOutput: TPanel;
27:     DBGridCalcOutput: TDBGrid;
28:     procedure BtnCloseClick(Sender: TObject);
29:     procedure FormCreate(Sender: TObject);
30:     procedure BtnClearClick(Sender: TObject);
31:     procedure PCSelectionFormChange(Sender: TObject);
32:     procedure BtnAddClick(Sender: TObject);
33:     procedure BtnDeleteClick(Sender: TObject);
34:     procedure BtnCheckCompoundsClick(Sender: TObject);
35:   private
36:     { Private declarations }
37:   public
38:     { Public declarations }
39:   end;
40:
41: var
42:   SelectionForm: TSelectionForm;
43:
44: implementation
45:
46: uses CompGenDMUnit, RunningDMUnit, Exceptns;
47:
48: {$R *.DFM}
49: { 'Close' the form, passes active focus to the main form
50: ****************************************************************************** }
51: procedure TSelectionForm.BtnCloseClick(Sender: TObject);
52: begin
53:   SelectionForm.Hide;
54: end; {end procedure TSelectionForm.BtnCloseClick(Sender: TObject); }
55:
56: { procedure to initialise selection form components
57: ****************************************************************************** }
58: procedure TSelectionForm.FormCreate(Sender: TObject);
59: begin
60:   // show the running information initially
61:   PCSelectionForm.ActivePage := TSRunning;
62:   BtnAdd.Enabled := False;
63:   SBSelectionForm.Panels[1].Text := 'Running Data Base'
64: end; {end procedure TSelectionForm.FormCreate(Sender: TObject); }
65:
66: { procedure to clear the table displayed in DBGridRunning
67:   only enabled if TSRunning is the active page.
68: ****************************************************************************** }
69: procedure TSelectionForm.BtnClearClick(Sender: TObject);
70: begin
71:   // first of all check whether TSRunning is the active page
72:   // although the button should not be enabled if this is not the case
```

```
  73:   if PCSelectionForm.ActivePage = TSRunning then begin
  74:     // TSRunning is active...
  75:     // confirmation
  76:     if MessageDlg ('Are you certain you want to clear '+
  77:      DBGridRunning.DataSource.DataSet.Name + '?', mtConfirmation, [mbYEs, mbNO],
  78:      0) = mrYes then begin
  79:       with DBGridRunning.DataSource.DataSet as TQuery do begin
  80:         if Active then Close;
  81:         if not RequestLive then RequestLive := True;
  82:         if not Active then Open;
  83:         while not EOF do begin
  84:           Delete;
  85:         end;
  86:         if Active then Close;                  // don't leave an empty table open.
  87:         ShowMessage ('Dataset '+ Name+ ' records deleted.');
  88:         if not RequestLive then RequestLive := True;
  89:       end; {end with DBGridRunning.DataSource.DataSet do begin }
  90:     end; { if MessageDlg ('Are you certain you want to clear '+ }
  91:   end {end if PCSelectionForm.ActivePage = TSRunning then begin }
  92:   else if PCSelectionForm.ActivePage = TSCalcOutput then begin
  93:     // TSRunning is active...
  94:     // confirmation
  95:     if MessageDlg ('Are you certain you want to clear '+
  96:      DBGridCalcOutput.DataSource.DataSet.Name + '?', mtConfirmation, [mbYEs, mbNO],
  97:      0) = mrYes then begin
  98:       with DBGridCalcOutput.DataSource.DataSet as TQuery do begin
  99:         if Active then Close;
 100:         if not RequestLive then RequestLive := True;
 101:         if not Active then Open;
 102:         while not EOF do begin
 103:           Delete;
 104:         end;
 105:         SQL.Clear;
 106:         SQL.Add('SELECT * FROM CALCOUTPUT');
 107:         if Active then Close;
 108:         ShowMessage ('Dataset '+ Name+ ' records deleted.');
 109:         if not RequestLive then RequestLive := True;
 110:       end; {end with DBGridCalcOutput.DataSource.DataSet do begin }
 111:     end; { if MessageDlg ('Are you certain you want to clear '+ }
 112:   end; {end else if PCSelectionForm.ActivePage = TSCalcOutput then}
 113: end; {end procedure TSelectionForm.BtnClearClick(Sender: TObject);}
 114:
 115:
 116: { procedure to display appropriate table according to the radio group and which
 117:   Tab sheet is the active page.
 118: ********************************************************************************* }
 119: procedure TSelectionForm.PCSelectionFormChange(Sender: TObject);
 120: begin
 121:   if PCSelectionForm.ActivePage = TSRunning then begin
 122:     // aim is to display database table dictated by radio buttons
 123:     // Running Database Page
 124:     SBSelectionForm.Panels[1].Text := 'Running data base';
 125:     BtnAdd.Enabled := False;
 126:     BtnDelete.Enabled := True;
 127:     BtnClear.Enabled := True;
 128:     // have a look at what is selected in the radio group
 129:     case RGRunning.ItemIndex of
 130:       0: begin
 131:         DBGridRunning.DataSource := RunningDM.RxnInfoDS;
 132:         SBSelectionForm.Panels[2].Text := 'Selected Reactions';
 133:         BtnCheckCompounds.Enabled := True;
 134:       end; {end 0: begin }
 135:       1: begin
 136:         DBGridRunning.DataSource := RunningDM.CompConcDS;
 137:         SBSelectionForm.Panels[2].Text := 'Selected Compounds';
 138:         BtnCheckCompounds.Enabled := False;
 139:      end; {end 1: begin }
 140:     end; {end case TableRg2.ItemIndex of }
 141:   end {end   if PCSelectionForm.ActivePage = TSRunning then begin }
 142:   else if PCSelectionForm.ActivePage = TSSpecies then begin
 143:     // aim is to display database table dictated by radio buttons
 144:     // Species Data Base Page
```

```
145:     SBSelectionForm.Panels[1].Text := 'Species data base';
146:     SBSelectionForm.Panels[2].Text := '';
147:     BtnCheckCompounds.Enabled := False;
148:     BtnDelete.Enabled := False;
149:     BtnClear.Enabled := False;
150:     // have a look at what is selected in the radio group
151:     case RGSpecies.ItemIndex of
152:       0: begin
153:         DBGridSpecies.DataSource := CompGenDM.CompGenDS;
154:         SBSelectionForm.Panels[2].Text := 'General Compound Information';
155:         BtnAdd.Enabled := True;
156:       end; {end 0: begin }
157:       1: begin
158:         DBGridSpecies.DataSource := CompGenDM.CompThermoDS;
159:         SBSelectionForm.Panels[2].Text := 'Thermodynamic Data';
160:         BtnAdd.Enabled := False;
161:       end; {end 1: begin}
162:       2: begin
163:         DBGridSpecies.DataSource := CompGenDM.RxnDescrDS;
164:         SBSelectionForm.Panels[2].Text := 'Available Reactions';
165:         BtnAdd.Enabled := True;
166:       end; {end 2: begin }
167:     end; {end case RGSpecies.ITemIndex}
168:   end {end else if PCSelectionForm.ActivePage = TSSpecies then begin }
169:   else if PCSelectionForm.ActivePage = TSCalcOutput then begin
170:     SBSelectionForm.Panels[1].Text := 'Gibbs calculation output ';
171:     SBSelectionForm.Panels[2].Text := '';
172:     BtnCheckCompounds.Enabled := False;
173:     BtnAdd.Enabled := False;
174:     BtnDelete.Enabled := False;
175:     BtnClear.Enabled := True;
176:   end; {end else if PCSelectionForm.ActivePage = TSCalcOutput then begin }
177:
178: end; {end procedure TSelectionForm.PCSelectionFormChange(Sender: TObject); }
179:
180: { procedure to add the current reaction or component to appropriate running
181:   database table.
182: ********************************************************************************* }
183: procedure TSelectionForm.BtnAddClick(Sender: TObject);
184: var
185:   RxnNoStr      : string;                    // Reaction number to be added
186:   CompNoStr     : string;                    // Compound number to be added
187:   temp          : string;                    // temporary string storage
188: begin
189:   // check whether it is a component or a reaction which is being added
190:   if PCSelectionForm.ActivePage = TSSpecies then begin
191:     if RGSpecies.ItemIndex = 0 then begin      // compound
192:       if MessageDlg ('Are you sure you want to add the current component?',
193:        mtConfirmation, [mbYes, mbNo],0) = mrYes then begin
194:         // get the compound number
195:         temp := SBSelectionForm.Panels[2].Text;
196:         SBSelectionForm.Panels[2].Text := 'Adding Compound';
197:         CompNoStr :=
198:          DBGridSpecies.DataSource.DataSet.FieldByName('CompNo').AsString;
199:         RunningDM.InsertCompoundCompConc (CompNoStr);
200:         SBSelectionForm.Panels[2].Text := temp;
201:       end; {end if MessageDlg ('Are you sure you want to add the current ...}
202:     end {end if RGSpeciesItemIndex = 0 then begin }
203:     else if RGSpecies.ItemIndex = 2 then begin      // reaction
204:       if MessageDlg ('Are you sure you want to add the current reaction?',
205:        mtConfirmation, [mbYes, mbNo],0) = mrYes then begin
206:         // get the compound number
207:         temp := SBSelectionForm.Panels[2].Text;
208:         SBSelectionForm.Panels[2].Text := 'Adding Reation';
209:         RxnNoStr :=
210:          DBGridSpecies.DataSource.DataSet.FieldByName('RxnNo').AsString;
211:         RunningDM.InsertReactionRxnInfo (RxnNoStr);
212:         SBSelectionForm.Panels[2].Text := temp;
213:       end; {end if MessageDlg ('Are you sure you want to add the current ...}
214:     end; {end else if RGSpeciesItemIndex = 2 then begin      // reaction }
215:   end; {end if PCSelectionForm.ActivePage = TSSpecies then begin }
216: end; {end procedure TSelectionForm.BtnAddClick(Sender: TObject); }
```

```
217:
218: { procedure to delete the current reaction or compound from the table
219: ********************************************************************** }
220: procedure TSelectionForm.BtnDeleteClick(Sender: TObject);
221: var
222:   temp : string;        // temporary storage for SBSelectionForm.Panels[2].Text
223: begin
224:   // confirmation
225:   if MessageDlg ('Are you sure you want to delete the current record?',
226:    mtConfirmation, [mbYES, mbNO],0) = mrYES then begin
227:     temp := SBSelectionForm.Panels[2].Text;
228:     SBSelectionForm.Panels[2].Text:= 'Deleting current record.';
229:     // get the active query object from the datasource...
230:     with DBGridRunning.DataSource do begin
231:       with DataSet as TQuery do begin
232:         if Active then Close;
233:         if not RequestLive then RequestLive := True;
234:         if not active then Open;
235:         Delete;
236:       end; {end with DataSet do begin }
237:     end; {end with DBGridRunning.DataSource do begin }
238:     SBSelectionForm.Panels[2].Text:= temp;
239:   end; {end if ModalResult = mrYES then begin }
240: end;  {end procedure TSelectionForm.BtnDeleteClick(Sender: TObject); }
241:
242: { routine to check whether all the compounds required by the selected rxns
243:   in RxnInfo are contained as records within CompConc
244:   Adds compounds if necessary.  Based on the rxn Stoichionmetry, it is recorded
245:   as a component (CompCategory := 0) or derived species (CompCategory := 1)
246: ********************************************************************************* }
247: procedure TSelectionForm.BtnCheckCompoundsClick(Sender: TObject);
248: var
249:   irxn            : Integer;                    // rxn ordinal counter
250:   icomp           : Integer;                    // compound ordingal counter
251:   LastRxnNo       : Boolean;                    // FindFirstRxnNoRxnInfo ...
252:   FirstRxnNo      : Boolean;                    // first RxnNo record?
253:   LastCompound    : Boolean;                    // FindFirstRxnStoich ...
254:   ErrorStatus     : Boolean;                    // true if error occurred
255:   RxnNoStr        : String;                     // storage variable for RxnNo
256:   CompNoStr       : String;                     // compound number (key)
257:   CompFormStr     : String;                     // compound formula
258:   CompStoichStr   : String;                     // compound stoich., RxnNo
259:   CompCategory    : Integer;                    // Boolean 0  - component,
260:                                                 //         1 - derived species
261:   SQLStrIns       : String;                     // eSQL string - insert
262:   SQLStrSel       : String;                     // eSQL string - select
263:   SQLStrUpd       : String;                     // eSQL string - update
264:   LocateOptions   : TLocateOptions;             // locate options
265: begin
266:   // initialise default values
267:   SQLStrIns     := 'INSERT INTO COMPCONC (CFORM) VALUES ("%s")';
268:   LocateOptions := [loCaseInsensitive]; // whole words only
269:   LastRxnNo     := False;     // default is that there are more records
270:   FirstRxnNo    := True;      // start with first entry
271:   LastCompound  := False;     // default is that there are more compounds
272:   ErrorStatus   := False;     // default is that no error has occurred
273:   irxn          := 0;         // zero reaction counter
274:   icomp         := 0;         // zero compound counter
275:
276:   try    // finally
277:     try  // except
278:       // mark all records in CompConc as non-current
279:       RunningDM.MarkCompConcNonCurrent;
280:       with RunningDM.RxnInfoQy do begin
281:         // start at the begining of the list contained by RxnInfo
282:         Close;
283:         SQL.Clear;
284:         SQL.Add ('SELECT * FROM RXN_INFO ORDER BY RXNNO');
285:         Open;
286:       end; {end with RunningDM.RxnInfoQy do ... }
287:       while not LastRxnNo do begin
288:         // check whether need to use FindFirst or FindNext
```

```
289:          if FirstRxnNo then begin
290:            // set RxnInfoQy to filtered, select all RxnNo > 0, i.e. all
291:            RunningDM.SetRxnInfoFilter;
292:            // findfirst RxnNo from RxnInfo
293:            if not RunningDM.FindFirstRxnNoRxnInfo then
294:              FirstRxnNo := False              // no longer first record
295:            else {else for if not FindFirstRxnNoRxnInfo }
296:              LastRxnNo := True;      // no record found
297:          end {end if FirstRxnNo then... }
298:          else {else for if First }
299:            LastRxnNo := RunningDM.FindNextRxnNoRxnInfo;
300:          // won't jump out in the middle so need to put another check here
301:          if not LastRxnNo then begin
302:            // try get current RxnNo
303:            RxnNoStr := RunningDM.RxnInfoQy.FieldByName('RXNNO').AsString;
304:            // initialise compound variables and filter RxnStoichQy with RxnNo
305:            icomp := 0;
306:            LastCompound := False;
307:            CompGenDM.SetRxnNoFilter (StrToInt(RxnNoStr));
308:            while not LastCompound do begin
309:              if icomp = 0 then begin       // i.e. if first compound for set RxnNo
310:                LastCompound := CompGenDM.FindFirstRxnStoich;
311:              end {end if icomp = 0 }
312:              else {else for if icomp = 0 }
313:                LastCompound := CompGenDM.FindNextRxnStoich;
314:              // won't jump out in the middle... double check (next) record found
315:              if not LastCompound then begin
316:                // try get current CompForm and CompStoich
317:                inc(icomp);
318:                CompNoStr := CompGenDM.RxnStoichQy.FieldByName('COMPNO').AsString;
319:                CompFormStr := CompGenDM.GiveCompFormula(StrToInt(CompNoStr),
320:                 ErrorStatus);
321:                if ErrorStatus then raise ECompFormNotFound.Create(
322:                 'CForm field not found in CompGen Table');
323:                CompStoichStr := CompGenDM.RxnStoichQy.
324:                 FieldByName ('STOICH_COEFF').AsString;
325:                // elucidate compound category: 0 - component, 1 - derived species
326:                if StrToInt(CompStoichStr) < 0                  // reactant
327:                  then CompCategory := 0                        // component
328:                else                                            // product
329:                  CompCategory := 1;                            // derived species
330:                // does it already exist in CompConc ? keyed by CompForm
331:                LocateOptions := []; // case sensitive and whole word only
332:                RunningDM.SelectAllCompConc;
333:                if not RunningDM.CompConcQy.Locate('CFORM', CompFormStr,
334:                   LocateOptions)  // has to be an open dataset to perform this
335:                then begin
336:                  MessageDlg ('Adding '+ CompFormStr + ' to CompConc table.',
337:                   mtInformation, [mbOK],0);
338:                  RGRunning.ItemIndex := 1;
339:                  with RunningDM.CompConcQy do begin
340:                    if RequestLive then RequestLive := False;
341:                    Close;
342:                    SQL.Clear;
343:                    SQL.Add (Format(SQLStrIns, [CompFormStr]));
344:                    ExecSQL;
345:                    SQL.Clear;
346:                  end; {end with RunningDM.CompConcQy do }
347:                end;  {end if not RunningDM.CompConcQy.Locate('CFORM', ... }
348:                // continue by adding other the rest of the information
349:                with RunningDM.CompConcQy do begin
350:                  if RequestLive then RequestLive := False;
351:                    RunningDM.RunningDB.StartTransaction;
352:                    Close;
353:                    SQL.Clear;
354:                    // insert CompCategoryString and make Current into CompConc
355:                    // first select record for current compound
356:                    SQLStrSel := ('SELECT * FROM COMPCONC WHERE (CFORM = "%s");');
357:                    SQL.Add (Format(SQLStrSel, [CompFormStr]));
358:                    ExecSQL;
359:                    // update record with current information
360:                    SQL.Clear;
```

```
361:                    SQLStrUpd := (' UPDATE COMPCONC SET CRNT_CALC = "%d", ' +
362:                     ' COMP_CATGRY = "%d" WHERE (CFORM = "%s")');
363:                    SQL.Add (Format (SQLStrUpd, [1, CompCategory, CompFormStr]));
364:                    ExecSQL;
365:                  // commit work to database
366:                  RunningDM.RunningDB.Commit;
367:                end; {end with RunningDM.CompConcQy do }
368:                RunningDM.SelectAllCompConc;
369:              end; {end if not LastCompound then }
370:            end; {end while not LastCompound do }
371:          end; { end if not LastRxnNo then begin }
372:        end; {end while not LastRxnNo do begin }
373:      // delete all non-current records in CompConcQy
374:      MessageDlg ('Deleting non-current compounds in CompConc.', mtInformation,
375:       [mbOK], 0);
376:      RunningDM.DelNonCurrentCompConc;
377:      except
378:        On E:EDatabaseError do begin
379:          MessageDlg(E.Message, mtError, [mbOK], 0);
380:          RunningDM.RunningDB.RollBack; // cancel any pending transactions
381:        end; {end On E:EDatabaseError do begin }
382:        On E:EDataError do  // should catch whole family of EDataError
383:          MessageDlg(E.Message, mtError, [mbOK], 0);
384:      end;  {end except }
385:    finally
386:      // clean up block
387:      with RunningDM.CompConcQy do begin
388:        if not RequestLive then RequestLive := True;
389:      end; {end with RunningDM.CompConcQy do begin }
390:      with RunningDM.RxnInfoQy do begin
391:        Filtered := False;
392:        Close;
393:        SQL.Clear;
394:        SQL.Add ('SELECT * FROM RXN_INFO ORDER BY RXNNO');
395:        Open;
396:        if not RequestLive then RequestLive := True;
397:      end;  {end with RunningDM.RxnIndoQy do }
398:      RunningDM.SelectAllCompConc;
399:    end;  {end finally }
400: end;  {end procedure TSelectionForm.BtnCheckCompoundsClick(Sender: TObject); }
401:
402: end.
```

```
 1: unit UnitsCheckUnit;
 2:
 3: { unit to contain routines for range and unit checking }
 4:
 5: interface
 6:
 7: type
 8:   { class declaration for methods to check ranges of values. }
 9:   TCheckUnits = class (TObject)
10:   private
11:     Message : string;
12:   public
13:     constructor Create;
14:     function Celsius (Value : single) : boolean;
15:     function TemperatureInterval (interval : single) : boolean;
16:
17:   end; {end TCheckUnits = class (TObject) }
18:
19: implementation
20: uses
21:   Dialogs, SysUtils,
22:   Constants;
23:
24: { constructor for TCheckUnits
25: ******************************************************************************* }
26: constructor TCheckUnits.Create;
27: begin
28:   Message := 'Nothing to report.';
29: end; {end constructor TCheckUnits.Create; }
30:
31: { function to check whether value falls within a feasible range for °C
32: ******************************************************************************* }
33: function TCheckUnits.Celsius (Value : single) : boolean;
34: begin
35:   if Value < MinTemp then begin
36:     Message := 'Temperature below minimum temperature limit ('
37:      + IntToStr(MinTemp) +'°C).';
38:     Result := False;
39:   end {end if Value < 0 then begin }
40:   else if Value > MaxTemp then begin
41:     Message := 'Temperature above maximum temperature limit ('
42:      + IntToStr(MaxTemp) +'°C).';
43:     Result  := False;
44:   end {end else if Value > 0 then begin }
45:   else
46:     // value in appropriate range
47:     Result := True;
48:   if not Result then MessageDlg (Message, mtwarning, [mbOK], 0);
49: end; {end function TCheckUnits.Celsius (Value : single) : boolean; }
50:
51: { function to check whether interval > 1 °C
52: ******************************************************************************* }
53: function TCheckUnits.TemperatureInterval (interval : single) : boolean;
54: begin
55:   if interval < 1 then Message := 'Temperature interval is less than 1 degree.';
56: end; {end function TCheckUnits.TemperatureInterval (Value : single) : boolean; }
57:
58:
59:
60:
61: end.
```

```
 1: unit UserClassUnit;
 2:
 3: interface
 4: type
 5:
 6:   { TUser class declaration
 7:   ********************************************************************** }
 8:   TUser = class (TObject)
 9:     UserIDStr : string;
10:   public
11:     RecalcFlag : boolean;
12:     constructor Create (User : string);
13:     procedure StoreUserID ( username : string );
14:     function PassUserID : string;
15:
16:   end; {end   TUser : class (TObject) }
17:
18: implementation
19:
20: { TUser constructor
21: ********************************************************************** }
22: constructor TUser.Create (User : string);
23: begin
24:   UserIDStr  := User;
25:   RecalcFlag := True;    // default to recalculate
26: end;  {end constructor TUser.Create (User : string); }
27:
28: { procedure to store TUser.UserID
29: ********************************************************************** }
30: procedure TUser.StoreUserID ( username : string );
31: begin
32:   UserIDStr := username;
33: end; {end procedure TUser.StoreUserID ( username : string ); }
34:
35: {function to pass TUser.UserID to the calling routine
36: ********************************************************************** }
37: function TUser.PassUserID : string;
38: begin
39:   Result := UserIDStr;
40: end; {end function TUser.PassUserID : string; }
41:
42:
43: end.
```