THE MEASUREMENT OF FLOW VELOCITY DISTRIBUTION

by

MICHAEL DANN

Submitted to the University of Cape Town in part fulfilment of the requirements for the degree of Master of Science in Engineering.

September 1981.
The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.
DECLARATION

I wish to certify that this thesis, unless indicated to the contrary in the text, is my own work.

M.S. DANN

[Signature]
ABSTRACT

A method for the improvement of the range and accuracy achieved by the crosscorrelation flowmeter is investigated. The principles of the flowmeter operation and fundamental digital signal processing techniques are reviewed. The process of the Fourier transform deconvolution is investigated. Computer simulation of the flow system is described and is shown to require impractical amounts of computer time to achieve the necessary averaging times. Consequently, correlation and velocity profile measurements are made from an experimental flow rig. A waveform analysis program is used to analyse these measurements. The Fourier transform deconvolution is shown in this case to have poor noise immunity. For this reason, an alternative method of Bayesian deconvolution is investigated. The correlation functions measured from the experimental flow rig are deconvolved using the Bayesian deconvolution algorithm. The resulting transit time distribution is shown to converge to the transit time distribution obtained from the velocity profile measurements. From an analysis of the flow signals the velocity distribution of the flow may thus be found.
I would like to thank my supervisor, Mr. J.R. Greene, for his interest in the project and for the many useful discussions and suggestions.

I would also like to thank Dr. D.E.H. Naudé for initiating this research and his enthusiasm and guidance.

The National Institute for Metallurgy has provided generous financial support for the project.

I am indebted to all the members of staff of the Department of Electrical Engineering at UCT for their comments and assistance.

Finally, thanks also go to my father, Mr. R.A. Dann, for reading and correcting the draft.
# CONTENTS

## CHAPTER 1

1.1 Introduction .................................................. 1  
1.2 Preliminary Mathematics ........................................ 2  
1.3 The Ultrasonic Crosscorrelation Flowmeter ..................... 3  
1.4 The Flow System ............................................... 4  
1.5 The Project in Perspective ..................................... 7  
References ..................................................................... 9  

## CHAPTER 2: DIGITAL SIGNAL PROCESSING

2.1 Sampling .......................................................... 12  
2.2 Windowing ........................................................ 13  
2.3 The Discrete Fourier Transform ................................ 16  
2.4 The Fast Fourier Transform ..................................... 17  
2.5 Discrete Correlation and Convolution ......................... 19  
2.6 Numerical Deconvolution ....................................... 21  
2.7 Applying the Fast Fourier Transform using a Microprocessor 26  
References ..................................................................... 28  

## CHAPTER 3: SOFTWARE AND EXPERIMENTAL DEVELOPMENT

Introduction ..................................................................... 31  
3.1 Flow System Modelling and Flow Signal Simulation .......... 32  
3.1.1 Introduction ..................................................... 32  
3.1.2 Program Specifications .......................................... 33  
3.1.3 Program Description ............................................ 33  
3.1.4 Simulation Results .............................................. 34  
3.1.5 Conclusion ....................................................... 38
### CHAPTER 4: DECONVOLUTION

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Deconvolution of Arbitrary Autocorrelation and Crosscorrelation Functions</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Deconvolution of the Autocorrelation and Crosscorrelation Functions obtained from the Computer Simulation of the Flow</td>
<td>49</td>
</tr>
<tr>
<td>4.3 Deconvolution of the Autocorrelation and Crosscorrelation Functions Measured from the Experimental Flow Rig</td>
<td>50</td>
</tr>
<tr>
<td>4.4 A Further Deconvolution Experiment</td>
<td>51</td>
</tr>
<tr>
<td>4.5 An Analysis of the Noise Effects on the Deconvolution Result</td>
<td>52</td>
</tr>
<tr>
<td>4.6 Summary</td>
<td>54</td>
</tr>
</tbody>
</table>

### CHAPTER 5: BAYESIAN DECONVOLUTION

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>55</td>
</tr>
<tr>
<td>5.2 Implementation</td>
<td>57</td>
</tr>
<tr>
<td>5.3 Results</td>
<td>58</td>
</tr>
</tbody>
</table>
CHAPTER 6: DISCUSSION, RECOMMENDATIONS AND CONCLUSIONS

6.1 Summary of Results 61
6.2 Further Work 63
6.3 Conclusions 67

APPENDICES

Appendix 1: A Signal Processing Microprocessor 70
Appendix 2: Simula V2.0 78
Appendix 3: Wavepack 84
Appendix 4: A Proof on the Deconvolution Result 134
Appendix 5: Deconvolution of Correlation Functions Obtained from the Simulated Flow System 136
Appendix 6: Fourier Transform Deconvolution in the Presence of Noise 138
1.1 INTRODUCTION

In many industries it has become increasingly important to be able to measure accurately the volume flow rate of a variety of fluids. This variety encompasses abrasive solids in suspension to blood flow in arteries.

The ultrasonic crosscorrelation flowmeter has been well developed and is at this stage suitable for flow measurement of industrial slurries. This flowmeter has at present a dynamic range of about 10:1 and an accuracy of 1% once calibrated for the particular velocity profile characterising the flow. The dynamic range over which the flow rate can be measured is limited in that large errors may be introduced if the particular flow velocity for which it has been calibrated were to change. If, for example, the instrument is calibrated for fully developed turbulent flow and the flow changes to fully developed laminar flow (possibly outside the 10:1 measurement range) errors of up to 33% may result (Reference 1.1).

The aim of this project is to investigate a technique to improve the range of operation and the accuracy of the crosscorrelation flowmeter. By determining the velocity
distribution in the flow being measured, some correction may be made to eliminate errors introduced due to variations of the velocity profile. It is proposed that this may be achieved by suitably processing the signals provided by the crosscorrelation flowmeter.

1.2 PRELIMINARY MATHEMATICS

The Fourier transform of a function of time \( h(t) \), satisfying the Dirichlet conditions (given in Reference 1.2) is:

\[
H(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt \quad \ldots \quad 1.1
\]

and the inverse Fourier transform is described by:

\[
h(t) = \int_{-\infty}^{\infty} H(f) e^{j2\pi ft} df \quad \ldots \quad i.2
\]

The concept of crosscorrelation of two functions may be described as a measure of the similarity between the functions for different values of delay between them. This is more exactly described by:

\[
\Phi_{xy}(\tau) = \int_{-\infty}^{\infty} x(\theta) y(\tau + \theta) d\theta = X(t) \ast y(t) \quad \ldots \quad 1.3
\]

where \( \Phi_{xy}(\tau) \) is the crosscorrelation function of \( x(t) \) and \( y(t) \) and \( \ast \) represents the crosscorrelation operator.
Figure 1.1 General schematic of the crosscorrelation flowmeter.

Figure 1.2 Typical flow signals and their crosscorrelation.
1.3 THE ULTRASONIC CROSSCORRELATION FLOWMETER

Ultrasonic crosscorrelation flowmetering is achieved by passing two ultrasonic beams, some distance apart, through the flowing stream. Variations in density and eddies present in the material being transported modulate the amplitude and frequency of the received signal. The flow signals, one from the upstream and the other from the downstream transducer, may be derived from either variations in amplitude or variations in frequency of the received signals. Variations in density and eddies at the upstream transducer appear at the downstream transducer a short while later having undergone some modifications in transit between the transducers. The eddies, however, tend to have a finite lifetime and their modulation effect on the ultrasonic beam changes exponentially with time. The positions of the eddies and the variations in their relative density is modified by the velocity profile of the flow. If the crosscorrelation of the two flow signals is calculated then the resulting function $\varphi_{xy}(\tau)$ will exhibit a global maximum at the delay at which the two signals are separated. The velocity of the material being transported may be found since the delay of $\beta$ seconds is a measure of the time taken for this material or fluid to travel the distance between the two transducers and the velocity is inversely proportional to this delay.
1.4 THE FLOW SYSTEM

Consider a pipe as shown in Figure 1.1 filled with a homogeneous, stationary fluid. The outputs of the two demodulators will then be zero. Now a particle of higher density than the fluid in the pipe, travelling with a velocity \( v_0 \), in the centre of the pipe, passes through the upstream ultrasonic beam. This causes a change in the upstream flow signal. The same change will appear in the downstream flow signal as the particle passes through the downstream ultrasonic beam. Crosscorrelating these two flow signals will produce a maximum in the crosscorrelation function at the delay corresponding to the time taken for the particle to travel from one beam to the other. (When considering the flow of an eddy a modification of the eddy occurs while travelling between the ultrasonic beams due to the eddy decay. This has the effect of reducing the height of the crosscorrelation function as the distance between the transducers is increased.) Assume now that when the aforementioned particle enters the upstream beam, another particle travelling at a lower velocity \( v_i \) also crosses the upstream beam. They will cross the downstream beam at different times and the resulting crosscorrelation of the two flow signals will exhibit two peaks, one corresponding to velocity \( v_0 \) and the other to velocity \( v_i \).

Instead of particles, consider now a flow channel divided
ULTRASONIC TRANSDUCERS

PIPE WALLS

DIRECTION OF FLOW

x(t) → g(t) → y(t)

Figure 1.4 The flow system.
by a partition into two subchannels. If the fluid in 
the two subchannels is travelling at \( V_1 \) and \( V_2 \), then 
the crosscorrelation of the upstream and downstream 
flow signals will exhibit two peaks at delays correspond­
ing to the velocities \( V_1 \) and \( V_2 \). The relative heights 
of the two peaks will depend on the signal strengths due 
to each subchannel. The signal strengths are directly 
related to the respective subchannel widths. More simply 
- the wider the subchannel with fluid at velocity \( V_i \), 
the higher the peak of the crosscorrelation function 
corresponding to this velocity. Now a parabolic velocity 
profile representing laminar flow can be represented by 
increasing the number of subchannels. The individual 
peaks of the resulting crosscorrelation will then merge 
to form a composite "smeared" crosscorrelation function 
the shape of which must contain information about the 
velocity profile.

This leads on to the idea of the flow between the two 
transducers as a system with the input being the upstream 
flow signal and the output the downstream flow signal. 
The impulse response of the system may be interpreted as 
the transit time distribution (Reference 1.3). Thus:

\[
    w(t) = g(t)
\]

where \( w(t) \) is the transit time distribution. The 
determination of the system impulse response is thus a 
system identification problem.
If the system is linear then the output $y(t)$ is the convolution of the input $x(t)$ and the system impulse response $g(t)$.

$$\quad y(t) = \int_{-\infty}^{\infty} x(\theta) g(t - \theta) d\theta \quad \ldots 1.4$$

or

$$y(t) = x(t) * g(t)$$

where * represents the convolution operator.

Various methods exist for finding the impulse response of a system, the simplest being to excite the system with a deterministic signal such as a step, ramp or impulse and observe the output. Random signals which may be applied or occur naturally may also be used to excite the system. Then the output of the system may be described by:

$$\varphi_{xy}(\tau) = \int_{-\infty}^{\infty} \varphi_{xx}(\theta) g(\tau - \theta) d\theta$$

where $\varphi_{xy}(\tau)$ is the cross-correlation of the upstream and downstream flow signals

$\varphi_{xx}(\tau)$ is the autocorrelation of the upstream flow signal

$g(t)$ is the system impulse response.

(This expression may be derived by Fourier transforming both sides of equation 1.4 above and assumes the signals to be both stationary and ergodic).
The velocity distribution may be derived from the transit time distribution:

\[ W(v) = \frac{w(t)}{dv/dt} = w(t) \frac{t^2}{L} \ldots 1.5 \]

Thus the correlation measurements should yield not only the mean velocity but also the complete velocity distribution independent of the transducer properties.

1.5 THE PROJECT IN PERSPECTIVE

The field of ultrasonic crosscorrelation flowmetering has been fairly well researched (References 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11) and of particular interest is the work done by Leitner (Reference 1.12). He has developed a crosscorrelation flowmeter using a microprocessor controlled two-point difference correlator. The aim of my work is to apply digital signal processing techniques to improve the range and accuracy of this microprocessor controlled correlator. Ultimately one may envisage an ultrasonic flowmeter incorporating a correlator and microcomputer which calculates the autocorrelation and crosscorrelation functions of the flow signals and then deconvolves these two functions to yield the flow system impulse response. This impulse response may then undergo further processing to yield the entire velocity distribution or simply a correction factor for the measured flowrate.
The task at hand is thus to develop and test a method for performing the required deconvolution.
REFERENCES - CHAPTER 1


Figure 2.1 The sampling process with $\frac{1}{T} > 2f_c$.

Figure 2.2 The sampling process with $\frac{1}{T} < 2f_c$. 
2.1 SAMPLING

Let \( \hat{h}(t) \) be a sampled representation of the continuous function \( h(t) \). This sampled function \( \hat{h}(t) \) may be considered as the continuous function \( h(t) \) multiplied by an infinite sequence of unit impulses.

\[
\hat{h}(t) = h(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t-nt) = \sum_{n=-\infty}^{\infty} h(t) \delta(t-nT) = h(nT)
\]

for \( n \in \mathbb{I} \) and \( T \) the sampling interval.

The Fourier transform of \( \hat{h}(t) \) may be found by noting that multiplication in the time domain is equivalent to convolution in the frequency domain, thus:

\[
\hat{H}(f) = H(f) \ast \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T})
\]

where \( H(f) \) and \( \hat{H}(f) \) are the Fourier transforms of \( h(t) \) and \( \hat{h}(t) \) respectively. These operations are represented graphically in Figure 2.1.

In this diagram we see that \( 1/T > 2f_c \) where \( f_c \) is the highest frequency component present in the signal \( h(t) \). If the sampling interval were increased to a value where sampling took place less often than twice each cycle of
Figure 2.3 Application of a rectangular window function.
the highest frequency component $f_c$, then the phenomenon known as aliasing would occur. Aliasing implies an overlapping and distortion of the desired Fourier transform of the sampled function. This is shown in Figure 2.2.

To eliminate aliasing one must ensure that $T > 2 f_c$. This is usually achieved by low pass filtering $h(t)$ before sampling at a frequency of anything between three and five times the cut off frequency of the anti-aliasing filter.

2.2 WINDOWING

Ultimately we wish to apply the relations defined for continuous functions to sampled waveforms using a computer. It is obviously impossible to deal with infinite records of data so a finite length sample of the data must be used. The most obvious method is simply to select $N$ data starting at time $t_1$ and ending at $t_2$. This is the same as applying a rectangular window function to the infinite sequence. As multiplication in the time domain is equivalent to convolution in the frequency domain, the resulting frequency domain representation has some ripple added to it as shown in Figure 2.3.

By examining Figure 2.3 one can appreciate the desire to keep the sidelobes of the Fourier transformed window function as small as possible while keeping the width as
narrow as possible. This can be done by either using many data points, that is, making $T_0$ large, or by a wise choice of the window function. The selection of window functions has been well researched and Childers and Durling (Reference 2.1) refer to this field as that of "window carpentry". Some of the more common window functions and their transforms are given here.

**Rectangular Window**

\[
w(t) = A \left\{ u(t+\frac{T_0}{2}) - u(t-\frac{T_0}{2}) \right\}
\]

\[
w(f) = A T_0 \frac{\sin \pi f T_0}{\pi f T_0}
\]

**Triangular Window**

\[
w(t) = A \left\{ 1 - \frac{|t|}{T_0/2} \right\}, |t| < \frac{T_0}{2}
\]

\[
w(f) = \frac{A T_0}{2} \left\{ \frac{\sin \pi/2 f T_0}{\pi/2 f T_0} \right\}^2
\]

\[= 0 \quad , \quad |t| > T_0/2\]
Hannning Window (after Von Haan)

\[
W(t) = A \cos^2 \frac{\pi t}{T_0} \quad \text{for} \quad t < \frac{T_0}{2}
\]

\[
= A \left\{ 1 + \cos \frac{2\pi t}{T_0} \right\} \quad \text{for} \quad t < \frac{T_0}{2}
\]

\[
= 0 \quad \text{for} \quad |t| > \frac{T_0}{2}
\]

Hamming Window

\[
w(t) = 0.54 + 0.46 \cos \frac{2\pi t}{T_0} \quad \text{for} \quad |t| < \frac{T_0}{2}
\]

\[
= 0 \quad \text{for} \quad |t| > \frac{T_0}{2}
\]

\[
W(f) = \frac{A T_0}{2} \left[ 0.54 \pi^2 - 0.08 (\pi f T_0)^2 \right] \frac{\sin \pi f T_0}{f T_0} \frac{1}{\left( \pi^2 - \pi^2 f^2 T_0^2 \right)}
\]
Figure 2.4 Development of the discrete Fourier transform.
The Cosine, Blackman, Tukey, Papoulis, Parzen, Kaiser and Dolph-Chebyshev are other window functions, each with their own various advantages and disadvantages.

2.3 **THE DISCRETE FOURIER TRANSFORM**

Let \( h(t) \) be represented by the sequence of \( N \) samples \( h(nT) \), \( 0 \leq n \leq N-1 \), where \( T \) is the sampling interval in the time domain. Similarly let \( H(\omega) \) be represented by the \( N \) samples \( H(k\omega) \), \( 0 \leq k \leq N-1 \) where \( \omega \) is the increment between samples in the frequency domain. The discrete Fourier transform may be written as an adaptation of equation 1.1:

\[
H(k\omega) = \sum_{n=0}^{N-1} h(nT) e^{-j\omega Tn} \quad k = 0,1,2 \ldots N-1
\]

where \( \omega = \frac{2\pi}{NT} \)

The graphical development of the discrete Fourier transform (DFT) along with the effects that sampling, aliasing and windowing produce is shown in Figure 2.4. Sampling the original waveform, the first operation shown in Figure 2.4, may give rise to an aliased frequency function. The next operation to be performed is time domain truncation and this introduces ripple into the frequency domain function. The final operation is frequency domain sampling which causes the time domain
Figure 2.5 Splitting the sixteen point sequence into two eight point sequences.
function to be periodic with a period defined by the N points of the original function after sampling and truncation.

2.4 THE FAST FOURIER TRANSFORM

The fast Fourier transform (FFT) is an efficient algorithm for calculating the discrete Fourier transform. To obtain an idea of the increase in efficiency, if \( N = 1024 \) (where \( N \) is the number of samples representing the data to be transformed), then the computational reduction over the DFT is more than 200 to 1. Almost every textbook relating to signal processing contains some explanation of the development of the FFT so discussion of the subject here will be limited.

The DFT was defined in expression 2.1 to be:

\[
H(k\Omega) = \sum_{n=0}^{N-1} h(nT) e^{-j\Omega Tn}
\]

where \( \Omega = \frac{2\pi}{NT} \) and \( k = 0, 1 \ldots N-1 \)

Let \( W = e^{-j\Omega T} = e^{-j\frac{2\pi}{N}} \) then we may write:

\[
H(k\Omega) = \sum_{n=0}^{N-1} h(nT) W^{nk}
\]

Assuming that we wished to transform the sixteen point sequence shown in Figure 2.5, we may split the sixteen
point sequence into two eight point sequences, where
\[ g(\ell T) = h(2\ell T) \quad \text{and} \quad f(\ell T) = h((2\ell + 1)T) \]
where \( \ell = 0, 1, 2 \ldots N/2 - 1 \).

The DFTs of these two sequences are also \( N/2 \) point sequences and may be written thus:

\[
\begin{align*}
G(k\Omega) &= \sum_{\ell=0}^{N/2-1} g(\ell T)(W^2)^{\ell k} \\
F(k\Omega) &= \sum_{\ell=0}^{N/2-1} f(\ell T)(W^2)^{\ell k}
\end{align*}
\]

The DFT of the entire sequence may be written as:

\[
H(k\Omega) = \sum_{\ell=0}^{N/2-1} [g(\ell T)W^{2\ell k} + f(\ell T)W^{(2\ell+1)k}]
\]

and, rewriting this expression, we get:

\[
H(k\Omega) = \sum_{\ell=0}^{N/2-1} g(\ell T)(W^2)^{\ell k} + W^k \sum_{\ell=0}^{N/2-1} f(\ell T)(W^2)^{\ell k}
= G(k\Omega) + W^k F(k\Omega)
\]

Using this method of splitting the original sequence into two shorter sequences reduces the number of calculations required to compute \( H(k\Omega) \). The number of operations required to compute \( G(k\Omega) \) and \( F(k\Omega) \) directly is \((N/2)^2\) for each and combining them to give \( H(k\Omega) \) requires \( N \) operations giving a total of \( N + N^2/2 \) operations. Computing \( H(k\Omega) \) directly would require \( N^2 \) operations so even at this stage a saving is evident. The same
process which was applied to \( H(k\omega) \) may now be applied to \( G(k\omega) \) and \( F(k\omega) \) in turn, giving four four-point sequences, so reducing still further the number of calculations required. Repeating this process until one point DFTs are required reveals the FFT process since the DFT of a single point is the point itself and thus the whole Fourier transform calculation has been reduced to one of a sequence of complex multiplications and additions.

The approach outlined above is known as decimation in time. Several other algorithms which exploit the properties of the data to be transformed have been developed. Brigham in Reference 2.2, develops the FFT as a sequence of matrix factorisations which provides insight into the analysis of the signal flow graph representation of the FFT. When dealing with data records where the number of points \( N \neq 2^m \) where \( m \) is an integer then \( N \) is factorised and the transform is reduced to elementary transforms of the dimension of the lowest factor. This process is described by Singleton in Reference 2.3.

2.5 DISCRETE CORRELATION AND CONVOLUTION

Discrete correlation and convolution do not necessarily require the use of the FFT but in the interests of efficiency the FFT is usually employed.
The correlation of two continuous functions was described previously by equation 1.3

\[ \varphi_{xy}(t) = \int_{-\infty}^{\infty} x(\theta) y(t+\theta) d\theta \]

When considering the sampled or discrete signals \( x(nT) \) and \( y(nT) \) (\( 0 \leq n \leq N-1 \)), we may rewrite equation 1.3 such that:

\[ \varphi_{xy}(kT) = \sum_{n=0}^{N-1} x(nT) y[(k+n)T] \quad \cdots 2.2 \]

Convolution is of more interest since the ultimate aim is to deconvolve two sampled waveforms. The convolution of two sampled waveforms, with reference to equation 1.4, may be written as:

\[ y(kT) = \sum_{n=0}^{N-1} g(nT) x[(k-n)T] \quad \cdots 2.3 \]

As for the correlation of two discrete waveforms, it can be shown that the computation of equation 2.3 may be simplified by using the DFT. In Reference 2.4 Hunt proves the discrete convolution theorem using matrix theory. The approach he adopts is to diagonalise the convolution matrix operator and in doing so the eigen values of this operator are shown to be identical to the discrete Fourier transform.
Figure 2.6 The system.
2.6 NUMERICAL DECONVOLUTION

Consider the system with impulse response $g(nT)$, output $y(nT)$ and input $x(nT)$ as shown in Figure 2.6. Neglecting the noise term we have:

$$y(t) = \int_{-\infty}^{\infty} g(\theta) \cdot x(t-\theta) d\theta \quad ... \ 2.4$$

The determination of $g(\theta)$ is thus a system identification problem. One approach to the identification of $g(nT)$ is the division of the Fourier transform of the output by the Fourier transform of the input, that is:

$$g(nT) = \frac{\text{FFT} \{y(nT)\}}{\text{FFT} \{x(nT)\}} \quad ... \ 2.5$$

An alternative to the above is a formulation of equation 2.5 using a matrix representation of the variables is given in Reference 2.5 and outlined below.

Let $x(t)$, a continuous function of time, be considered constant between sampling periods. Each sample of $x(t)$ takes the value at the beginning of the sampling period, that is:

$$x(t) = x(nT) \quad \text{for} \ nT < t < (n+1)T$$

Similarly assuming $g(t)$ is constant over the sample interval but with the value at the midpoint of the interval so that:
\[
g(t) = g\left(\frac{2n+1}{2} \cdot T\right) \quad \text{for } nT < t < (n+1)T
\]

Rewriting equation 2.4 assuming \(x(t) = 0\) for \(t < 0\):

\[
y(t) = \int_{0}^{\infty} g(\theta) x(t-\theta) d\theta
\]

and if \(t = nT\), then:

\[
y(nT) = T \sum_{i=0}^{n-1} g\left(\frac{2n-1}{2} \cdot T - iT\right) x(iT)
\]

so for \(n = 1\), \(y(T) = T[g(T/2) \cdot x(0)]\)

\(n = 2\), \(y(2T) = T[g(3T/2) \cdot x(0) + g(T/2) \cdot x(T)]\)

\[\vdots\]

\(n = N\), \(y(NT) = T[g(2N-1/2 \cdot T) \cdot x(0) + \ldots\)

\[
\ldots + g(T/2) \cdot x(N-1)]
\]

and setting:

\[
y = \begin{bmatrix}y(T) \\ y(2T) \\ y(3T) \\ \vdots \\ y(NT)\end{bmatrix}; \quad g = \begin{bmatrix}g(T/2) \\ g(3T/2) \\ g(5T/2) \\ \vdots \\ g(2N-1/2 \cdot T)\end{bmatrix}
\]
\[ X = \begin{bmatrix}
x(0) & 0 & 0 & \ldots \\
x(T) & x(0) & 0 \\
x(2T) & x(T) & x(0) \\
\vdots & \vdots & \vdots \\
x[(N-1)T] & x[(N-2)T] & \ldots & \ldots & x(0)
\end{bmatrix} \]

and we may write:

\[ y = T \cdot X \cdot g \]

We now need to solve for \( g \) in the above equation.

If \( x(0) \neq 0 \) (a finite shift in time may be required to achieve this), then \( \det X = [x(0)]^N \neq 0 \) and \( X \) is non-singular. So:

\[ g = \frac{1}{T} \cdot X^{-1} y \] ... 2.6

Thus \( g \) may be computed using matrix methods or alternatively each element of the above matrix equation may be written in terms of the others:

\[ h_n = \frac{i}{x(0)} \left\{ \frac{y(nT)}{T} - \sum_{i=1}^{n-1} g_{n-i} \cdot x(iT) \right\} \]
where $h_n = h \left( \frac{2n-1}{2} \right) T$ and $h_1 = \frac{y(T)}{T \cdot x(0)}$

An iterative procedure may be developed if one considers the system impulse response to be, as a first approximation, equal to the system output. This initial estimate is then corrected by adding the difference between the system output and the convolved current approximation of the impulse response. We may therefore write the recursive relation:

$$\hat{g}_{n+1}(t) = \hat{g}_n(t) + \left\{ y(t) - \int_{-\infty}^{\infty} x(t-\theta) \hat{g}_n(\theta) d\theta \right\} \ldots 2.7$$

where $\hat{g}_{n+1}(t)$ is the current estimate of $g(t)$.

Transforming this into the frequency domain we have:

$$\hat{G}_{n+1}(w) = \hat{G}_n(w) + Y(w) - X(w) \cdot \hat{G}_n(w)$$

with $\hat{G}_0(w) = Y(w)$

This technique forms the basis of the work described by Balslev et al. in their paper (Reference 2.6) on the deconvolution of experimentally broadened spectra in the field of spectroscopy. Kennet et al. (in Reference 2.7) also use a similar iterative procedure to achieve improved spectral resolution by applying Bayesian deconvolution.
In the field of seismic exploration several deconvolution methods are being examined with varying degrees of success. Among these are predictive deconvolution, homomorphic deconvolution, Kalman filtering and deterministic deconvolution. An overview of these methods is given in Reference 2.8 where Arya and Holden point out that homomorphic deconvolution has still to be fully developed and they also note that at the time of their writing (1978), there was no reported application of Kalman filtering to seismic deconvolution.

Eisenstein and Cerrato (in Reference 2.9) describe a deconvolution technique which they use to improve electrocardiograms by removing the effects of the measuring system. In their work they also use an iterative technique to improve the estimate of the system impulse response except that a model of the system to be identified is formed and the parameters varied to minimise the error between the real system output and the estimated output.

Several researchers claim success using the Fourier transform techniques described by equation 2.5. Among them are Nabel and Mundry (Reference 2.10) in the field of ultrasonic testing and echo evaluation and Kuchel (in Reference 2.11) in his work on a flow assay device for the study of steady-state enzyme kinetics.
The work of this dissertation will be concentrated on the approach described by equation 2.5.

2.7 APPLYING THE FAST FOURIER TRANSFORM USING A MICROPROCESSOR

The importance of the fast Fourier transform arises due to the fact that most deconvolution methods seem to be based upon the FFT. To obtain some idea of the feasibility of computing the FFT on a microprocessor, the hardware was developed (see Appendix 1) and the task of writing the necessary software was set as an undergraduate thesis co-supervised by me. The hardware comprised a standard Intel SDK85 development kit, 2K of additional random access memory (RAM), 1K of erasable programmable read only memory (EPROM), two eight bit analogue to digital converters and a display control unit which permitted the graphical display on an oscilloscope of data. The software to enable the sampling of an analogue waveform and the subsequent computation of its Fourier transform using the fast Fourier transform was developed (Reference 2.12). The execution time required for a 256 point FFT, using eight bit number representation, was found to be approximately two seconds. Using a sixteen bit number representation, a 256 point transform takes approximately nine seconds. These execution times are excellent considering the very limited arithmetic capabilities of the 8085 microprocessor. It became
apparent that to reduce truncation error to an acceptable level representation of the data using 16 bits was imperative. Computation time would be reduced considerably if a present generation 16 bit microprocessor were employed. The use of a microprocessor to calculate the FFT was successful and a dedicated microprocessor, incorporated into the existing flowmeter, could be used to perform the Fourier transform analysis and possibly the entire deconvolution.
REFERENCES - CHAPTER 2


2.6 Balslev, J.E. et al.: Noise Amplification and Resolution Improvement in Deconvolution of Experimental Spectra.


2.12 Weeks, D.P.: Microprocessor-Based Fast Fourier Transform.

 Bibliography

Harris, R.W. and Ledwidge, T.J.: Introduction to Noise Analysis.


SOFTWARE AND EXPERIMENTAL DEVELOPMENT

The actual mechanism of multiphase fluid flow is fairly complex and when considering the generation of flow signals in the presence of a curved velocity profile, the formulation of the problem using a mathematical analysis becomes very difficult. The comparison of the computed velocity distribution and the actual velocity distribution may be made by setting up a flow system from which the flow signals and measurements of the velocity profile are obtained. The flow signals are analysed to find a calculated velocity distribution using the techniques described earlier. This velocity distribution or alternatively the transit time distribution is then compared with the measured velocity or transit time distribution to determine the accuracy and/or feasibility of the proposed techniques.

Investigating methods for finding the velocity distribution in multiphase flow requires the ability to measure and control accurately the velocity profile in the flow of interest. Measurement and control of the velocity profile is by no means a simple matter and this led to the idea of simulating the flow system on a computer. This idea is attractive, bearing in mind that the analysis, in the development stage, will almost certainly be carried
Figure 3.1 Obtaining the flow signals optically.
out on a computer. To eliminate the effects of any assumptions made when modelling the flow system, it would also be desirable to analyse real flow signals.

The development thus consists of three parts:

1. The flow system modelling and flow signal generation;
2. The experimental flow rig; and
3. The analysis of the flow signals.

3.1 FLOW SYSTEM MODELLING AND FLOW SIGNAL SIMULATION

3.1.1 Introduction

The ultrasonic transmitter-receiver pair used to generate the flow signals from variations in the flow each exhibit directivity patterns characteristic of ultrasonic transducers. This gives rise to a complex ultrasonic beam across the pipe. It becomes difficult to determine exactly when any particle or eddy enters the beam. Cross-coupling between the upstream and downstream transducers may also occur. To minimise these complexities optical transducers producing a well-defined beam across the fluid are used. It is reasonable to assume that the fluid contains small particles or bubbles which modulate these light beams to produce the flow signals. For the purposes of modelling this optical
Figure 3.2 The computer representation of the pipe and transducers.
crosscorrelation flowmeter, it is assumed that the beam is wider than the largest particle in the fluid. These particles may then be seen to either scatter or absorb the light with the amount of scattering or absorption depending on the particle size.

3.1.2 Program Specifications

The simulation program is to generate flow signals and calculate the autocorrelation and crosscorrelation functions of the upstream and downstream flow signals. The user should specify the prevailing velocity profile, the transducer spacing and eddy decay time constant.

3.1.3 Program Description

The actual flow signal generation is achieved by using a two-dimensional array as shown in Figure 3.2. The squares of the data array represent the fluid elements of the real pipe. These individual elements are assumed to be travelling at constant velocity. The elements of the array are represented by memory locations of the computer and are filled with normally distributed random numbers to model the random variations in the fluid. The flow signals are then generated by summing the random numbers in the array which are in the field of view of the upstream or downstream transducers. In Figure 3.2,
Figure 3.3 An example of the velocity profile effects.

Figure 3.4 A simulated flow signal and its spectral density.
the beamwidth has been chosen to be 10 fluid elements wide. This means that the beamwidth is at least ten times the width of the smallest particle.

The data in the array is shifted along the pipe and the next sample of the flow signals is generated. The number of shifts the fluid elements in the subchannels undergo each time interval is determined by the specified velocity profile. Only the length of pipe between the transducers is of interest so when shifting a row or subchannel along, new numbers are moved in at one end and dropped off at the other. The eddy decay effect is achieved by adding uncorrelated noise to the numbers in the individual fluid elements so that the crosscorrelation coefficient is inversely proportional to transducer spacing. The amount of noise added increases exponentially with time.

3.1.4 Simulation Results

An example of a typical flow signal and its spectral density is given in Figure 3.4. The folding about the centre of the spectrum is a property of the discrete Fourier transform used in the computation of the spectral density (refer to Section 2.3). This spectral density compares well with those given in Reference 3.1. An analysis of this flow signal was carried out to compare the characteristics with those of real flow signals.
Figure 3.5 The velocity profile.

Figure 3.6 The ACF and CCF of the simulated flow signals using only one sample of the flow signals.

Figure 3.7 The ACF and CCF of the simulated flow signals using 100 samples of the flow signals.
A chi squared goodness of fit test was performed to determine whether the amplitude of the flow signal is normally distributed. The results showed this assumption to be valid at the 50% significance level. One would expect the amplitude to be normally distributed because the signals are simulated by forming a summation of normally distributed random numbers (by the central limit theorem).

The simulation was carried out using an infinite eddy decay time constant and the velocity profile shown in Figure 3.5. The resulting crosscorrelation function (shown in Figure 3.6) is expected to have peaks corresponding only to the two velocities present. The heights of the two peaks should be equal considering the volume of fluid travelling at these two velocities is the same. This crosscorrelation has been computed using only one flow sample and it is felt that a better estimate may be obtained if it were computed from several flow samples and the average taken. Figure 3.7 shows the autocorrelation and crosscorrelation functions computed using 100 flow signal samples and the velocity profile shown in Figure 3.5.

The simulation results using this stepped velocity profile may be compared with the correlation functions obtained from the following simple experiment. The experimental layout is shown schematically in Figure 3.8.
Figure 3.8 The flow system.

Figure 3.9 The autocorrelation and cross-correlation functions obtained from the system shown in figure 3.8 above.
The flow signals are generated using a Gaussian noise generator. The delay between the upstream and downstream flow signals and the simulation of the stepped velocity profile is achieved using a tapped delay line. The resulting autocorrelation and crosscorrelation functions are given in Figure 3.9.

The question arises as to the relation between the averaging time or number of flow signal samples used and the variance of the correlation functions. Examining only the autocorrelation function, we may rewrite equation 1.3 with the limits of integration expressed slightly differently:

$$\theta_{xx}(\tau) = \lim_{T \to \infty} \frac{1}{T} \int_{0}^{T} x(\theta + \tau)x(\theta) d\theta$$

The averaging time $T$ must be finite for any practical measurement and it can be shown (in Reference 3.2) that the expression for the variance of the autocorrelation estimate of bandlimited white noise of bandwidth $B$, zero mean value and over an averaging time $T$ is given conservatively by:

$$\text{Var} [\hat{\theta}_{xx}(\tau)] = \frac{1}{2BT} \left\{ \theta_{xx}^{2}(0) + \theta_{xx}^{2}(\tau) \right\}$$

where $\hat{\theta}_{xx}(\tau)$ is the estimated value of the true autocorrelation function $\theta_{xx}(\tau)$. It is further stipulated that for the above expression to hold the following
Figure 3.10 The autocorrelation function.

Figure 3.11 The velocity profile for simulation.
inequalities should be satisfied:

\[ BT > 5 \quad \text{and} \quad T > 10 \cdot |t| \]

So for the computer model results to have any statistical significance, it is necessary to average the autocorrelation function over several estimates.

The computing time required to generate two flow signals and to compute the autocorrelation and crosscorrelation functions is approximately three minutes using the program developed. The total time required increases as a linear function of the number of estimates required. Thus assuming the desired variance is 1\% of the peak of the autocorrelation function (this implies a standard deviation of 10\%), one hundred estimates are needed or approximately three hundred minutes of computer time!

A second and more efficient simulation program (listed in Appendix 2) has been developed. SIMULA V2.0 as it is called, allows the user to specify the number of averages over which the correlation functions are to be calculated. This program also facilitates relatively easy changes of pipe length, pipe width, transducer beamwidth and the number of data representing each flow signal.

Results using the stepped velocity profile shown in Figure 3.11 are in the form of a comparative study.
Figure 3.12 The comparative curves.
The relative heights of the peaks corresponding to the velocities \( V_1 \) and \( V_2 \) are plotted as the width of the subchannel moving at that velocity is varied. The curves are plotted for 10 and 50 averages of the cross-correlation function. A linear relation between the peak height of the crosscorrelation function and the subchannel width is evident if one examines the results given in Figure 3.12. A similar study was carried out plotting the ratio of signal peak to noise peak of the crosscorrelation function as a function of subchannel width. The signal peak is found by recording the height of the crosscorrelation function at the delay corresponding to the velocity of interest, while the noise peak is found by examining the maxima of the crosscorrelation function at delays other than those corresponding to the velocities present.

Further simulation results using different velocity profiles are given in Appendix 2.

3.1.5 Conclusion

The simulated flow signals appear to have similar characteristics as those studied by Leitner (Reference 3.1). The spectral density of the flow signal should really be calculated for several different samples of the flow signal then averaged. This would yield a smoother estimate of the spectral density function. The calculated
autocorrelation and crosscorrelation functions found using the stepped velocity profile (as shown in Figure 3.6) are poor estimates. As the averaging time is increased, the estimate of the autocorrelation and crosscorrelation functions improve as is illustrated by Figure 3.7, where 100 samples were used.

Further work comparing the peak heights of the crosscorrelation function and the ratio of signal peak to noise peak for different subchannel widths reveals the expected linear relations between these parameters. The effect of the averaging on the estimated correlation functions is evident if one compares Figures 3.6 and 3.7.

The results show that accurate simulation of the flow signals and the computation of their autocorrelation and crosscorrelation functions requires lengthy, and in this case impractical, amounts of computer time. The simulation exercise has proved very useful in providing a better understanding of the mechanism of flow signal generation. The alternative to the computer simulation is to analyse real flow signals using a correlator to perform online calculation of the autocorrelation and crosscorrelation functions. This process will require less calculation time because the flow signal generation and the correlation calculations are carried out by separate, dedicated devices.
Figure 3.13 A schematic of the flow circuit.

Figure 3.14a The flow circuit and pump.
3.2 THE EXPERIMENTAL FLOW RIG

3.2.1 Introduction

The need for an experimental flow rig arises due to the excessive and impracticable requirements of computer time needed to perform the simulation of the flow signals.

3.2.2 Objectives

The motivation behind this experiment is the desire to measure the velocity distribution in a flow system while computing the autocorrelation and crosscorrelation functions of the upstream and downstream flow signals for the same flow conditions.

3.2.3 Description of the Experiment.

The flow rig simply consists of a pump-driven water circuit. This circuit has a section of pipe through which light beams may be passed so that two optically derived flow signals may be obtained and some optical measurement of the velocity distribution made. Some control of the flow velocity is provided by way of two valves and a bypass. Photographs and a schematic are given in Figures 3.13 and 3.14.
Figure 3.14b A view showing the photocell electronics, the laser, the beamsplitter and the mirror arrangement.

Figure 3.14c A close up showing the parallel laser beams passing through the flow.
The optical system comprises of a helium-neon laser source, a beam splitter and two photocells to produce the flow signals. Particles and bubbles in the water modulate the beams received by the photocells which produce an electrical current proportional to the intensity of the incident light. The square box arrangement around the experimental section is filled with water and this reduces the refraction of the light beams at the air-pipe interfaces. The calculation of the autocorrelation and crosscorrelation functions of the flow signals is then carried out using a Honeywell correlation and probability analyser. Copies of the correlation functions are obtained from the correlator memory via a chart recorder. The correlator has two modes of operation:

(i) A clipped mode, where the flow signals are quantised to one of two levels and

(ii) A full mode where no quantisation is applied.

It has been shown (in Reference 3.3) that the true correlation functions (computed using the correlator in mode (ii)) and those obtained when using the correlator in clipped mode are related by a sine transform. This particular correlator produced better correlation functions when used in the clipped mode.

Finding the velocity distribution requires the measurement of the velocity profile in the flow. The technique used
Figure 3.15 Velocity profile measurement using crossed-beam correlation.

Figure 3.16 The velocity profile and corresponding distribution.
was that of crossed-beam correlation explained in Reference 3.4. This is a completely independent process to that described earlier, although the basic technique remains the same. To achieve crossed-beam correlation, the two laser beams are now set at 90° to one another some small distance apart. This is shown more clearly in Figure 3.15. The correlator is then used to find the time taken for the particles or bubbles travelling in the elemental volume of fluid common to both beams, to move the distance between the laser beams. The velocity profile is obtained by fixing the position of one beam while traversing the other across the pipe diameter.

3.2.4 Results

Sample autocorrelation and crosscorrelation functions are given in Figure 3.17. The velocity profile measured for the same flow conditions is shown in Figure 3.16. The asymmetry of the crosscorrelation function caused by this velocity profile is apparent if one examines Figure 3.16.

3.2.5 Conclusion

The flow rig functions well as may be seen from the results presented. Although this is a more elaborate
Figure 3.17 The measured autocorrelation and crosscorrelation functions.
method of finding the velocity profile and the auto-
correlation and crosscorrelation functions, the results
show a definite improvement over the computer simulation
results.

3.3 THE ANALYSIS PROGRAM

3.3.1 Introduction

This program has been developed so that deconvolution
based on the process described by equation 2.5 may be
thoroughly investigated. The package has been developed
using a Univac 1106 mainframe computer and the pro-
gramming language used is Ascii Fortran (level 8R1).
The software is designed to be used interactively so that
an on line analysis of any waveform may be carried out.

3.3.2 Specifications

The user may operate on any waveform by entering a
sampled version of the waveform into one of four work
areas. All the operations are then performed on the
respective work area. The program enables the user to:

1. Store and recall any waveforms from permanent or
temporary memory.
2. Set up labels for the various waveforms which have
been temporarily stored.
3. List these labels.

4. Set up labels for each work area.

5. List these work area labels.

6. Compute the Fourier transform of a complex waveform.

7. Compute the inverse Fourier transform of a complex waveform.

8. Apply one of five different data windows to any waveform.

9. Perform polar to rectangular and rectangular to polar conversions on a complex waveform.

10. Fold a waveform about the central point.

11. Convolve two real waveforms.

12. Set parts of any work area equal to a constant value.


14. Apply sine and arcsine transforms to any waveform.

15. Shift a waveform cyclically.

16. Equate various elements of any work area.

17. Examine and change individual data values of each work area.

18. Perform step by step deconvolution of two complex waveforms using the division of the Fourier transforms.

19. List all data values in any work area on the visual display unit (vdu) screen.

20. Print data values on the line printer.

21. Plot the data values of any work area on the vdu screen using a character graph.

22. Plot the data values of any work area using the Calcomp plotter.
23. Add white noise of variable standard deviation and amplitude to any waveform.
24. Perform most arithmetic operations on any work area.
25. Perform arithmetic operations with the work areas as variables.
26. Compute the sum of all the elements in any work area.
27. Correlate two waveforms.
28. List all the commands available.

3.3.3 Description

The basic method of operation is to enter correlation functions obtained either from a computer simulation or experimentally via the terminal keyboard. Operations from those listed above may then be applied to the waveforms to obtain some idea of the usefulness of the proposed analysis techniques. The main program, the various subroutines and some description are given in Appendix 3. Many of the techniques used have been described in Chapter 2.

3.3.4 Results

Almost all the subsections of this program are used to produce many of the results given in Chapter 4. Examples of typical Fourier transforms, inverse Fourier transforms, the data windows, the convolution and deconvolution of
two functions, data listing and the noise function are given in Appendix 3.

3.3.5 Conclusion

This package is a very powerful tool for the application of online waveform analysis and the testing of signal processing techniques. It should not be too great a task to modify or add to the package any particular function seen to be useful.

3.4 SUMMARY

This chapter has dealt with the proposed computer simulation of the flow mechanism and the flow signal generation with a description of the model used. The problems which the computer simulation raises are discussed along with the results. Due to the simulation failure, an experimental flow rig has been developed and sample measurements of the various parameters given. The analysis of the measured parameters has been made possible by the development of a waveform analysis package described briefly here. The detailed working is given in Appendix 3 and the fundamental theory in Chapter 2. Analysis of the parameters measured and some discussion of further investigation into the deconvolution method forms the following chapter.
REFERENCES - CHAPTER 3


BIBLIOGRAPHY

Figure 4.1 The deconvolution of an arbitrary ACF and CCF.
The results of the deconvolution of simulated autocorrelation and crosscorrelation functions and those measured from the experimental flow rig are examined in this Chapter. Some discussion as to the success or failure of the method in each case is presented at the end of the Chapter.

### 4.1 Deconvolution of Arbitrary Autocorrelation and Crosscorrelation Functions

Prior to the development of the simulation and the analysis programs (SIMULA and WAVEPACK), the deconvolution of two arbitrary autocorrelation and crosscorrelation functions was carried out. These two functions as well as the resulting transit time distribution are shown in Figure 4.1.

Given a system with its input signal $x(t)$, output signal $y(t)$ and impulse response $g(t)$, the system output is related to the input by the convolution of the input and the system impulse response. It can be shown (refer to Appendix 4) that if the input signal $x(t)$ is replaced by the autocorrelation of the input, namely $\mathcal{R}_{xx}$, and $y(t)$ is replaced by the crosscorrelation of the input and
Figure 4.2a The autocorrelation function.

Figure 4.2b The crosscorrelation function.

Figure 4.2c The transit time distribution.

Figure 4.2d The velocity distribution.

Figure 4.2e The velocity profile.
output signals, namely $\xi_{xy}$, then these correlation functions may be used instead of $x(t)$ and $y(t)$ in the convolution relation which still holds.

Considering now the system with its input the autocorrelation function (ACF), output the crosscorrelation function (CCF) and the transit time distribution (TTD) as its impulse response, the effect of the TTD is to smear the CCF. The deconvolution allows one to evaluate this smearing function, the transit time distribution. The results shown in Figure 4.1, although somewhat qualitative, look promising.

4.2 DECONVOLUTION OF AUTOCORRELATION AND CROSSCORRELATION FUNCTIONS OBTAINED FROM THE COMPUTER SIMULATION OF THE FLOW

The autocorrelation and crosscorrelation functions shown in Figure 4.2 were calculated from flow signals simulated using the stepped velocity profile shown in Figure 4.2e. The correlation functions have been averaged over 100 samples. The CCF is expected to exhibit two peaks corresponding to the two velocities present in the velocity profile. The noise due to the finite averaging time is clearly evident if one examines the CCF. The transit time distribution obtained by the deconvolution of the ACF and CCF is shown as well as the velocity distribution obtained by applying
Figure 4.3a The autocorrelation function.

Figure 4.3b The crosscorrelation function.

Figure 4.3c The transit time distribution.

Figure 4.3d The velocity distribution.

Figure 4.3e The measured transit time distribution (obtained from velocity profile measurements, refer to section 3.2).
equation 1.5. Insufficient averaging time gives rise to the irregular shape of the deconvolved transit time distribution.

Results using flat and stepped velocity profiles and the subsequent deconvolutions are given in Appendix 5.

4.3 DECONVOLUTION OF THE AUTOCORRELATION AND CROSSCORRELATION FUNCTIONS MEASURED FROM THE EXPERIMENTAL FLOW RIG

The ACF and CCF measured from the flow rig are given in Figures 4.3a and 4.3b. The calculated transit time and velocity distributions are given in Figures 4.3c and 4.3d. There is very little similarity between the calculated and expected transit time distributions.

Examining the Fourier transforms of the ACF and CCF at two frequencies, the numerical values of the Fourier transform of the ACF are very much smaller (approximately 2000 times) than the corresponding values of the Fourier transform of the CCF. Consequently, when dividing these two transforms, the resulting function has two sharp peaks at these frequencies. When inverse transforming these peaks give rise to the periodic component displayed by the TTD. Various methods of filtering were tried to remove these transmission zeros, but with no success.
Figure 4.4 Schematic of the simplified flow system showing how the two delays $\tau_1$ and $\tau_2$ are modelled.

Figure 4.5 The measured autocorrelation and crosscorrelation functions.

Figure 4.6 The deconvolution result.
In order to gain further understanding of the deconvolution process, it was decided to experiment with a simplified model of the flow system. This model, shown schematically in Figure 4.4, has a system impulse response or transit time distribution consisting of two delayed unit impulses. These conditions are identical to those produced by pipe or channel flow in the presence of a stepped velocity profile. The aim of this experiment is to deconvolve the autocorrelation and crosscorrelation functions calculated from these "flow signals" in an effort to arrive at the known transit time distribution.

A random noise generator was used as the upstream flow signal and the downstream flow signal was produced using a tapped analogue delay line as the flow process. The downstream flow signal was actually produced by using the upstream flow signal as an input to the delay line and summing the output from two different taps along the delay line.

The ACF and CCF (shown in Figure 4.5) were measured using the correlator in full mode so that the application of the sine transform was not necessary. The deconvolved TTD shown in Figure 4.6 also exhibits the periodicity resulting from noise present in the crosscorrelation function.
4.5 **AN ANALYSIS OF THE NOISE EFFECTS ON THE DECONVOLUTION RESULT:**

Given the linear system with input $x(t)$, output $y(t)$ and impulse response $g(t)$, then we may write, as before:

$$ y(t) = \int_{-\infty}^{\infty} g(t-\tau) x(\tau) \, d\tau $$

Using the Fourier transform domain to simplify the analysis, we can show that:

$$ G(w) = \frac{X(w)}{Y(w)} $$

Now, when dealing with deterministic and noise-free functions the above relations describe the deconvolution process very well. Inevitably the waveforms of interest have some added noise and these expressions must be amended to reflect this. Basically, the expression relating the input and output in the above equation must be modified:

$$ y(t) = \int_{-\infty}^{\infty} g(t - \tau) x(\tau) \, d\tau + n(t) \quad \ldots \quad 4.1 $$

so that:

$$ G(w) = \frac{Y(w)}{X(w)} - \frac{N(w)}{X(w)} \quad \ldots \quad 4.2 $$

It can be seen from the above equations that if $X(w_1)$ is very small, the effect of the noise term on the deconvolution result may be very large if $Y(w_1)$ is also
small. In fact, if one considers the noise level to be constant then when the value of $Y(w_i)$ is small, the effect of the noise on the deconvolution result becomes more prominent than the desired signal.

This deconvolution problem has been tackled in other areas such as astronomy where the effects of the telescope transfer function are removed from the observed signals. A fundamental difference between the deconvolution of the autocorrelation and crosscorrelation functions and the deconvolution of waveforms observed in astronomy is that the telescope transfer function is well defined, usually a smooth function which can be represented analytically and in most cases has no transmission zeros. If one equates the function $X(w)$ as discussed previously with this telescope transfer function, we see that equation 4.2 converges for all $X(w_i)$ non zero. But if one allows $X(w_i)$ to tend towards zero for some $w_i$, then equation 4.2 may diverge. The results given earlier in this section display these divergent properties.

Deconvolution of two waveforms with an increasing amount of noise added to the output $y(t)$ was carried out. The results given in Appendix 6 show that the deconvolution result is very sensitive to small amounts of added noise.
4.6 SUMMARY

The deconvolution program has been used to analyse simulated and experimentally measured correlation functions. The best deconvolution results have been those calculated from the simulated correlation functions. Successive deconvolution of two functions with increasing amounts of added noise have shown that this deconvolution technique yields very poor results when applied to real waveforms in the presence of noise. The application of simple mathematics has substantiated the sensitivity of the deconvolution result to noise inherent in the waveforms of interest.

It therefore appears that straightforward deconvolution has limited application and when deconvolving noisy waveforms, some process which uses an averaging technique to reduce the noise effects is desirable. One such technique known as Bayesian deconvolution is discussed in the following chapter.
CHAPTER 5

BAYESIAN DECONVOLUTION

5.1 INTRODUCTION

Investigation into other deconvolution techniques arising due to the failure of the method discussed in Chapters Two and Four has yielded the idea of Bayesian deconvolution. A brief treatment is given here with some preliminary results which are shown to be very promising.

Richardson, in Reference 5.1, has developed the idea of applying probability theory to the restoration of noisy, degraded spectra as found in spectroscopy. Using Bayes' theorem (Reference 5.2), Richardson has formulated an expression for the iterative estimation of an original image given the degraded image and the point spread function of the measuring system. Using matrix notation the iterative estimation may be described by:

\[ T_i(n+1) = T_i(n) \frac{\sum_k R_{k,i} M_k}{\sum_j \sum_k R_{k,j} T_j(n)} \]  

where \( R \) is the response matrix, \( M \) the measured spectrum, \( T \) the true spectrum and \( n \) the iteration index.

Kennet et al. in Reference 5.3 have applied Bayesian deconvolution to photoneutron, x-ray and optical...
spectroscopy amongst others. It appears that this method of deconvolution may readily be applied to the calculation of the transit time distribution by deconvolution of the autocorrelation and crosscorrelation functions as found in crosscorrelation flowmetering.

The parameters of equation 5.1 need to be changed to suit this slightly different application. Assuming $y$ to be the system output vector, $g$ the impulse response vector and $x$ the input vector, we may rewrite equation 5.1 thus:

$$g_i(n+1) = g_i(n) + \sum_{k} \sum_{j} X_{k,i} Y_{k,j} G_j(n)$$

Assuming $s$ and $p$ to be working vectors, the implementation of this expression as used by Kennet et al. may be described by the following five steps:

1. $g = y$
2. $p = x * g$
3. $s = Y / P$
4. $p = x * s$
5. $g = g * p$

Where * and $\oplus$ denote the convolution and correlation operations respectively. The above algorithm is easier to implement using the speed and convenience of the fast Fourier transform. Using upper case letters for the respective Fourier transforms and representing the
Figure 5.1 The deconvolution of a smeared step function and a Gaussian function. Figure a shows the original step function, figure g the smeared step function and figures f to b the estimates after 2, 8, 32, 128 and 512 iterations respectively.
transformation operation via the Fourier transforms by an arrow, this algorithm may be rewritten:

i. \( g = x \)

ii. \( x \rightarrow X \)

iii. \( g \rightarrow G \)

iv. \( P = X.G \)

v. \( P \rightarrow p \)

vi. \( S = y/p \)

vii. \( s \rightarrow S \)

viii. \( P = X.S^* \)

ix. \( P \rightarrow p \)

x. \( g = g.p \)

The accuracy of the estimate of \( g \) improves as the number of iterations increases.

5.2 IMPLEMENTATION

The five step algorithm given above was incorporated as another, single instruction of the WAVEPACK program. The actual code required is only about twenty lines. The more efficient, ten step algorithm would be suited to a microprocessor implementation of this deconvolution technique.
Figure 5.2 The deconvolution of the measured ACF and CCF from the experimental flow rig. The measured ACF and CCF are shown in a and b respectively, the transit time distribution obtained from the velocity profile measurement is given in figure c above. The convergence of the deconvolution is shown in d to i where the number of iterations is 0, 10, 50, 100, 150 and 200 respectively.
5.3 RESULTS

In order to validate the code written, similar tests to those carried out in Reference 5.2 were undertaken. A Gaussian function was convolved with the stepped function (given in Figure 5.1a) to produce a smeared step function (shown in Figure 5.1g). The deconvolution of this smeared step and the Gaussian functions was then carried out with an increasing number of iterations. The results are given in Figure 5.1 and they resemble very closely those obtained by Kennet et al.

The following step is obviously to apply this deconvolution technique to the autocorrelation and crosscorrelation functions measured experimentally and to compare the result with the measured transit time distribution. This has been done and the results are shown in Figure 5.2. The deconvolution produces a good approximation of the transit time distribution which improves as the number of iterations is increased. The variance between the measured and calculated transit time distributions falls fairly rapidly as the number of iterations is increased. This is shown in Figure 5.3.

5.4 CONCLUSION

The Bayesian deconvolution algorithm has been successfully applied to the deconvolution of a smeared step
Figure 5.3 The variance between the measured and the calculated transit time distribution as a function of the number of iterations.
function and subsequently to the deconvolution of the autocorrelation and crosscorrelation functions as measured from the experimental flow rig described in Section 3.2. It remains now to propose possible implementations of this algorithm and to put forward further research ideas.
REFERENCES - CHAPTER 5


DISCUSSION, RECOMMENDATIONS AND CONCLUSIONS

6.1 SUMMARY OF RESULTS

The aim of this project has been to investigate a technique to improve the range of operation and the accuracy of the crosscorrelation flowmeter. Efforts have been directed towards the calculation of the velocity distribution using signal processing techniques to analyse the autocorrelation and crosscorrelation functions. Errors introduced by the variation of the velocity profile in the flow being measured may be eliminated if the velocity distribution can be calculated from the flowmeter outputs.

Initially it was hoped to simulate the flow mechanism and the flow signal generation so that different correlation functions in the presence of various velocity profiles may be obtained. The simulation of the flow using a computer model appeared very convenient because the ease with which the different parameters may be varied. Two programs were envisaged, a simulation program in which a particular flow would be simulated followed by an analysis program to operate on the simulated flow signals in order to arrive back at the velocity distribution of the flow system.
For accurate simulation of the flow, the computer program was found to require excessive and impracticable amounts of computer time despite the various attempts at optimising the code. This led to the construction of a test rig so that the correlation functions and the prevailing velocity profile could be measured. These measured functions were then analysed as before. The flow rig proved to be very useful and the measurement of the velocity profile by crossed-beam correlation yielded a good result.

The analysis package was developed so that the deconvolution of the autocorrelation and crosscorrelation functions could be carried out. Many digital signal processing functions have been incorporated into this package. The deconvolution of simulated and experimentally measured correlation functions was carried out using the method of the division of the Fourier transforms. This technique has proved to be very sensitive to noise and the transit time distributions thus calculated bear no resemblance to the expected transit time distributions.

Experiments using a tapped delay line were carried out and the results have shown the noise present in the correlation functions to give rise to transmission zeros. A theoretical analysis has substantiated this explanation of these divergent results.
The failure of this direct method of deconvolution has led to a brief investigation into Bayesian deconvolution. The deconvolution of a stepped function and a Gaussian was carried out successfully using this algorithm. The deconvolution of the correlation functions measured experimentally agree well with the transit time distribution measured from the flow rig. The variance of the estimate is seen to reduce to 2% of the transit time distribution when calculated over 200 iterations. Implementation of this Bayesian deconvolution was achieved by adding it as a function carried out by the waveform analysis package.

The results given in this work detail only the investigation of a suitable algorithm which can be used to deconvolve the autocorrelation and crosscorrelation functions. No work has yet been carried out to implement this deconvolution as part of a crosscorrelation flowmeter.

6.2 FURTHER WORK

When using the crosscorrelation flowmeter, inaccuracies are introduced due to the presence of a curved velocity profile in the flow being measured. The effect of the velocity profile is to make the crosscorrelation function asymmetrical. If the requirement is to measure the mean transit time of the flow, then the
simplest strategy is to compute the transit time distribution using for example Bayesian deconvolution and then to calculate from this the average transit time. The average velocity may then be calculated from the average transit time. One can envisage a modern generation, sixteen bit microprocessor sampling the two flow signals, calculating their autocorrelation and crosscorrelation functions, then applying Bayesian deconvolution to calculate the transit time distribution and from this computing the mean transit time and hence the average velocity. The calculation of the correlation functions and the implementation of the Bayesian deconvolution algorithm would require the implementation of a fast Fourier transform using a microprocessor. This has already been shown to be realistic even though a dedicated unit may be required.

The correlation functions calculated using a polarity correlator are related to the direct correlation function by the Van Vleck equation, that is:

\[ R_D = \sin \left( \frac{\pi}{2} R_p \right) \]

where \( R_D \) is the normalised, direct correlation function and \( R_p \) the polarity correlation function. Jordan (Reference 6.1) suggests a method for simplifying the calculation of the discrete Fourier transform of a correlation function found using a polarity correlator.
Figure 6.1 Movement of particles in a channel of width $d$ through the beam quantized into $W$ smaller beams.
He uses the above equation to reduce the calculation of the discrete Fourier transform to trigonometric and addition functions in much the same way as the fast Fourier transform is developed. This idea may be applied to the calculation of the transit time distribution via the Bayesian deconvolution algorithm.

Further research may be aimed at the actual realisation of this crosscorrelation flowmeter which incorporates all the above innovations.

Rametti (in Reference 6.2) has proposed the notion of identifying the flow system using only a single measuring station. If one considers this single transducer to produce an interrogating beam which is wider than the largest particle in the flow, then the process of the flow signal generation may be modelled as the measurement of the flow particles through a quantized beam. This is shown schematically in Figure 6.1 where there are \( W \) smaller beams. This proposal assumes a rectangular beam cross-section although this is not necessary. The approach adopted here is similar to the operation of the parallel-slit image velocity sensor as discussed in Reference 6.3.

Consider now a fluid element representing a fraction \( b_i \) of the pipe total cross-sectional area, which proves at a velocity \( V_i \) (as shown in Figure 6.2). This will
Figure 6.2 Cross-sectional view through pipe showing the quantized velocity profile.

Figure 6.3 Identification of the system by modelling the system parameters.
cause a given disturbance in the $i^{th}$ transducer beam after a delay of $T_i$ sampling intervals. The transport mechanism may be expressed mathematically as:

$$y(k) = b_0 u(k-T_0) + b_1 u(k-T_1) + ...$$

$$... + b_w u(k-T_w)$$

$$= B(Z^{-1}) u(k)$$

Well accepted methods for computing $B(Z^{-1})$ are outlined in Reference 6.4 although in this case the flow system is of simpler form than those in the reference. In general terms the system to be identified is modelled and an error function formed from the calculated and the measured system outputs is minimised by suitable variation of the model parameters (see Figure 6.3).

The system identification program developed by Mr. L.B. Rametti was run successfully with flow signals simulated with a flat velocity profile as the system inputs and outputs. This program optimises the model parameters using a least squares error minimisation algorithm. Unfortunately, major changes are required to the existing software to cater for the unusually long delays which characterise the flow system.

This system identification approach should be investigated
to determine the feasibility of applying this least squares error minimisation to an on-line flow system identification scheme.

6.3 CONCLUSIONS

It has been the aim of this work to investigate a method for calculating, from the outputs of a cross-correlation flowmeter, the velocity distribution in the flow being measured. The need to measure the velocity distribution arises from the efforts to increase the accuracy and range of the crosscorrelation flowmeter. Measurement of the flow velocity in the presence of complex velocity profiles gives rise to error in the estimation of the flow transit time. The approach adopted here has been to consider the flow and transducer as a system with the flow signals as inputs and outputs and to then identify this system to obtain the transit time distribution.

Several deconvolution methods have been outlined in Chapter 2 and one of these examined in detail. Due to the extreme sensitivity of this first approach to noise, a second deconvolution method has been examined. The results of this later investigation appear to overshadow the work done in applying the simpler deconvolution method. Experiments have shown that the presence of
additive noise led to the failure of this otherwise satisfactory deconvolution algorithm. Valuable experience in the application and theory of digital signal processing has been gained and this work is felt to have been worthwhile. This alternative method of Bayesian deconvolution has been briefly examined with good results which show this technique to be directly applicable to the calculation of the transit time distribution.

Some of the groundwork for the application of digital signal processing techniques to crosscorrelation flowmetering has been carried out. The techniques discussed here have definite application to the improvement of range and accuracy of the existing flowmeter. The implementation of these methods to the measurement of the mean flow velocity in the presence of complex velocity profiles has still to be carried out.

It is hoped that the work presented here will provide a contribution to the field of crosscorrelation flowmetering and electrical engineering in general.
REFERENCES - CHAPTER 6


A SIGNAL PROCESSING MICROPROCESSOR

It is intended to develop a microprocessor based, crosscorrelation flowmeter at some later stage incorporating into it a correction for the velocity profile in the flow being measured. This microprocessor system was developed as a building block for the flowmeter.

Basically it consists of five parts, a standard Intel SDK85, extended RAM and EPROM facilities, two analogue to digital converters (ADCs), an interrupt clock and a display unit. The reason for the two ADCs was to make it possible to use the microprocessor system to perform the correlation of the flow signals.

The SDK85 is well known and needs no explanation.

The two analogue to digital converters are 8 bit, bus compatible ADC0804s. These have a maximum conversion time of 100 µsecs which is fairly slow by normal standards but adequate for this application. The ADCs are memory mapped and to start conversion the desired ADC is written to. At the end of conversion the corresponding bit of PORT 1 of the SDK85 is set low. The ADC is reset when the 8 bit value is read from
it. ADC 1 is addressed at 9800 to 9FFF and ADC2 at A000 to A7FF.

The interrupt clock operation is very simple; a 12 bit counter which has one of its outputs switch selectable. The selected counter output is then used to clock a '1' into a D type flip flop whose output is connected to the RST6.5 interrupt line of the SDK85. It is necessary to use the flip flop because the RST6.5 interrupt is level sensitive. It is necessary to clear the flip flop having completed the RST6.5 interrupt service routine, which is done by sending a '0' to bit 5 of PORT 1.

The display section allows 1K of memory to be displayed on an oscilloscope screen independently of the central processing unit. The display RAM is memory mapped but may only be written to or read from when it has been selected by the CPU. To select the display RAM bit 4 of PORT 1 is set to a '1'. This has the effect of blanking the display, using the oscilloscopes Z-mod input, and switching the address and data lines of the CPU to the display RAM. When deselected, i.e. bit 4 of PORT 1 is set to '0', each location of the display RAM is addressed sequentially using a counter. The data contained in these memory locations is then converted to an analogue voltage using a digital to analogue converter (DAC). Originally it was intended to use
the oscilloscope in the X-Y mode, thus a voltage ramp is provided by another DAC which simply uses the RAM address lines as its input. It has proved to be less problematic to use the ramp as an external trigger so the DAC which generates this ramp is not really needed, a single pulse would be sufficient to trigger the timebase.

This display RAM is addressed from A800 to AFFF (with a 1K foldback). The oscilloscope trace is blanked while the RAM address is changing. This is done using a monostable which is triggered by the lowest order address line which modulates the Z-mod input to the oscilloscope (shown in Figure A1.4).

Examples of a sampled waveform, its Fourier transform and inverse Fourier transform is shown in Figure A1.5. These are taken from the work done by Weeks in Reference 2.12.
Figure A1.2

<table>
<thead>
<tr>
<th>CONTENTS OF PORT A FROM CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>PCE</td>
</tr>
</tbody>
</table>

X = don't care.
When conversion is complete, PCE sets specific bit to '1'. When ADC is read by CPU, this bit is reset.
Figure A1.5a Sampled portion of a sine-wave.

Figure A1.5b FFT magnitude function showing leakage due to the sharp discontinuity of the data.
This program is designed to simulate multiphase fluid flow in a pipe and generate flow signals. The program outputs are the autocorrelation and crosscorrelation functions of the upstream and downstream signals.

The code is well documented and fairly self-explanatory. Various parameters which are of interest are: NPTS which specifies the number of data points desired in the calculation of the autocorrelation and crosscorrelation functions; LENGTH and WIDTH which determine the desired pipe length and width; AVLNGT specifies the transducer beamwidth and ACFILE and CCFILE set up the desired output file numbers.

The program inputs are (i) the number of times the autocorrelation and crosscorrelation functions are to be averaged, and (ii) the desired velocity profile consisting of integers between zero and nine. A typical runstream and the subsequent program printout is shown overleaf.

In this example SIMULA writes the autocorrelation and crosscorrelation functions to the data files ACF100-V2 and CCF100-V2 respectively. The program prints out
the parameters used and the normalising factors, as is seen here. Subsequent processing, plotting or printing is done using the WAVEPACK program which is detailed in the following Appendix. Further examples are given in Chapter 4 and Appendix 5, where the simulated correlation functions are analysed.

The autocorrelation and crosscorrelation functions resulting from this runstream are given in Figure A2.2.
Autocorrelation and crosscorrelation functions found from the simulated flow signals computed using more complex velocity profiles are shown in Figures A2.3, A2.4 and A2.5. All these correlation functions have been computed by averaging over 100 flow signal samples.

Figure A2.5a The stepped velocity profile.

Figure A2.5b The autocorrelation function.

Figure A2.5c The crosscorrelation function.
TITLE: SIMULA

DESCRIPTION:

This program is designed to simulate multiphase fluid flow in a pipe and to generate flow signals. The program outputs are the auto and crosscorrelation functions of the upstream and downstream signals.

CALLS: RAN

CODE:

Definition of variables to be used.
===================================================================

IMPLICIT COMPLEX (A-Z)

PARAMETER NPTS = 64
PARAMETER LENGTH = 50, WIDTH = 20
PARAMETER AVLNGT = 10
PARAMETER ACFILE = 11, OFILE = 12

REAL UPSIG(0:NPTS-1), DUNSIG(0:NPTS-1), ACF(0:NPTS-1)
REAL CDF(0:NPTS-1)
REAL DATA(1:WIDTH, 1:LENGTH), RANDARY(1:WIDTH, IMPAND(1)
REAL NOSUMS, RNDAEC, NOSUMS

INTEGER TEMP, INDEX, NOAVE, WHERE, SHIFT
INTEGER I, J, K, L

REAL U'SIG(0:NPTS-1), DWNSIG(0:NPTS-1), ACF(0:NPTS-1)
REAL CC(0:NPTS-1)
REAL ATALL:WIDTH, L:LENGTH, RNDAEC(1:WIDTH)

C Printing header.
C
PRINT 100
C
C Reading the number of records the acf and the ccf are to be averaged over and the desired velocity profile.
C
C
READ *NOAVE
READ *(VPFPROFI, I=1,WIDTH)
C
C Printing out the selected parameters.
C
PRINT 20, LENGTH, WIDTH, NPTS, AVLNGT

FORMAT(//,3X,'PIPE LENGTH = ',13,3X,'PIPE WIDTH = ',13,3X,
1//,3X, 'NUMBER OF SAMPLES = '),4X, 'AVERAGEING WIDTH = ',
1//,3X,
PRINT 30, WHERE, VPRFPROFI(1), I=1,WIDTH)

FORMAT(//,3X,'NUMBER OF RECORDS AVERAGED = ',14,3X,
4 'VELOCITY PROFILE USED IS : ',3X,20(2X,12)
C
C
C Initializing variables.
C
IMAEC(1) = 314159
RNDAEC(1) = 23509
NOSUMS = FLAT(AVLNGT * WIDTH)
C
C Initializing fluid elements in the pipe.
C
C
DO 100 I = 1,LENGTH
C
RNDAEC(I) = RANDARY(I) + 100000000.
CALL RANDARY(5)
C
DO 1010 J = 1,WIDTH
DATA(1,J) = RANDARY(J)
1010 CONTINUE

C

1000 CONTINUE
Computing the flow signals, ACF and CCF for each record.

DO 2000 I = 1,NOAVE

Repeat the following section for each sample of the flow signals.

DO 3000 INDEX = 1,NPTS-1

Implementing velocity profile effects for each sample.

DO 3100 J = 1,WIDTH

TEMP = VPROF(J)

IF ( TEMP .NE. 0 ) THEN

DO 3110 K = 1,TEMP

DO 3120 L = LENGTH,2,-1

DATA(J,L) = DATA(J,L-1)

CONTINUE

3120

TMFDND(1) = TMFDND(1) * 1000000.

CALL RANDNI TMFDND, 1, 0., 0.5

DATA(J,1) = TMFDND(1)

3110 CONTINUE

ENDIF

3100 CONTINUE

Calculating the flow signal samples.

DO 3200 J = 1,AVLNGT

WHERE = LENGTH - AVLNGT + J

DO 3210 K = 1,WIDTH

UPSIG(INDEX) = UPSIG(INDEX) + DATA(K,J)

DUNSIG(INDEX) = DUNSIG(INDEX) + DATA(K,WHERE)

3210 CONTINUE

3200 CONTINUE

Normalizing the flow signals.

UPSIG(INDEX) = UPSIG(INDEX) / NOBUNS

DUNSIG(INDEX) = DUNSIG(INDEX) / NOBUNS

3000 CONTINUE

Calculating the ACF and the CCF from the flow signals.

DO 4000 J = 0,NPTS-1

DO 4010 K = 0,NPTS-1

SHIFT = MOD(K+J,NPTS)

ACF(J) = ACF(J) + UPSIG(K) * UPSIG(SHIFT)

CCF(J) = CCF(J) + UPSIG(K) * DUNSIG(SHIFT)

4010 CONTINUE

4000 CONTINUE

DO 10 J = 0,NPTS-1

UPSIG(J) = 0.

DUNSIG(J) = 0.

10 CONTINUE

2000 CONTINUE
C Normalizing the averaged ACF and CCF.

C******************************************************************************

MAXACF = ABS(ACF(0))
MAXCCF = ABS(CCF(0))

DO 5000 I = 1,NPTS-1
   IF ( ABS(ACF(I)) .GT. MAXACF ) MAXACF = ABS(ACF(I))
   IF ( ABS(CCF(I)) .GT. MAXCCF ) MAXCCF = ABS(CCF(I))
5000 CONTINUE

DO 6000 I = 0,NPTS-1
   ACF(I) = ACF(I) / MAXACF
   CCF(I) = CCF(I) / MAXCCF
6000 CONTINUE

C Printing the values of ACFMAX and CCFMAX.
C******************************************************************************

PRINT 300, MAXACF, MAXCCF

C Writing the ACF and CCF data to a permanent file.
C******************************************************************************

DO 7000 I = 0,NPTS-1,8
   WRITE (ACFILE, 200) ( ACF(I+J), J = 0,7 )
   WRITE (CCFILE, 200) ( CCF(I+J), J = 0,7 )
7000 CONTINUE

C STOP

C Format statements.
C******************************************************************************

100 FORMAT ( 'SIMULA V2.0 17/02/81 ')
200 FORMAT ( 8(3X,F12.6) )
300 FORMAT (/3X,'ACF NORMALIZING FACTOR = ',F12.4,3X,
     'CCF NORMALIZING FACTOR = ',F12.4//)

END
END OF FILE
A3.1 INTRODUCTION

This program has been designed to function as an interactive waveform analysis package. It is intended to be used as a tool to simplify the implementation of signal processing techniques to various practical problems. Motivation for the development of the program has been to aid the work done to deconvolve two waveforms using Fourier transform techniques. The software has been written in Ascii Fortran level 8R1 and implemented on a Univac 1106 multiprocessor, time-sharing system.

A3.2 SPECIFICATIONS

The program functions almost as a powerful pocket calculator with fairly complex operations on large quantities of data. Four work areas WA1, WA2, WA3 and WA4 are available to the user as scratchpad type memory. It is on these work areas that all the commands are carried out. To execute a command the user types in from the terminal, the command name followed by the various parameters required. One may consider each work area to be an array which contains a sampled waveform.
For example, these work areas may be used to plot graphs of data, convolve or deconvolve two waveforms. Many other functions which are available are described in detail below. The basic theory behind each of the signal processing commands is covered in Chapter 2 of the main text.

A3.3 PROGRAM DESCRIPTION

WAVEPACK calls several useful subroutines which are also listed here. They are:

1. FFT a subroutine which implements the fast Fourier transform.

2. INVFFT computes the inverse fast Fourier transform.

3. LINFLD allows the program to have flexible command formats.

4. GRAPHW and GRAPHV generate a character plot of the data in any work area.

5. PRINT lists the data in any work area.

6. WINDOW applies various data windows to any work area.

7. SLICIT operates as an interface for the Calcomp plotter so that accurate plots of the data in any work area may be generated.
8. NEWPAG and PAGNAM are subroutines required to interface to the Graphics Display Package and are system library commands (hence not listed here).

Specification of the number of data elements in the work areas or the number of temporary storage areas are declared in the code as parameters. These may be changed by the user if the need arises by simply changing the source code, recompiling and collecting. In the listing given here the number of data is 64 points per work area. The parameters are:

1. M and NPTS, where NPTS specifies the number of data elements required for each work area and M is simply found from the relation $NPTS = 2^M$.

2. NFLDS specifies the maximum number of input fields required for the WAVEPACK commands. Commas separate each field in a line entered by the user. If modifications are made and commands added which require more than 5 fields then this parameter should be changed accordingly.

3. CLNGTH is a parameter which specifies the maximum number of characters in any command.

4. FLNGTH specifies the maximum number of characters in each field.
5. NSTORS specifies the number of temporary storage areas available (see Figure A3.1).

6. NWAREA specifies the number of work areas available. Changes to source code will, however, be required if the functions available are to be implemented using more than four work areas.

7. PFILE specifies the device number which will refer to the printfile.

The program allows the user to set up labels for each temporary storage area and each work area. These labels may then be listed so that each storage area can be easily identified. The labels which are printed for the graphs and data listings may have a maximum length of 12 characters.

A3.4 THE WAVEPACK COMMANDS

Notation:

XXX = WA1, WA2, WA3 or WA4
A = an integer 0 ≤ A ≤ NPTS - 1
B = an integer A ≤ B ≤ NPTS - 1
N = an integer the value of which depends upon the particular command.

C, STDEV, AMPL
SEED = any real number
The Commands:

1. a. HELP  
   Prints an index of the WAVEPACK commands.

   b. HELP,ALL  
   Prints all the WAVEPACK commands page by page.

   c. HELP,A  
   Prints the commands on page A.

2. SAVE,N,XXX  
   Stores work area XXX permanently in Fortran file number N+10.

3. RETN,N,XXX  
   Returns work area XXX from Fortran file number N+10.

4. RCL,N,XXX  
   Recalls work area XXX from temporary store N.

5. SLABEL,XXX,'label'  
   Sets the label of work area XXX.

6. RLABEL,XXX  
   Prints the current label of work area XXX.

7. SLABEL,N,'label'  
   Sets the label of temporary store N.

8. RLABEL,N  
   Prints the current label of temporary store N.

9. LLLABEL  
   Lists all the current used defined labels.

10. PRINT,XXX  
    Prints the data in work area XXX on the vdu screen.

11. PRINTF,XXX  
    Sends the data in work area XXX to the print file.
13. GRAPH,XXX  
Plots a character graph of work area XXX on the vdu screen.

14. GRAPHF,XXX  
Sends a character graph of the data in work area XXX to the printfile.

15. FFT  
Calculates the fast Fourier transform of a complex function using the data in work area 1 as the real part and the data in work area 2 as the imaginary part. The real and imaginary parts of the resulting Fourier transform are returned to work areas 1 and 2 respectively.

16. INVFFT  
Computes the inverse fast Fourier transform using the work areas as for the FFT command above.

16. WINDOW,XXX  
This command applies one of five windows to work area XXX. On execution a further prompt is issued for the user to type in the window code and window centre. The window codes are:
1 - zero order window
2 - first order window
3 - Hanning window
4 - Hamming window
5 - Blackman window
The window centre is specified by typing in an integer value of the desired x-axis centre point. The various windows and their Fourier transform are plotted in Figure A3.4 and an example of the use of this command is given in Section A3.6.
18. R-P
This command initiates a rectangular to polar conversion with work areas 1 and 2 as the real and imaginary parts on input which are replaced by the magnitude and angle respectively on output.

19. P-R
This initiates a polar to rectangular conversion with the data in work areas 1 and 2 as the magnitude and phase components on input which are replaced with the real and imaginary parts respectively as output.

20. CONV
This command implements the cyclic convolution of the data contained in work areas 1 and 2. The result is placed in work area 3.

21. a. DECONV1
This command is the first of three which implement the step by step deconvolution of two waveforms. Initially WA1 = Re(x), WA2 = Im(x), WA3 = Re(y) and WA4 = Im(y). After execution of DECONV1 we will have WA1 = Re(FT(x)), WA2 = Im(FT(x)), WA3 = Re(FT(y)) and WA4 = Im(FT(y)).

b. DECONV2
This initiates the second stage of the deconvolution by performing WA3 = Re(FT(y)/FT(x)) and WA4 = Im(FT(y)/FT(x)). (FT(y)/FT(x)).

c. DECONV3
This is the last stage of the deconvolution and implements WA1 = Re(FT⁻¹ FT(y)/FT(x)) and WA2 = Im(FT⁻¹ FT(y)/FT(x)).
22. SUBST,XXX,A

Enters the substitute mode with work area XXX starting at data element A. This command enables the user to enter, change or delete any element of the work area. For an example of the operation of this command, see Section A3.6.

23. FILL,XXX,C,A,B

This sets the data elements from A to B inclusive of work area XXX equal to C.

24. NORM,XXX

Normalises all the data in work area XXX and prints the normalising factor.

25. SINE,XXX

Performs the sine transform on work area XXX as given by the Van Vleck relation (see Section 3.2.3).

26. ARCSINE,XXX

Performs the inverse sine transform (see above).

27. ROTATE,XXX,A

Carries out the cyclic shifting of work area XXX moving each data element A times. Positive A denotes a right shift and negative A a left shift.

28. XXX=YYY

Sets the data in any work area equal to the data in any other work area.

29. SWA1=WA2,A,B

Sets the data elements A to B in work area 1 equal to the corresponding data elements in work area 2.

30. SWA=WA1,A,B

See above.
31. \( WA1 = WA1 + WA2 \)  
   The following commands are self-explanatory

32. \( WA1 = WA1 / WA2 \)

33. \( WA1 = WA2 / WA1 \)

34. \( WA1 = WA1 + WA2 \)

35. \( WA1 = WA1 - WA2 \)

36. \( WA1 = WA2 - WA1 \)

37. \( WA1 = 1 / WA1 \)

38. \( WA1 = WA1 * , C \)  
   Multiplication of each element of work area 1 by the value C

39. \( WA1 = WA1 / , C \)  
   Division of each element of work area 1 by the value C.

40. \( WA1 = WA1 + , C \)  
   Addition of C to each element of work area 1.

41. \( WA1 = WA1 - , C \)  
   Subtraction of C from each element of work area 1.

42. \( \text{NOISE}+, XXX, \text{STDEV}, \text{AMPL}, \text{SEED} \)  
   Performs the addition of a pseudo-random noise sequence with a normally distributed amplitude to work area XXX. The standard deviation, amplitude and seed required are also entered.

43. \( \text{PLOT}, XXX \)  
   This command allows the user to send the data in work area XXX to the Calcomp plotter. A detailed example illustrating the use of this command is given in Section A3.6.
44. BAYDECRS,N

This command implements the Bayesian deconvolution described in Chapter 5 of the main text. The number of iterations desired is specified by the value N. Work area 1 should contain the input waveform data and work area 2 the output waveform data. The resulting impulse response is returned in work area 3.

45. STOP

Terminates WAVEPACK

A3.5 SUBROUTINES

The subroutines called by WAVEPACK are listed. The detailed operation of each subroutine is fairly well described by the source code explanatory notes. For specific details the reader is referred to the source code listings.

A3.6 USING THE ANALYSIS PACKAGE

A typical sequence of commands required to initiate program execution is given overleaf. It assumes that all the files required have already been catalogued. Some simplification may be achieved by incorporating these commands into a file element for use with the @ADD processor. For further details of this, refer to the UNIVAC 1106 operating manual.
Type HELP to list commands
ENTER COMMAND
>FILL, WAI, 1..0,63

WAI set equal to 1.000000 from 0 to 63 inclusive.
ENTER COMMAND
>WINDOW, WAI

Window codes are:
1 - Zero order window.
2 - First order window.
3 - Hannning window.
4 - Hamming window.
5 - Blackman window.

ENTER window code, window centre >>
>>3,32

Window number 3 applied to WAI
ENTER COMMAND
>SCALE, WAI, HANNING-WIND
ENTER COMMAND
>GRAPH, WAI
### HANNING-WIND DATA

#### DATA INDEX:

<table>
<thead>
<tr>
<th>INDEX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.002449</td>
<td>0.009773</td>
<td>0.021899</td>
<td>0.038709</td>
<td>0.060038</td>
<td>0.085677</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.115375</td>
<td>0.148842</td>
<td>0.185748</td>
<td>0.225733</td>
<td>0.268405</td>
<td>0.313346</td>
<td>0.360116</td>
<td>0.408256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.457294</td>
<td>0.506751</td>
<td>0.556412</td>
<td>0.604983</td>
<td>0.652796</td>
<td>0.699111</td>
<td>0.743476</td>
<td>0.785456</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.824639</td>
<td>0.860442</td>
<td>0.893111</td>
<td>0.921730</td>
<td>0.946217</td>
<td>0.966332</td>
<td>0.981679</td>
<td>0.992705</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.998764</td>
<td>0.998018</td>
<td>0.996035</td>
<td>0.987392</td>
<td>0.973975</td>
<td>0.955914</td>
<td>0.933387</td>
<td>0.904614</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.875958</td>
<td>0.94199</td>
<td>0.903125</td>
<td>0.742877</td>
<td>0.719544</td>
<td>0.674060</td>
<td>0.626871</td>
<td>0.578438</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>52</th>
<th>53</th>
<th>54</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.529238</td>
<td>0.479751</td>
<td>0.430462</td>
<td>0.381854</td>
<td>0.334404</td>
<td>0.288576</td>
<td>0.244820</td>
<td>0.203563</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0.145211</td>
<td>0.130118</td>
<td>0.098689</td>
<td>0.071171</td>
<td>0.047855</td>
<td>0.028967</td>
<td>0.014695</td>
<td>0.005177</td>
</tr>
</tbody>
</table>

---

**ENTER COMMAND**

`>FILL,WAI,0.0,63`

`WAI` set equal to `.000000` from 0 to 63 inclusive.

`>FILL,WAI,1.0,9`

`WAI` set equal to `1.000000` from 0 to 9 inclusive.

**ENTER COMMAND**

`>FFT`

FFT computed with Re = WAI and Im = WAI2

**ENTER COMMAND**

`>PLOT,WAI`

GMP plotting interface.

**ENTER** `>>` xaxis-label

**ENTER** `>>` yaxis-label

**ENTER** `>>` plot-title

`>>` plot-title

**ENTER** `>>` axes length in cms. xaxis-length, yaxis-length

`>>` axes length in cms. xaxis-length, yaxis-length

`>>` axes length in cms. xaxis-length, yaxis-length

`>>` axes length in cms. xaxis-length, yaxis-length

**ENTER** `>>` xmin, ymin, xmax, ymax

`>>` xmin, ymin, xmax, ymax

`>>` xmin, ymin, xmax, ymax

`>>` xmin, ymin, xmax, ymax
ENTER >> GAP page name for this plot
XFT-PULSE

WA1 plotted.
ENTER COMMAND
> FILL, WA1, 1.414, 0, 21

WA3 set equal to 1.414000 from 0 to 21 inclusive.
ENTER COMMAND
> SUBST, WA3, 19

SUBSTITUTE NODE - type E to exit.

( 19) = 1.414000
>
( 20) = 1.414000
> 34.9932
( 20) = 34.993200
>
( 21) = 1.414000
> 0.
( 21) = 0.000000
>
( 22) = 0.000000
> E

SUBSTITUTE NODE terminated.
ENTER COMMAND
> STOP

WAVEPACK TERMINATED goodbye!

>FINI
Figure A3.2 The graph generated by the runstream given here having been sent to the Calcomp plotter.
With reference to the example terminal session all the commands entered by the user are preceded by a prompt (> ) from the computer. The other messages are generated by the program. The first command used is FILL which sets all the data elements in work area 1 equal to unity. The Hanning window is then applied to this data with the window centre at 32. The work area is then labelled and plotted using the character graph. The data in the work area is then listed. The following pair of FILL commands set all the work area data to zero and then a pulse shaped function is entered. The magnitude of the Fourier transform of this pulse function is then calculated and plotted. The graph generated using this runstream was plotted on the Calcomp plotter and is given in Figure A3.2. The various titles entered into the program are seen on the plotted curve. The axis length defines the size of the plot and for an A4 size plot should be approximately 25cm x 15cm. Having generated the required plot the Graphics Display Package is used to send the plot to the system plotter.

The use of the SUBST command is also illustrated. The various data windows and their Fourier transforms are given in Figure A3.4. An example of the convolution command is shown in Figure A3.5 where rectangular and triangular pulses are convolved.
Figure A3.3 An illustration of the NOISE+ function.
A3.7 CONCLUSION

The development of this analysis package has generated some interest by other parties. The GRAPHV subroutine has been adopted by the University of Cape Town Computing Service as a library routine. Research into the application of digital filtering techniques to problems arising in radar is under way using WAVEPACK. It may thus be concluded that the facilities which are provided by the program are generally useful.
Figure A3.4a The zero-order window.

Figure A3.4b The Fourier transform of the zero-order window.

Figure A3.4c The first-order window.

Figure A3.4d The Fourier transform of the first-order window.

Figure A3.4e The Hanning window.

Figure A3.4f The Fourier transform of the Hanning window.
Figure A3.4g The Hamming window.

Figure A3.4h The Fourier transform of the Hamming window.

Figure A3.4i The Blackman window.

Figure A3.4j The Fourier transform of the Blackman window.
Figure A3.5 Convolution of rectangular and triangular pulses.
TITLE: WAVEPACK

DESCRIPTION:
This program is designed to function as an interactive waveform analysis package. Contained in this package are facilities for implementing digital signal processing. Motivation for its inception has been to deconvolve two waveforms using Fourier transform techniques. For the program to operate, four Fortran files are needed and one printfile and one plotfile.

CALLS:
FFT, INUFFT, LINFIL, GRAPHH, GRAPHU, PRINT, WINDOW, NEWPAG,
PAGFAG, SLICIT.

CODE:

Defining variables used.

**Implicit Complex (A - Z)**

PARAMETER NPTS = 44, M = 6
PARAMETER NFLDS = 5, CLNGTH = 20, NSTORS = 10
PARAMETER PFILE = 21, NUAREA = 4

CHARACTER TITLE*12 (0:NSTORS-1), TITUA*t2 (t:NUAREAl
CHARACTER *20 XLABE1, YLABE1, PTITLE
CHARACTER GDPNAM*12

INTEGER A, B, I, J, ERRFLG, WAXNO, STOHRD
INTEGER UNM, CENTRE, SHIFT, N, NTIMES
INTEGER NXCHAR, NYCHAR, NTCHAR
REAL AXIS(O:NPTS-1), UAREA(l:NUAREA,O:NPTS-1)
REAL DATA(O:NPTS-1), STORE(O:NSTORS-1,O:NPTS-1)
REAL VDATA(O:NPTS), TEMP, x, im, r, o
REAL REDACO:NPTS), REDACO:NPTS-1, PRESSO:NPTS-1)
REAL XREAL(O:NPTS), XIMAG(O:NPTS), XREAL(O:NPTS), XIMAG(O:NPTS)
REAL YREAL(O:NPTS), YIMAG(O:NPTS), YREAL(O:NPTS), YIMAG(O:NPTS)
REAL YREAL(O:NPTS), YIMAG(O:NPTS), YREAL(O:NPTS), YIMAG(O:NPTS)
REAL XAXIS, YAXIS, XMIN, XMAX, TMIN, TMAX

Setting up axis for graphing results.

**DO 1000 I = 0,NPTS-1
AXIS (I) = I**

1000 CONTINUE

Initializing labels.

**DO 2000 I = 0,NSTORS-1
TITUA(t) = *'**

2000 CONTINUE

**DO 2010 I = 1,NUAREAl
TITUA(t) = TITLE(I)**

2010 CONTINUE

Printing a heading.

**PRINT 90003
PRINT 90004**

Soliciting user for command.

**90001 PRINT 90002
READ (0,90006, ERROR=90001) COMMAND**

CALL LINFIL (COMMAND, CLNGTH, FIELD, FLNGTH, NFLDS, KNFLDE)
Testing for a command syntax error.

IF ( ERRFLG .NE. 0 ) THEN
  PRINT 99993
  GO TO 90001
ENDIF

Testing for the commands ++++++++

Implementing the SAVE and RETN commands.

Validating the contents of field three and setting WAINDX.

Validating and setting Fortran file number before reading or writing to or from disk.

IF ( WAINDX .NE. 0 ) THEN
  READ ( 90006, FIELD(2) ) STORNO
  STORNO = STORNO + 10
ENDIF

IF ( FIELD(1) .EQ. 'SAVE' ) THEN
  DO 10010 I = 0,NPTS-8,0
  WRITE ( STORNO, 98002 ) ( WAREA (WAINDX,I+J), J = 0,7 )
  CONTINUE
  PRINT 96009, WAINDX, STORNO-10
ELSE
  DO 10020 I = 0,NPTS-8,8
  READ ( STORNO, 99002 ) ( WAREA (WAINDX,I+J), J = 0,7 )
  CONTINUE
  PRINT 96007, WAINDX, STORNO-10
ENDIF
ELSE
  PRINT 99002
ENDIF
ELSE IF (FIELD(1) .EQ. "LABEL") THEN

  IF (FIELD(2) .EQ. "WA1") THEN
    TITLE(1) = FIELD(3)
  ELSE IF (FIELD(2) .EQ. "WA2") THEN
    TITLE(2) = FIELD(3)
  ELSE IF (FIELD(2) .EQ. "WA3") THEN
    TITLE(3) = FIELD(3)
  ELSE IF (FIELD(2) .EQ. "WA4") THEN
    TITLE(4) = FIELD(3)
  ELSE
    DECODE (98001, FIELD(2), ERR=11030) STORNO
    ErrFLG = 1
    DO 11010 I = 0, NSTORS-1
        IF (STORNO .EQ. I) ErrFLG = 0
    CONTINUE
    IF (ErrFLG .NE. 0) GO TO 11000
    TITLE(STORNO) = FIELD(3)
    GO TO 11020

11000 PRINT 99005
11020 CONTINUE

ENDIF

ELSE IF (FIELD(1) .EQ. "LABEL") THEN

  IF (FIELD(2) .EQ. "WA1") THEN
    PRINT 98010, 1, TITLE(1)
  ELSE IF (FIELD(2) .EQ. "WA2") THEN
    PRINT 98010, 2, TITLE(2)
  ELSE IF (FIELD(2) .EQ. "WA3") THEN
    PRINT 98010, 3, TITLE(3)
  ELSE IF (FIELD(2) .EQ. "WA4") THEN
    PRINT 98010, 4, TITLE(4)
  ELSE
    DECODE (98001, FIELD(2), ERR=11030) STORNO
    ErrFLG = 1
    DO 11040 I = 1, NWAREA
        PRINT 98010, I, TITLE(I)
    CONTINUE
    PRINT 98012, STORNO, TITLE(STORNO)
    GO TO 11050

11030 PRINT 99005
11050 CONTINUE

ENDIF

ELSE IF (FIELD(1) .EQ. "LABEL") THEN

  PRINT 98013
  DO 11070 I = 1, NWAREA
      PRINT 98010, I, TITLE(I)
  CONTINUE
  PRINT 98012, TITLE(I)

11070 CONTINUE

ENDIF

C
C
C Implementing STD and RCL commands.
C
C
ELSE IF (FIELD(1) .EQ. "STO", OR. FIELD(1) .EQ. "RCL") THEN

  Testing the validity of field 3.

  IF (FIELD(3) .EQ. "WA1") THEN
    WAIHNO = 1
  ELSE IF (FIELD(3) .EQ. "WA2") THEN
    WAIHNO = 2
  ELSE IF (FIELD(3) .EQ. "WA3") THEN
    WAIHNO = 3
  ELSE IF (FIELD(3) .EQ. "WA4") THEN
    WAIHNO = 4
  ELSE
    PRINT 99001
    WAIHNO = 0
  ENDIF
Having detected an error in field 3 exit section.

```plaintext
 erguson = -1

  DO 12020 I = 0, NSTORES-1
  IF (FIELD(2) .EQ. TITLE(I)) STORNO = 1
  CONTINUE

  IF field 3 corresponds to label - store work area.

  IF (STORNO .GE. 0) THEN
    IF (FIELD(1) .EQ. 'STO') THEN
      DO 12030 I = 0, NPTS-1
      STORE(STORNO, I) = WAREA(UINDEX, I)
      CONTINUE
      PRINT 98014, UINDEX, TITLE(STORNO)
    ELSE
      DO 12040 I = 0, NPTS-1
      WAREA(UINDEX, I) = STORE(STORNO, I)
      TITLE(UINDEX) = TITLE(STORNO)
      PRINT 98015, UINDEX, TITLE(STORNO)
    ENDIF
  ELSE
    DECODE (98001, FIELD(2), ERR=12050) STORNO
    IF (STORNO .LT. 0 .OR. STORNO .GT. NSTORES-1) GO TO 12060
    IF (FIELD(1).EQ. 'PRINT' .OR. FIELD(1).EQ. 'PRINTF' .OR. FIELD(1).EQ. 'GRAPH') THEN
      IF (FIELD(2).EQ. 'UA1') THEN
        UINDEX = 1
      ELSE IF (FIELD(2).EQ. 'UA2') THEN
        UINDEX = 2
      ELSE IF (FIELD(2).EQ. 'UA3') THEN
        UINDEX = 3
      ELSE IF (FIELD(2).EQ. 'UA4') THEN
        UINDEX = 4
      ELSE
        UINDEX = 0
      PRINT 99005
    ENDIF
    GO TO 12090
  ENDIF

  IF (STORNO .EQ. 0) GO TO 12010

  Testing to see if field 3 is a label.

  DO 12020 I = 0, NSTORES-1
  IF (FIELD(2) .EQ. TITLE(I)) STORNO = 1
  CONTINUE
```

Implementing PRINT and GRAPH commands.

```plaintext
  ELSE IF (FIELD(1).EQ. 'PRINT' .OR. FIELD(1).EQ. 'PRINTF' .OR. FIELD(1).EQ. 'GRAPH') THEN
    IF (FIELD(2).EQ. 'UA1') THEN
      UINDEX = 1
    ELSE IF (FIELD(2).EQ. 'UA2') THEN
      UINDEX = 2
    ELSE IF (FIELD(2).EQ. 'UA3') THEN
      UINDEX = 3
    ELSE IF (FIELD(2).EQ. 'UA4') THEN
      UINDEX = 4
    ELSE
      UINDEX = 0
    PRINT 99005
  ENDIF
```
Having detected an error in field 2 exit section.

```
IF ( UAINDX .NE. 0 ) THEN
  Interface a 2 dimensional array with PRINT and GRAPH.
  DO 13020 I = 0,NPTS-1
     DATA(I) = WAREA(UAINDX,I)
  CONTINUE
  IF ( FIELD(1) .EQ. 'PRINT' ) THEN
    CALL PRINT ( DATA, NPTS, TITLE(UAINDX), 5 )
  ELSE IF ( FIELD(1) .EQ. 'PRINTF' ) THEN
    CALL PRINT ( DATA, NPTS, TITLE(UAINDX), FILE )
  ELSE IF ( FIELD(1) .EQ. 'GRAPH' ) THEN
    CALL GRAPHU ( AXIS, DATA, TITLE(UAINDX),
                    XEXP, NPTS, 5 )
  ELSE
    CALL GRAPHU ( AXIS, DATA, TITLE(UAINDX),
                   XEXP, NPTS, FILE )
  ENDIF
  ENDIF

Implementing FFT (Fast Fourier Transform) command.
```

```
ELSE IF ( FIELD(1) .EQ. 'FFT' ) THEN
  DO 23010 I = 0,NPTS-1
     REDATA(I+1) = WARA(I,1)
     WDATA(I+1) = WARE(2,1)
  CONTINUE
  CALL FFT ( REDATA, WDATA, M )
  DO 23020 I = 0,NPTS-1
     WAREA(1,1) = REDATA(I+1)
     WAREA(2,1) = WDATA(I+1)
  CONTINUE
  PRINT 98018
```

```
Implementing INVFFT (Inverse Fast Fourier Transform) command.
```

```
ELSE IF ( FIELD(1) .EQ. 'INVFFT' ) THEN
  DO 23030 I = 0,NPTS-1
     REDATA(I+1) = WARE(I,1)
     WDATA(I+1) = WAREA(2,1)
  CONTINUE
  CALL INVFFT ( REDATA, WDATA, M )
  DO 23040 I = 0,NPTS-1
     WARE(1,1) = REDATA(I+1)
     WARE(2,1) = WDATA(I+1)
  CONTINUE
  PRINT 98019
  ```
ELSE IF (FIELD(1),EQ., 'WINDOW') THEN

IF (FIELD(2),EQ., 'UA') THEN
  WAINDX = 1
ELSE IF (FIELD(2),EQ., 'UA2') THEN
  WAINDX = 2
ELSE IF (FIELD(2),EQ., 'UA3') THEN
  WAINDX = 3
ELSE IF (FIELD(2),EQ., 'UA4') THEN
  WAINDX = 4
ELSE
  PRINT 99005
  WAINDX = 0
END IF

Having detected an error in field 2 exit section.

IF (WAINDX,EQ., 0) GO TO 14010

Printing window menu.

PRINT 98020
READ (8,98006) COMAND
CALL LINFDU (COMAND, CLNGTH, FIELD, FLNGTH, NFLDS, ERRFLG)
DECODA (98001, FIELD(1), ERR=14020) UMUN
DECOD (98001, FIELD(2), ERR=14030) CENTRE

Adjusting data to be in correct position before windowing.

IF (CENTRE, GE., 0 AND CENTRE, LT., NPTS/2-1) THEN
  DO 14040 I = 1, NPTS/2-1-CENTRE
      TEMP = WAREA(WAINDX,NPTS-1)
      DO 14050 J = 0, NPTS-2
          WAREA(WAINDX,NPTS-1-J) = WAREA(WAINDX,NPTS-2-J)
      14050 CONTINUE
      WAREA(WAINDX,0) = TEMP
  14040 CONTINUE
ELSE IF (CENTRE, GT., NPTS/2-1 AND CENTRE, LE., NPTS-1) THEN
  DO 14060 I = 1, CENTRE+1-NPTS/2
      TEMP = WAREA(WAINDX,I)
      DO 14070 J = 0, NPTS-2
          WAREA(WAINDX,J) = WAREA(WAINDX,J+1)
      14070 CONTINUE
  14060 TEMP
ELSE
  CONTINUE
END IF

Entering data to be windowed into an array with slightly different dimensions to interface with the WINDOW subroutine.

DO 14130 I = 0, NPTS-1
    UDATA(I+1) = WAREA(WAINDX,I)
14130 CONTINUE
CALL WINDOW (UDATA, WDATA, NPTS, NFLDS, UMUN)

Replacing windowed data into work area.

DO 14080 I = 0, NPTS-1
    WAREA(WAINDX,I) = WDATA(I+1)
14080 CONTINUE
Adjusting data back to original position.

---

IF (CENTRE .GE. 0 .AND. CENTRE .LT. NPTS/2 ) THEN
  DO 14090 I = 1,NPTS/2-CENTRE
    TEMP = WAREA(WAINDX,0)
    DO 14100 J = 0,NPTS-2
      WAREA(WAINDX,J) = WAREA(WAINDX,J+1)
    CONTINUE
    WAREA(WAINDX,NPTS-1) = TEMP
  CONTINUE
  ELSE IF (CENTRE .GT. NPTS/2 .AND. CENTRE .LE. NPTS ) THEN
    DO 14110 I = 1,CENTRE+1-NPTS/2
      WAREA(WAINDX,NPTS-1) = TEMP
    CONTINUE
    ELSE
      CONTINUE
    ENDIF
  PRINT 98021, WNUM, WAINDX
  GO TO 14010
  PRINT 99006
  GO TO 14010
ELSE IF (FIELD(I) .EQ. 'R-P' ) THEN
  DO 15010 I = 0,NPTS-1
    R = SORTK(WAREA(I,1)**2 + WAREA(I,2)**2)
    O = ATAN(WAREA(I,2)/WAREA(I,1))
    IF (WAREA(I,1) .LT. 0 .AND. WAREA(I,2) .GT. 0)
      O = O + 3.141592654
    IF (WAREA(I,1) .LT. 0 .AND. WAREA(I,2) .LT. 0)
      O = O - 3.141592654
  CONTINUE
  PRINT 98022
ELSE IF (FIELD(I) .EQ. 'P-R' ) THEN
  DO 15020 I = 0,NPTS-1
    Re = WAREA(I,1) * COS(WAREA(I,2))
    Im = WAREA(I,1) * SIN(WAREA(I,2))
    WAREA(I,1) = Re
    WAREA(I,2) = Im
  CONTINUE
  PRINT 98023
ELSE IF (FIELD(I) .EQ. 'CON' ) THEN
  DO 17010 I = 0,NPTS-1
    DATA(I) = 0.
    DO 17020 J = 0,NPTS-1
      SHIFT = I-J
      IF (SHIFT .LT. 0) SHIFT = SHIFT + NPTS
      DATA(I) = DATA(I) + WAREA(I,J) * WAREA(J,SHIFT)
    CONTINUE
  CONTINUE
ELSE IF (FIELD(I) .EQ. 'CONV' ) THEN
  DO 17010 I = 0,NPTS-1
    DATA(I) = 0.
    DO 17020 J = 0,NPTS-1
      SHIFT = I-J
      IF (SHIFT .LT. 0) SHIFT = SHIFT + NPTS
      DATA(I) = DATA(I) + WAREA(I,J) * WAREA(J,SHIFT)
    CONTINUE
  CONTINUE
  ELSE
    CONTINUE
  ENDIF
ENDIF
Setting UA3 equal to the resulting convolution.

DO 17030 I = 0,NPTS-1
    WAREA(I,1) = DATA(I)
17030    CONTINUE
    PRINT 98024

Implementing FOLD command.

ELSE IF ( FIELD(11) .EQ. 'FOLD' ) THEN
    IF ( FIELD(2) .EQ. 'UA1' ) THEN
        WAINDX = 1
    ELSE IF ( FIELD(2) .EQ. 'UA2' ) THEN
        WAINDX = 2
    ELSE IF ( FIELD(2) .EQ. 'UA3' ) THEN
        WAINDX = 3
    ELSE IF ( FIELD(2) .EQ. 'UA4' ) THEN
        WAINDX = 4
    ELSE
        PRINT 99005
        WAINDX = 0
    ENDIF

    IF ( WAINDX .NE. 0 ) THEN
        DO 16010 I = 1,NPTS/2-1
            UAREA(IJAINDX,I) = WAREA(WAINDX,NPTS-I-I)
        16010    CONTINUE
        PRINT 98025, WAINDX
    ENDIF

   ELSE IF ( FIELD(11) .EQ. 'FILL' ) THEN
    IF ( FIELD(2) .EQ. 'UA1' ) THEN
        WAINDX = 1
    ELSE IF ( FIELD(2) .EQ. 'UA2' ) THEN
        WAINDX = 2
    ELSE IF ( FIELD(2) .EQ. 'UA3' ) THEN
        WAINDX = 3
    ELSE IF ( FIELD(2) .EQ. 'UA4' ) THEN
        WAINDX = 4
    ELSE
        PRINT 99005
        WAINDX = 0
    ENDIF

    IF ( WAINDX .NE. 0 ) THEN
        DECODE ( 98001, FIELD(4), ERR=18010 ) A
        DECODE ( 98001, FIELD(5), ERR=18020 ) B
        DECODE ( 98001, FIELD(3), ERR=18030 ) TEMP
        DO 18040 I = A,B
            WAREA(WAINDX,I) = TEMP
        18040    CONTINUE
        PRINT 98026, WAINDX, TEMP, A, B
        GO TO 18050
    ENDIF

    ELSE IF ( FIELD(11) .EQ. 'EXCH' ) THEN
    DO 19010 I = 0,NPTS-1
        TEMP = WAREA(I,1)
        WAREA(I,1) = WAREA(I,2)
        WAREA(I,2) = TEMP
    19010    CONTINUE
        PRINT 98027

Implementing the EXCH command.
**Implementing NORM Command.**

```c
ELSE IF ( FIELD(1) .EQ. 'NORM' ) THEN
    IF ( FIELD(2) .EQ. 'UA1' ) THEN
        UAINDX = 1
    ELSE IF ( FIELD(2) .EQ. 'UA2' ) THEN
        UAINDX = 2
    ELSE IF ( FIELD(2) .EQ. 'UA3' ) THEN
        UAINDX = 3
    ELSE IF ( FIELD(2) .EQ. 'UA4' ) THEN
        UAINDX = 4
    ELSE
        PRINT 99005
        UAINDX = 0
ENDIF

IF ( UAINDX .NE. 0 ) THEN
    TEMP = ABS( WAREA(UAINDX,0) )
    DO 20010 I = 1,NPTS-1
        IF ( ABS( WAREA(UAINDX,1) ) .GT. TEMP )
            TEMP = ABS( WAREA(UAINDX,1) )
    CONTINUE

    DO 20020 I = 0,NPTS-1
        WAREA(UAINDX,1) = WAREA(UAINDX,1) / TEMP
    CONTINUE

    PRINT 98029, UAINDX, TEMP
END IF

ELSE IF ( FIELD(1) .EQ. 'SINE' .OR. FIELD(1) .EQ. 'ARCSINE' ) THEN
    IF ( FIELD(2) .EQ. 'UA1' ) THEN
        UAINDX = 1
    ELSE IF ( FIELD(2) .EQ. 'UA2' ) THEN
        UAINDX = 2
    ELSE IF ( FIELD(2) .EQ. 'UA3' ) THEN
        UAINDX = 3
    ELSE IF ( FIELD(2) .EQ. 'UA4' ) THEN
        UAINDX = 4
    ELSE
        PRINT 99005
        UAINDX = 0
ENDIF

IF ( UAINDX .NE. 0 ) THEN
    TEMP = ABS( WAREA(UAINDX,0) )
    DO 21010 I = 1,NPTS-1
        IF ( ABS( WAREA(UAINDX,1) ) .GT. TEMP )
            TEMP = ABS( WAREA(UAINDX,1) )
    CONTINUE

    IF ( FIELD(1) .EQ. 'SINE' ) THEN
        DO 21020 I = 0,NPTS-1
            WAREA(UAINDX,1) = TEMP * SIN( WAREA(UAINDX,1)/TEMP ) * 1.570796327
        CONTINUE
        PRINT 98030, UAINDX
    ELSE
        DO 21030 I = 0,NPTS-1
            WAREA(UAINDX,1) = TEMP * 0.636619772 * ASIN( WAREA(UAINDX,1) / TEMP )
        CONTINUE
        PRINT 98031, UAINDX
    END IF
ELSE
    PRINT 98032, UAINDX
END IF
```

**Implementing the SINE and ARCSINE Commands.**

```c
```
Implementing the ROTATE command.

```
ELSE IF (FIELD(1) .EQ. 'ROTATE') THEN
  IF (FIELD(2) .EQ. 'UA1') THEN
    UAINDX = 1
  ELSE IF (FIELD(2) .EQ. 'UA2') THEN
    UAINDX = 2
  ELSE IF (FIELD(2) .EQ. 'UA3') THEN
    UAINDX = 3
  ELSE IF (FIELD(2) .EQ. 'UA4') THEN
    UAINDX = 4
  ELSE
    PRINT 99005
    UAINDX = 0
ENDIF

IF (UAINDX .NE. 0) THEN
  DECODE (98001, FIELD(3), ERR=22010) A

  IF (A .GT. 0) THEN
    DO 22020 I = 1, A
      TEMP = AREA(UAINDX, NPTS-1)
      DO 22020 J = 0, NPTS-2
        AREA(UAINDX, NPTS-1-J) = AREA(UAINDX, NPTS-2-J)
  CONTINUE
  CONTINUE
  PRINT 98032, UAINDX, A

  ELSE IF (A .LT. 0) THEN
    DO 22040 I = 1, ABS(A)
      TEMP = AREA(UAINDX, 0)
      DO 22040 J = 0, NPTS-2
        AREA(UAINDX, J) = AREA(UAINDX, J+1)
  CONTINUE
  CONTINUE
  PRINT 98033, UAINDX, ABS(A)

  ELSE
    CONTINUE
ENDIF

ELSE
  GO TO 22060
C 22010 PRINT 99001
22060 CONTINUE
C ENDIF
```

Implementing arithmetic commands.

```
ELSE IF (FIELD(1) .EQ. 'UA1=UA2') THEN
  DO 25010 I = 0, NPTS-1
    AREA(I,1) = AREA(2,I)
  CONTINUE
  PRINT 98035, 1, 2

ELSE IF (FIELD(1) .EQ. 'UA2=UA1') THEN
  DO 25020 I = 0, NPTS-1
    AREA(2,I) = AREA(1,I)
  CONTINUE
  PRINT 98035, 2, 1

ELSE IF (FIELD(1) .EQ. 'UA1=UA3') THEN
  DO 25030 I = 0, NPTS-1
    AREA(I,1) = AREA(3,I)
  CONTINUE
  PRINT 98035, 1, 3

ELSE IF (FIELD(1) .EQ. 'UA3=UA1') THEN
  DO 25040 I = 0, NPTS-1
    AREA(3,I) = AREA(1,I)
  CONTINUE
  PRINT 98035, 1, 3

ELSE IF (FIELD(1) .EQ. 'UA1=UA4') THEN
  DO 25050 I = 0, NPTS-1
    AREA(I,1) = AREA(4,I)
  CONTINUE
  PRINT 98035, 1, 4

ELSE IF (FIELD(1) .EQ. 'UA4=UA1') THEN
  DO 25060 I = 0, NPTS-1
    AREA(4,I) = AREA(1,I)
  CONTINUE
  PRINT 98035, 1, 4
```
ELSE IF (FIELD(1).EQ. 'UA1=UA4') THEN
   DO 25510 I = 0,NPTS-1
      UAREA(I,1) = UAREA(4,I)
   CONTINUE
   PRINT 90035, 1, 4

IMPLEMENTING UAA=UA3 AND UAA-UA4.

ELSE IF (FIELD(1).EQ. 'UA2-UA3') THEN
   DO 25580 I = 0,NPTS-1
      UAREA(I,1) = UAREA(3,I)
   CONTINUE
   PRINT 90035, 2, 3

ELSE IF (FIELD(1).EQ. 'UA2-UA4') THEN
   DO 25590 I = 0,NPTS-1
      UAREA(I,1) = UAREA(4,I)
   CONTINUE
   PRINT 90035, 2, 4

IMPLEMENTING UA3 = UA1 AND UA3 = UA2.

ELSE IF (FIELD(1).EQ. 'UA3-UA4') THEN
   DO 25520 I = 0,NPTS-1
      UAREA(I,1) = UAREA(1,I)
   CONTINUE
   PRINT 90035, 3, 1

ELSE IF (FIELD(1).EQ. 'UA4-UA2') THEN
   DO 25530 I = 0,NPTS-1
      UAREA(I,1) = UAREA(2,I)
   CONTINUE
   PRINT 90035, 3, 2

IMPLEMENTING UA3 = UA4 AND UA4 = UA1.

ELSE IF (FIELD(1).EQ. 'UA3-UA4') THEN
   DO 25540 I = 0,NPTS-1
      UAREA(I,1) = UAREA(4,I)
   CONTINUE
   PRINT 90035, 3, 4

ELSE IF (FIELD(1).EQ. 'UA4-UA1') THEN
   DO 25550 I = 0,NPTS-1
      UAREA(I,1) = UAREA(1,I)
   CONTINUE
   PRINT 90035, 4, 1

IMPLEMENTING UA4 = UA2 AND UA4 = UA3.

ELSE IF (FIELD(1).EQ. 'UA4-UA2') THEN
   DO 25560 I = 0,NPTS-1
      UAREA(I,1) = UAREA(2,I)
   CONTINUE
   PRINT 90035, 4, 2

ELSE IF (FIELD(1).EQ. 'UA4-UA3') THEN
   DO 25570 I = 0,NPTS-1
      UAREA(I,1) = UAREA(3,I)
   CONTINUE
   PRINT 90035, 4, 3
Setting selected areas of work area equal.

ELSE IF (FIELD(1) .EQ. 'WIA=WIA2' .OR. FIELD(1) .EQ. 'WA1=WA2') THEN

DO 25050 I = A, B
WAREA(1,I) = WAREA(2,I)
25050 CONTINUE
PRINT 98036, A, B

ELSE
DO 25040 I = A, B
WAREA(2,I) = WAREA(I)
25040 CONTINUE
PRINT 98037, A, B
ENDIF

ELSE IF (FIELD(1) .EQ. 'WIA=WIA2') THEN
DO 25090 I = 0, NPTS-1
WAREA(1,I) = WAREA(1,I) / WAREA(2,I)
25090 CONTINUE
PRINT 98038

ELSE IF (FIELD(1) .EQ. 'WA1=WA1/UA2') THEN
DO 25100 I = 0, NPTS-1
WAREA(1,I) = WAREA(1,I) / WAREA(2,I)
25100 CONTINUE
PRINT 99040

ELSE IF (FIELD(1) .EQ. 'UA1=UA1+WA2') THEN
DO 25110 I = 0, NPTS-1
WAREA(1,I) = WAREA(1,I) + WAREA(2,I)
25110 CONTINUE
PRINT 99041
! Implementing WA1=UA1-UA2 and WA1=UA2-UA1.

C ELSE IF ( FIELD(1) .EQ. 'WA1=UA1-UA2' ) THEN
  DO 25120 I = 0,NPTS-1
  WAREA(1,I) = WAREA(1,I) - WAREA(2,I)
  CONTINUE
  PRINT 98042

C ELSE IF ( FIELD(1) .EQ. 'WA1=UA2-UA1' ) THEN
  DO 25130 I = 0,NPTS-1
  WAREA(1,I) = WAREA(2,I) - WAREA(1,I)
  CONTINUE
  PRINT 98043

C ! Implementing WA1=UA1,constant.

C ELSE IF ( FIELD(1) .EQ. 'WA1=UA1*' ) THEN
  DECODE ( 98001, FIELD(2), ERR=25300 ) TEMP
  DO 25140 I = 0,NPTS-1
  WAREA(1,I) = WAREA(1,I) * TEMP
  CONTINUE
  PRINT 98044, TEMP
  GO TO 25150

C 25300 PRINT 99005
25150 CONTINUE

C ! Implementing WA1=UA1/constant.

C ELSE IF ( FIELD(1) .EQ. 'WA1=UA1/' ) THEN
  DECODE ( 98001, FIELD(2), ERR=25160 ) TEMP
  DO 25170 I = 0,NPTS-1
  WAREA(1,I) = WAREA(1,I) / TEMP
  CONTINUE
  PRINT 98045, TEMP
  GO TO 25180

C 25160 PRINT 99005
25180 CONTINUE

C ! Implementing WA1=UA1/constant.

C ELSE IF ( FIELD(1) .EQ. 'WA1=UA1+' ) THEN
  DECODE ( 98001, FIELD(2), ERR=25190 ) TEMP
  DO 25200 I = 0,NPTS-1
  WAREA(1,I) = WAREA(1,I) + TEMP
  CONTINUE
  PRINT 98046, TEMP
  GO TO 25210

C 25190 PRINT 99005
25210 CONTINUE

C
ELSE IF (FIELD(1) .EQ. "UA1=UA1-" ) THEN

DECODE ( 98001, FIELD(2), ERR=25220 ) TEMP
DO 25230 I = 0,NPTS-1
   UAREA(1,I) = TEMP(1,I) - TEMP
25230 CONTINUE

PRINT 98047, TEMP
GO TO 25240

25220 PRINT 99005
25240 CONTINUE

ELSE IF ( FIELD(1) .EQ. "UA1=UA1-" ) THEN

DO 25250 I = 0,NPTS-1
   UAREA(1,I) = 1 / UAREA(1,I)
25250 CONTINUE

PRINT 98048

ELSE IF ( FIELD(1) .EQ. "UA1=UA1-" ) THEN

DO 25250 I = 0,NPTS-1
   UAREA(1,I) = 1 / UAREA(1,I)
25250 CONTINUE

PRINT 98048

ELSE IF ( FIELD(1) .EQ. "UA1=UA1-" ) THEN

DO 25250 I = 0,NPTS-1
   UAREA(1,I) = 1 / UAREA(1,I)
25250 CONTINUE

PRINT 98048

ELSE IF ( FIELD(1) .EQ. "UA1=UA1-" ) THEN

DO 25250 I = 0,NPTS-1
   UAREA(1,I) = 1 / UAREA(1,I)
25250 CONTINUE

PRINT 98048

ENDIF

ELSE IF ( FIELD(1) .EQ. "SUBST" ) THEN

IF ( FIELD(2) .EQ. "UA1" ) THEN
   UAINDX = 1
ELSE IF ( FIELD(2) .EQ. "UA2" ) THEN
   UAINDX = 2
ELSE IF ( FIELD(2) .EQ. "UA3" ) THEN
   UAINDX = 3
ELSE IF ( FIELD(2) .EQ. "UA4" ) THEN
   UAINDX = 4
ELSE
   PRINT 99005
   UAINDX = 0
ENDIF

IF ( UAINDX .NE. 0 ) THEN

DECODE ( 98001, FIELD(3), ERR=26050 ) A
IF ( A .GE. 0 .AND. A .LE. NPTS-1 ) THEN
   PRINT 99049
   I = A
   PRINT 98050, TITUAINDX, I, UAREA(UAINDX,I)
END IF

CALL LINFLD ( COMAND, CLNGTH, FIELD, FLENGTH, NFLDS, ERRFL6 )

IF ( FIELD(1) .EQ. "E" ) GO TO 26010
IF ( FIELD(1) .EQ. " " ) THEN
   I = I +1
ELSE
   DECODE ( 98001, FIELD(1), ERR=26020 ) TEMP
   UAREA(UAINDX,I) = TEMP
END IF
GO TO 26030

26020 PRINT 99006
26030 CONTINUE.

ENDIF

GO TO 26040

ELSE
   PRINT 99009
ENDIF

26010 PRINT 98051
GO TO 26060

26050 PRINT 99001
26060 CONTINUE

ENDIF
Implementing DECONVOLUTION commands.

Implementing DECONV1.

ELSE IF (FIELD(1).EQ. 'DECONV1') THEN

DO 27010 I = 0,NPTS-1
  REDATA(I+1) = UAREA(1,1)
  UDATA(I+1) = UAREA(2,1)
27010 CONTINUE

CALL FFT (REDATA, UDATA, M)

DO 27020 I = 0,NPTS-1
  UAREA(1,1) = REDATA(I+1)
  UAREA(2,1) = UDATA(I+1)
  REDATA(I+1) = UAREA(1,1)
  UDATA(I+1) = UAREA(2,1)
27020 CONTINUE

CALL FFT (REDATA, UDATA, M)

DO 27040 I = 0,NPTS-1
  UAREA(1,1) = REDATA(I+1)
  UAREA(2,1) = UDATA(I+1)
27040 CONTINUE

PRINT 98053

C 1Mplete1entin'3 DECONV2 
COf'll'l<md.

ELSE IF (FIELD(1).EQ. 'DECONV2') THEN

DO 27050 I = 0,NPTS-1
  TEMP = UAREA(1,1)*UAREA(1,1)+UAREA(2,1)*UAREA(2,1)
  Re = UAREA(3,1)
  Im = UAREA(4,1)
  UAREA(3,1) = (Re*UAREA(1,1)+Im*UAREA(2,1)) / TEMP
  UAREA(4,1) = (Im*UAREA(1,1)-Re*UAREA(2,1)) / TEMP
27050 CONTINUE

PRINT 98054

C 1Mplete1entin'3 DECONV3 
COf'll'l<md.

ELSE IF (FIELD(1).EQ. 'DECONV3') THEN

DO 27060 I = 0,NPTS-1
  REDATA(I+1) = UAREA(3,1)
  UDATA(I+1) = UAREA(4,1)
27060 CONTINUE

CALL INVFFT (REDATA, UDATA, M)

DO 27070 I = 0,NPTS-1
  UAREA(1,1) = REDATA(I+1)
  UAREA(2,1) = UDATA(I+1)
27070 CONTINUE

PRINT 98055

C 1Mplete1entin'3 ELPl 
COf'll'l<md.

ELSE IF (FIELD(1).EQ. 'HELP') THEN
IF ( FIELD(2) .EQ. '1' ) THEN
  PRINT 98061
ELSE IF ( FIELD(2) .EQ. '2' ) THEN
  PRINT 98058
ELSE IF ( FIELD(2) .EQ. '3' ) THEN
  PRINT 98059
ELSE IF ( FIELD(2) .EQ. '4' ) THEN
  PRINT 98060
ELSE IF ( FIELD(2) .EQ. 'ALL' ) THEN
  PRINT 98057
  PAUSE 'Type CR for next page.'
  PRINT 98058
  PAUSE 'Type CR for next page.'
  PRINT 98059
  PAUSE 'Type CR for next page.'
  PRINT 98060
ELSE
  PRINT 98061
ENDIF

Implementing NOISE+ command.
=================================
ELSE IF ( FIELD(1) .EQ. 'NOISE+' ) THEN

  IF ( FIELD(2) .EQ. 'WA1' ) THEN
    WAINDX = 1
  ELSE IF ( FIELD(2) .EQ. 'WA2' ) THEN
    WAINDX = 2
  ELSE IF ( FIELD(2) .EQ. 'WA3' ) THEN
    WAINDX = 3
  ELSE IF ( FIELD(2) .EQ. 'WA4' ) THEN
    WAINDX = 4
  ELSE
    PRINT 99005
    WAINDX = 0
ENDIF

  DECODE ( 98001, FIELD(3), ERR=28010 ) TEMP
  DECODE ( 98001, FIELD(4), ERR=28010 ) r
  DECODE ( 98001, FIELD(5), ERR=28010 ) REDATA(1)

  CALL RANDN ( REDATA, NPTS, 0., TEMP )

DD 28020 I = 0,NPTS-1
  WAREA=WAREA(WAINDX,1)+r*REDATA(I+1)
28020 CONTINUE
PRINT 98063, WAINDX
GO TO 28030

28030 CONTINUE

Implementing PLOT command.
============================
ELSE IF ( FIELD(1) .EQ. 'PLOT' ) THEN

  IF ( FIELD(2) .EQ. 'WA1' ) THEN
    WAINDX = 1
  ELSE IF ( FIELD(2) .EQ. 'WA2' ) THEN
    WAINDX = 2
  ELSE IF ( FIELD(2) .EQ. 'WA3' ) THEN
    WAINDX = 3
  ELSE IF ( FIELD(2) .EQ. 'WA4' ) THEN
    WAINDX = 4
  ELSE
    WAINDX = 0
    PRINT 99005
ENDIF

  DO 29010 I = 0,NPTS-1
    DATA(I) = WAREA(WAINDX,1)
 29010 CONTINUE
!PRINT 90065
!PRINT 90066
READ (8, 90066) COMMAND
CALL LINFLD (COMMAND, CLNGTH, FIELD, FLAGTH, NFLDS, ERRFLG)
XLABEL = FIELD(1)
!PRINT 90067
READ (8, 90066) COMMAND
CALL LINFLD (COMMAND, CLNGTH, FIELD, FLAGTH, NFLDS, ERRFLG)
YLABEL = FIELD(1)
!PRINT 90068
READ (8, 90066) COMMAND
CALL LINFLD (COMMAND, CLNGTH, FIELD, FLAGTH, NFLDS, ERRFLG)
TITLE = FIELD(1)
!PRINT 90069
READ 98001, XAXIS, YAXIS
!PRINT 90070
READ 98001, XMAX, XMIN, YMAX, YMIN
!PRINT 90071
READ 98002, GDPNAM

Initiate plot.

CALL NEWPAG
CALL PAGNAM (GDPNAM)
CALL SLICIT (AXIS, DATA, XAXIS, YAXIS, XMAX, XMIN, YMAX, YMIN, YAX, ILABEL, 20, TLABEL, 20, TITLE, 20, NPTS)
!PRINT 90073, WAINDX
ENDIF

Implementing BAYDECRS command.

ELSE IF (FIELD(1) .EQ. 'BAYDECRS') THEN
DECIDE (98001, FIELD(2)) NTIMES

Setting i=y.

DO 31010 I = 1, NPTS-1
WAREAC3(I) = WAREAC2(I)
CONTINUE

DO 31020 N = 1, NTIMES
Calculating p=xg.

DO 31030 I = 1, NPTS-1
PRSA(I) = 0.
DO 31040 J = 0, NPTS-1
SHIFT = I - J
IF (SHIFT .LE. 0) SHIFT = SHIFT + NPTS
PRSA(I) = PRSA(I) + WAREA(I, J) * WAREA(3, SHIFT)
CONTINUE

Calculating s=y/p.

DO 31050 I = 0, NPTS-1
FRSA(I) = WAREA(2, I) / FRSA(I)
CONTINUE

Calculating p=x/s.

DO 31060 I = 0, NPTS-1
FRSA(I) = 0.
DO 31070 J = 0, NPTS-1
SHIFT = MOD(I + J, NPTS)
FRSA(I) = FRSA(I) + WAREA(1, J) * SSA(SHIFT)
CONTINUE

CONTINUE
Calculating grs.p

DO 31000 I = 0,NPTS-1
    AREA(3,I) = AREA(3,I) + PRSA(I)
31000 CONTINUE

DO 31020 I = 0,NPTS-1
    AREA(3,I) = AREA(3,I) + PRSA(I)
31020 CONTINUE

PRINT 98075
PRINT *,' using real space approach.'

C Implementing CORREL command.
C ========================================================
C
ELSE IF ( FIELD(1) .EQ. 'CORREL' ) THEN

DO 32010 I = 0,NPTS-1
    AREA(3,I) = 0.
    DO 32020 J = 0,NPTS-1
        SHIFT = MOD(I+J,NPTS)
        AREA(3,I) = AREA(3,I) + AREA(1,J) + AREA(2,SHIFT)

32020 CONTINUE
32010 CONTINUE

C PRINT 98085

C Implementing SUM command.
C ========================================================
C
ELSE IF ( FIELD(1) .EQ. 'SUM' ) THEN

IF ( FIELD(2) .EQ. 'WA1' ) THEN
    WAINDX = 1
ELSE IF ( FIELD(2) .EQ. 'WA2' ) THEN
    WAINDX = 2
ELSE IF ( FIELD(2) .EQ. 'WA3' ) THEN
    WAINDX = 3
ELSE IF ( FIELD(2) .EQ. 'WA4' ) THEN
    WAINDX = 4
ELSE
    PRINT 99005
    WAINDX = 0
ENDIF

IF ( WAINDX .NE. 0 ) THEN

    TEMP = 0.
    DO 33010 I = 0,NPTS-1
        TEMP = TEMP + AREA(WAINDX,I)
    33010 CONTINUE

C PRINT 98087, WAINDX, TEMP
C ENDIF

C Implementing STOP command.
C ========================================================
C
ELSE IF ( FIELD(1) .EQ. 'STOP' ) THEN

PRINT 98007
STOP
.

C Printing error message if command is unrecognized.
C ========================================================
C
ELSE
    PRINT 99004
C ENDIF
Going back for next command.

FORMAT STATEMENTS.

98001 FORMAT ( )
98002 FORMAT ( 8(3X,F12.6) )
98003 FORMAT ( 1H1,/,3X,'WAVEPACK V1.3 12/6/81',///)
98004 FORMAT ( /,3X,'Type HELP to list commands')
98005 FORMAT ( 3X,'ENTER COMMAND')
98006 FORMAT ( ABO )
98007 FORMAT ( /,3X,'WAVEPACK TERMINATED goodbye!',///)
98008 FORMAT ( /,3X,'WA',11,' returned from store number ',11,/)　
98009 FORMAT ( /,3X,'WA',11,' label is ',A12 )　
98010 FORMAT ( /,3X,'WA',11,' stored in store ',12,/)　
98011 FORMAT ( /,3X,'Window codes are :',/,
12,3X,'1 - Zero order window.',
12,3X,'2 - First order window.',
12,3X,'3 - Hann window.',
12,3X,'4 - Hamming window.',
12,3X,'5 - Blackman window.',
12,3X,'ENTER window code, window centre')　
98012 FORMAT ( /,3X,'Window number ',11,applied to WA',11,/)　
98013 FORMAT ( /,3X,'Rectangular to polar conversion carried out.')　
98014 FORMAT ( /,3X,'Convolution of WA3 = WA1 * WA2 carried out. ')　
98015 FORMAT ( /,3X,'Rectangular to polar conversion carried out.' )　
98016 FORMAT ( /,3X,'Arcsine transform applied to WA',11,/)　
98017 FORMAT ( /,3X,'Sine transform applied to WA',11,/)　
98018 FORMAT ( /,3X,'Window codes are :',/,
12,3X,'1 - Zero order window.',
12,3X,'2 - First order window.',
12,3X,'3 - Hann window.',
12,3X,'4 - Hamming window.',
12,3X,'5 - Blackman window.',
12,3X,'ENTER window code, window centre')　
98019 FORMAT ( /,3X,'Window number ',11,applied to WA',11,/)　
98020 FORMAT ( /,3X,'Rectangular to polar conversion carried out. ')　
98021 FORMAT ( /,3X,'Polar to rectangular conversion carried out.' )　
98022 FORMAT ( /,3X,'Convolution of WA3 = WA1 * WA2 carried out. ')　
98023 FORMAT ( /,3X,'Rectangular to polar conversion carried out.' )　
98024 FORMAT ( /,3X,'Arcsine transform applied to WA',11,/)　
98025 FORMAT ( /,3X,'Sine transform applied to WA',11,/)　
98026 FORMAT ( /,3X,'WA',11,' set equal to ',F12.6, from ',
12,3X,'13,' to ',13, inclusive.' )　
98027 FORMAT ( /,3X,'WA1 and WA2 values have been exchanged.' )　
98028 FORMAT ( /,3X,'WA',11,' normalized by factor = ',F24.6 )　
98029 FORMAT ( /,3X,'WA',11,' set equal to WA',11,/)　
98030 FORMAT ( /,3X,'Arcsine transform applied to WA',11,/)　
98031 FORMAT ( /,3X,'Sine transform applied to WA',11,/)　
98032 FORMAT ( /,3X,'WA',11,' rotated ',11, ' positions right.' )　
98033 FORMAT ( /,3X,'WA',11,' rotated ',13, ' positions left.' )　
98034 FORMAT ( /,3X,'WA',11, 'set equal to WA',11,/)　
98035 FORMAT ( /,3X,'WA',11, 'set equal to WA',11,)
& FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
& & FORMAT
C
90060 FORMAT (/,5X,'SUBST,XXX,A - enter substitute mode'
+ 'with xxx at element A',
+ '/,5X,'STOP - terminates WAVEPACK',
+ '/,5X,'HELP,A - prints these commands or page A only',
+ '/,5X,'NOISE+,XXX,stdev,split,seed - adds noise to xxx'
+ '/,5X,'SUM,xxx - computes the summation of all the data',
+ '/,5X,'elements of work area xxx',
+ '/,5X,'BAYESIAN,N - implements Bayesian deconvolution N times',
+ '/,5X,'CORREL - computes crosscorrelation of WAI and WA2',
+ 'result in WA3'.)

C
90061 FORMAT (/,3X,'Index to commands $1',/1X,22('=',*'),*'
+ '/,5X,'page 1 -- starting and recalling data,label commands',
+ '/,5X,'page 2 -- displaying results,spectral processing',
+ '/,5X,'page 3 -- sine transforms,arithmetic commands',
+ '/,5X,'page 4 -- substitute,help,stop commands',
+ '/,5X,'TYPE - HELP,A where A = desired page number,'/

C
90063 FORMAT (/,3X,'Noise added to WA',',I1')
C
90075 FORMAT (/,3X,'Bayesian deconvolution carried out')
C
90075 FORMAT (/,3X,'GDP plotting interface.')
C
90266 FORMAT (/,3X,'ENTER >> xaxis-label')
C
90067 FORMAT (/,3X,'ENTER >> yaxis-label')
C
90068 FORMAT (/,3X,'ENTER >> plot-title')
C
90069 FORMAT (/,3X,'ENTER >> axes length in cms. xaxis-length'
+ 'yaxis-length')
C
90070 FORMAT (/,3X,'ENTER >> xaxis,xmin,xmax,ymin')
C
90071 FORMAT (/,3X,'ENTER >> GDP page name for this plot')
C
90072 FORMAT (A12)
C
90073 FORMAT (/,3X,'WA',',I1,' plotted.')
C
90085 FORMAT (/,3X,'Correlation of WAI = WA1 & WA2 carried out.')
C
90087 FORMAT (/,3X,'Sum of all the elements of WA',',I1,' =','F12.6')

C
C
C
C
99001 FORMAT (/,3X,'*** ERROR - contents of field 3 is',
+ 'unrecognizable ***')
C
99002 FORMAT (/,3X,'*** ERROR - contents of field 2 out',
+ 'of range ***')
C
99003 FORMAT (/,3X,'*** ERROR - command syntax ***')
C
99004 FORMAT (/,3X,'*** ERROR - unrecognizable command ***')
C
99005 FORMAT (/,3X,'*** ERROR - contents of field 2',
+ 'unrecognizable ***')
C
99006 FORMAT (/,3X,'*** ERROR - contents of field 1',
+ 'unrecognizable ***')
C
99007 FORMAT (/,3X,'*** ERROR - contents of field 4',
+ 'unrecognizable ***')
C
99008 FORMAT (/,3X,'*** ERROR - contents of field 5',
+ 'unrecognizable ***')
C
99009 FORMAT (/,3X,'*** ERROR - contents of field 3',
+ 'out of range ***')
C
99010 FORMAT (/,3X,'*** ERROR - noise command error ***')
C
END
ENP OF FILE
TITLE: LINFLD

DESCRIPTION:

This subroutine accepts as input a character variable containing the input card just read. LINFLD then breaks up the input card and into fields separated by commas and returns these fields to the calling program. Termination of input occurs when the first blank in the input string is detected. An error flag is available if needed. All the fields are cleared and the error flag is set or cleared.

The call line is:

CALL LINFLD (LINEIN, LENGTH, FLIGOUT, FLAGS, NFLDS, ERFILG)

The parameters are:

LINEIN - a character variable containing the input record (input).
LENGTH - an integer specifying the number of characters in LINEIN must be set to 80 in the main program (input).
FLIGOUT - a character array of dimension NFLDS (input).
LENGTH - an integer specifying the maximum number of alphanumeric characters in each FLIGOUT character variable. Must be set to 20 in the main program (input).
NFLDS - an integer specifying the number of character variables in the FLIGOUT character array, probably set as a parameter in the main program (input).
ERFILG - an integer set by the subroutine:

0 = no error
1 = error

An error may arise if the specified field length or line length is exceeded.

In the main program the required definitions are:

CHARACTER LINEIN*80, FLIGOUT*20 (1NFLDS)

CALLS: nothing

CODE:

SUBROUTINE LINFLD (LINEIN, LENGTH, FLIGOUT, FLGTH, NFLDS, ERFILG)

Defining the variables used.

============================

INTEGER LENGTH, FLGTH, NFLDS, ERFILG
INTEGER COUNT, FCOUNT, FLNO, I, J

CHARACTER LINEIN*80, FLIGOUT*20 (1NFLDS)

Initializing variables.

============================

ERFILG = 0
20 100 I=1,NFLDS
     FLIGOUT(I) = 20
100 CONTINUE

FLNO = 1
     CCOUNT = 1
     FCOUNT = 1
C Sorting the fields from the input record.
C =================================================================================
C 55 IF ( LINEIN(COUNT:COUNT) .EQ. "" ) THEN
C     RETURN
C     ELSE
C     IF ( LINEIN(COUNT:COUNT) .EQ. "", ) THEN
C     FLDNO = FLDNO+1
C     COUNT = COUNT +1
C     IF ( FLDNO .GT. FLDGS ) THEN
C     ERFLAG = 1
C     RETURN
C     ELSE
C     GO TO 55
C     ENDIF
C     ELSE
C     FLDNO = FLDNO+(COUNT:COUNT) - LINEIN(COUNT:COUNT)
C     COUNT = COUNT +1
C     IF ( FLDNO .GT. FLDN0 ) THEN
C     ERFLAG = 2
C     RETURN
C     ELSE
C     GO TO 55
C     ENDIF
C     ENDIF
C     ENDIF
C     END
C END OF FILE

C TITLE : GRAPHW
C DESCRIPTION :
C THIS SUBROUTINE IS USED AS A DRIVER ROUTINE FOR THE
C GRAPHW SUBROUTINE WHICH PRINTS A CHARACTER GRAPH
C ON THE VDU SCREEN. THIS ROUTINE ACCEPTS AS INPUT
C THE DATA ARRAY TO BE PLOTTED AND NORMALISES IT BEFORE
C PLOTTING. IT CALLS GRAPHW.
C CALLS : GRAPHW
C THE CALL LINE IS :
C CALL GRAPHW ( AXIS, DATA, YEXP, XEXP, NPTS, DEVICE )
C PARAMETERS :
C AXIS - A REAL ARRAY OF DIMENSION NPTS CONTAINING THE
C X AXIS VALUES.
C DATA - A REAL ARRAY OF DIMENSION NPTS CONTAINING THE
C Y AXIS VALUES.
C YEXP - A 12 CHARACTER TITLE.
C XEXP - A 72 CHARACTER TITLE.
C NPTS - AN INTEGER SPECIFYING THE NUMBER OF POINTS TO
C BE PLOTTED.
C DEVICE - AN INTEGER SPECIFYING THE DEVICE TO WHICH
C THE OUTPUT IS TO BE SENT.
C CODE :
C SUBROUTINE GRAPHW ( AXIS, DATA, YEXP, XEXP, NPTS, DEVICE )
C C DEFINING THE VARIABLES TO BE USED
C ********************************************************************
C IMPLICIT COMPLEX (A-Z)
C INTEGER NPTS, 1, DEVICE
C REAL AXIS(C(NPTS-1)), DATA(C(NPTS-1))
C REAL MAX, MIN, MAX, MINT
C CHARACTER YEXP '12, XEXP '72

C
COMPUTING THE LIMITS OF THE GRAPH

C

C MAXX = AXIS (0)
C MINX = AXIS (0)
C MAXY = DATA (0)
C MINY = 0.
C
C DO 100 I = 0,NPTS-1
C IF ( DATA (I) .GT. MAXY ) MAXY = DATA (I)
C IF ( DATA (I) .LT. MINY ) MINY = DATA (I)
C IF ( AXIS (I) .GT. MAXX ) MAXX = AXIS (I)
C IF ( AXIS (I) .LT. MINX ) MINX = AXIS (I)
C
C END
C
C SETTING THE GRAPH LIMITS

C

C TOP = ( MAXY - MINY ) * .1 + MAXY
C BOTTOM = MINY - ( MAXY - MINY ) * .1
C
C CALL GRAPHV ( AXIS, DATA, YEXP, XEXP, MINX, MAXX, BOTTOM, TOP, NPTS, ' ', DEVICE )
C
C RETURN
C
C END OF FILE

C TITLE : GRAPHV
C
C DESCRIPTION :
C
C THIS SUBROUTINE PRINTS A CHARACTER GRAPH ON A STANDARD
C 24 LINE VDU SCREEN. THE USER HAS THE CHOICE OF SPECIFYING
C THE CHARACTERS WHICH WILL MAKE UP THE PRINTED CURVE OR ALLOWING
C THE PROGRAM TO IMPLEMENT A FORM OF BEST CHARACTER FIT USING
C THE CALL LINE IS :
C
C CALL GRAPHV( X, Y, YEXP, XEXP, XMIN, XMAX, YMIN, YMAX, NPTS, CHAR )
C
C PARAMETERS :
C
C X - A REAL ARRAY OF DIMENSION 'NPTS' CONTAINING THE X-
C COORDINATES OF ALL THE POINTS TO BE PLOTTED.
C ARRAY DIMENSION SHOULD BE < NPTS-1 >.
C
C Y - A REAL ARRAY OF DIMENSION 'NPTS' CONTAINING THE Y-
C COORDINATES OF ALL THE POINTS TO BE PLOTTED.
C ARRAY DIMENSION SHOULD BE < NPTS-1 >.
C
C YEXP - A STRING OF 12 CHARACTERS IN LENGTH WHICH WILL
C BE PRINTED AS THE Y-AXIS TITLE.
C
C XEXP - A STRING OF 72 CHARACTERS IN LENGTH WHICH WILL
C BE PRINTED AS THE X-AXIS TITLE.
C
C XMIN - A REAL VALUE, THE SMALLEST VALUE OF X TO BE
C PLOTTED.
C
C XMAX - A REAL VALUE, THE LARGEST VALUE OF X TO BE
C PLOTTED.
C
C YMIN - A REAL VALUE, THE SMALLEST VALUE OF Y TO BE
C PLOTTED.
C
C YMAX - A REAL VALUE, THE LARGEST VALUE OF Y TO BE
C PLOTTED.
C
C NPTS - AN INTEGER, THE TOTAL NUMBER OF POINTS TO BE
C PLOTTED.
C
C CHAR - A CHARACTER STRING, ONE CHARACTER IN LENGTH.
C THIS INDICATES TO 'GRAPHV' WHICH TYPE OF CHARACTERS ARE TO BE USED TO PRINT THE CURVE.
C IF:
C
C CHAR = "any character" - THE SPECIFIED
C CHARACTER IS
C USER.
C
C CHAR = "blank" - EITHER A " " WILL BE
C USED TO ACHIEVE A "BEST
C FIT.
C
C DEVICE - AN INTEGER SPECIFYING THE DEVICES IN WHICH THE
C PRINTOUT IS TO BE DISPLAYED (FORAM SPECIES).
SUBROUTINE GRAPH ( X, Y, YEXP, XEXP, XMIN, XMAX, 
& YMIN, YMAX, NPTS, CHAR, DEVICE )

DEFINING THE VARIABLES TO BE USED.
*************************************************************************

IMPLICIT COMPLEX (A - Z)

PARAMETER XSTEP = 14, YSTEP = 5
PARAMETER WIDTH = 4 * XSTEP + 1, LENGTH = 4 * YSTEP + 1

INTEGER I, J, NPTS
INTEGER XCOORD, YCOORD, TEMP, DEVICE

REAL X (0:NPTS-1), Y (0:NPTS-1)
REAL XMIN, XMAX, YMIN, YMAX
REAL XLABEL (5)

CHARACTER * 1 GRAPH ( LENGTH, WIDTH )
CHARACTER * 12 XLABEL ( LENGTH )
CHARACTER YEXP * 12, XEXP * 72
CHARACTER * 1 CHAR, PCCHAR
CHARACTER * 1 APPROX ( 3 )

IN THE ABOVE DEFINITION BLOCK THE INPUT PARAMETERS
ARE DEFINED ALONG WITH SEVERAL INTERNAL VARIABLES
AND INTERNAL PARAMETERS.

LOCAL VARIABLES AND PARAMETERS : 

XSTEP - THE NUMBER OF PRINT CHARACTERS BETWEEN
THE X DISTANCE MARKERS.

YSTEP - THE NUMBER OF PRINT CHARACTERS BETWEEN
THE Y DISTANCE MARKERS.

WIDTH - GRAPH WIDTH AS DEFINED IN TERMS OF XSTEP

LENGTH - GRAPH LENGTH DEFINED IN TERMS OF YSTEP

I, J - DO LOOP RUNNING VARIABLES
XCOORD, YCOORD - ARRAY INDICES FOR THE CARTIL
ESIAN TYPE COORDINATES OF THE

TEMP - TEMPORARY INTEGER VALUE

XLABEL - THE FIVE DISTANCE MARKERS TO BE PRINTED
FOR THE X - AXIS.

CHARACTERS :

GRAPH - AN ARRAY IN WHICH THE CURVE AND ITS AXES
IS GENERATED

XLABEL - AN ARRAY WHICH CONTAINS THE Y AXIS MARKERS AND LABEL

PCCHAR - THE ACTUAL CHARACTER REPRESENTING THE
CURVE TO BE PRINTED.

APPROX - A 3-DIMENSIONAL ARRAY CONTAINING , *
C INITIALIZING THE VARIABLES TO BE USED

C******************************************************************************
C
C APPROX ( 1 ) = '_'
C APPROX ( 2 ) = ' -'
C APPROX ( 3 ) = ' -'

C DO 100 I = 1, LENGTH
C YLABEL ( I ) = 12H
C DO 200 J = 1, WIDTH
C GRAPH ( I, J ) = ' -'

200 CONTINUE
C 100 CONTINUE
C C C
C DRAWING THE AXES ONTO THE GRAPH

C******************************************************************************
C
C DO 300 I = 1, WIDTH
C GRAPH ( I, 1 ) = ' -'
C GRAPH ( LENGTH, I ) = ' -'
C 300 CONTINUE
C
C DO 400 I = 1, LENGTH
C GRAPH ( I, 1 ) = ' -'
C GRAPH ( I, WIDTH ) = ' -'
C 400 CONTINUE
C
C DO 500 I = 1, LENGTH, YSTEP
C DO 600 J = 1, WIDTH, XSTEP
C GRAPH ( I, J ) = ' +'
C 600 CONTINUE
C C C
C SETTING UP AXIS LABELS AND MARKERS

C******************************************************************************
C
C YLABEL ( LENGTH / 2 - 2 ) = YEXT
C DO 700 I = 1,5
C YLABEL ( I ) = XMNT + ( XMAX - XMNT ) * ( I - 1 ) / 4
C ENCODE ( 1000, YLABEL ( YSTEP * ( I - 1 ) + 1 ) )
C YLABEL = YMAX + ( YMAX - YMIN ) * ( I - 1 ) / 4
C 700 CONTINUE
C
C C C
C ENTERING THE CURVE CHARACTERS INTO THE GRAPH ARRAY

C******************************************************************************
C
C DO 800 I = 0,NPTS-1
C PCHAR = CHAR
XCOORD = INT ( WIDTH * ( X(I) - XMNT ) / ( XMAX - XMNT ) ) + 1
C TESTING TO DETERMINE IF THE X COORDINATE IS WITHIN
C THE MAXIMUM AND MINIMUM VALUES. IF NOT IT IS SET
C EQUAL TO THE CORRESPONDING LIMIT.
C IF ( XCOORD .GT. 1 ) XCOORD = WIDTH
C IF ( XCOORD .LT. 1 ) XCOORD = 1
C TESTING TO DETERMINE IF THE Y COORDINATE IS WITHIN
C THE MAXIMUM AND MINIMUM VALUES. IF NOT IT IS SET
C EQUAL TO THE CORRESPONDING LIMIT.
C YTEMP = INT ( 3 * LENGTH * ( Y(I) - YMIN ) / ( YMAX - YMIN ) )
C IF ( YTEMP .GT. 3 * LENGTH - 1 ) YTEMP = 3 * LENGTH - 1
C IF ( YTEMP .LT. 0 ) YTEMP = 0
TESTING IF THE "BEST FIT" MUST BE APPLIED AND IF SO
PRINTING THE CHARACTER WHICH APPROXIMATES THE CURVE

IF ( CHAR .EQ. ' ' ) CHAR = APPROX ( MOD ( YTEMP, 3 ) + 1 )

YCOORD = YTEMP / 3 + 1

ACTUALLY PLACING THE CHARACTER ON THE GRAPH

GRAPH ( YCOORD, XCOORD ) = CHAR

CONTINUE

PRINTING THE GRAPH ON THE VDU SCREEN

=========================================

WRITE ( DEVICE, 5000 )

DO 900 I = 1, LENGTH
    WRITE ( DEVICE, 2000 ) XLABEL ( I ), ( GRAPH ( LENGTH+1-I, J),
    J = 1, WIDTH )
900 CONTINUE

WRITE ( DEVICE, 3000 ) XLABEL
WRITE ( DEVICE, 4000 ) YLABEL
RETURN

FORMAT STATEMENTS

1000 FORMAT ( 4X, F8.2 )
2000 FORMAT ( 2X, A12, 2X, 20A1 )
3000 FORMAT ( 2X, 5&4X, F8.2i )
4000 FORMAT ( A2 )
5000 FORMAT ( 1X )

END
END OF FILE

TITLE : PRINT VI.0

DESCRIPTION:

THIS SUBROUTINE PRINTS OUT NPTS OF DATA ALONG WITH
THE INDEX FOR AT ROW. THE DATA IS PRESENTED BELOW
ITS RESPECTIVE INDEX AT 8 PER LINE.

PARAMETERS :

NPTS - NUMBER OF DATA POINTS IN ARRAY,
MUST BE A POWER OF TWO.
DATA - ARRAY CONTAINING THE RELEVANT DATA.
TITLE - A 12 CHARACTER WHICH IS USED AS A HEADING
DEVICE - AN INTEGER SPECIFYING TO WHICH DEVICE PRINTOUT
IS TO BE SENT ( USUAL FORTRAN SPECIFICATIONS).

CODE:

SUBROUTINE PRINT ( DATA, NPTS, TITLE, DEVICE )
INITIALIZING THE VARIABLES TO BE USED
IMPLICIT COMPLEX ( A - Z )
INTEGER NPTS, A, B, DEVICE
REAL DATA ( NPTS-1 )
CHARACTER * 12 TITLE
C PRINTING THE DATA ROW BY ROW
C
WRITE (DEVICE, 4000) TITLE
C
DO 100 A = 0, NPTS-6, 6
C WRITE (DEVICE, 1000) ( B+A, B = 0, 7 )
WRITE (DEVICE, 2000) ( DATA (A+B), B = 0, 7 )
C 100 CONTINUE
C
C PRINTING SOME BLANK LINES
C
WRITE(DEVICE, 3000)
C
FORMAT
C 1000 FORMAT (///, "DATA INDEX: ", 3X, 8(1A, 8X))
C 2000 FORMAT (/, 7X, "DATA:", 8(F10.6, 2X))
C 3000 FORMAT (// )
C 4000 FORMAT (1HI, ///, 49X, 12, 5X, "DATA", /, 38X, 25(=) )
C
RETURN
END
END OF FILE
->

C*****************************************************************
C THIS SUBROUTINE COMPUTES THE FAST FOURIER TRANSFORM USING THE
C COOLEY-TUKEY ALGORITHM
C RADIX 2, DECIMATION IN TIME
C*****************************************************************

SUBROUTINE FFT(X,Y,M)
DIMENSION X(1:), Y(1:)
INTEGER REF, DISP
PI=3.1415927
N=2**M
C
THE FOLLOWING SECTION REGOURES THE INPUT DATA
C SO THAT THE TRANSFORM MAY BE COMPUTED IN PLACE
C
NM1=N-1
J=1
DO 30 I=1, NM1
IF (I.GE.J) GO TO 10
T1=X(I)
X(I)=T1
T2=Y(I)
Y(I)=T2
10 J(J)=I:
GO TO 20
30
C
THE FOLLOWING CODE ACTUALLY COMPUTES THE FFT
C
DO 40 J=1, M
REF=2**J
DISP=REF/2
ARG=2*PI/REF
DO 40 J=1, DISP
TWF=J**ARG
C=COS(TWF)
S=SIN(TWF)
DO 40 K=1, M, REF
J2=K+DISP
11=C*X(J2)+S*Y(J2)
12=-S*X(J2)+C*Y(J2)
X(J2)=X(J2)+11
Y(J2)=Y(J2)+12
X(K)=X(K)+11
Y(K)=Y(K)+12
40 CONTINUE
C
RETURN
END
Y(...) = B
THE TOTAL NUMBER OF POINTS MUST BE A POWER OF 2
AND IT IS THE POWER TO WHICH 2 MUST BE RAISED.
I.E. 2**N

SUBROUTINE INVFFT(X,Y,N)
DIMENSION X(N),Y(N) M=2**N DO 10 I=1,N Y(I)=Y(I)
10 CONTINUE
CALL FFT(X,Y,M)
DO 20 J=1,N X(J)=X(J)/N
20 CONTINUE
RETURN
END

DESCRIPTION:

THIS SUBROUTINE APPLIES ONE OF FIVE WINDOWS TO THE ARRAY SPECIFIED. THE INPUT AND OUTPUT ARRAYS MAY BE THE SAME ARRAY.

THE CALL LINE IS:

CALL WINDOW ( DATAIN, DATAOUT, N, L, WIND )

PARAMETERS:

DATAIN - A REAL ARRAY OF DIMENSION N CONTAINING THE INPUT DATA.
DATAOUT - A REAL ARRAY OF DIMENSION N CONTAINING THE WINDOWED DATA.
N - AN INTEGER SPECIFYING THE NUMBER OF DATA POINTS IN THE WINDOW.
L - AN INTEGER WHICH SPECIFIES THE WINDOW TO BE APPLIED TO THE DATA.

# 1 - ZERO ORDER WINDOW
# 2 - FIRST ORDER WINDOW
# 3 - HANNING WINDOW
# 4 - HAMMING WINDOW
# 5 - BLACKMAN WINDOW

CODE:

SUBROUTINE WINDOW ( DATAIN, DATAOUT, N, L, WIND )
IMPLICIT COMPLEX (A-Z)
INTEGER N, L, WIND, I
REAL DATAIN (N), DATAOUT (N)
ALAL AI, AZ

TITLE : WINDOW
DESCRIPTION:

THIS SUBROUTINE APPLIES ONE OF FIVE WINOOGS TO THE ARRAY SPECIFIED.
THE INPUT AND OUTPUT ARRAYS MAY BE THE SAME ARRAY.

THE CALL LINE IS:

CALL WINDOW ( DATAIN, DATAOUT, N, L, WIND )

PARAMETERS:

DATAIN - A REAL ARRAY OF DIMENSION N CONTAINING THE INPUT DATA.
DATAOUT - A REAL ARRAY OF DIMENSION N CONTAINING THE WINDOWED DATA.
N - AN INTEGER SPECIFYING THE NUMBER OF DATA POINTS IN THE WINDOW.
L - AN INTEGER WHICH SPECIFIES THE WINDOW TO BE APPLIED TO THE DATA.

# 1 - ZERO ORDER WINDOW
# 2 - FIRST ORDER WINDOW
# 3 - HANNING WINDOW
# 4 - HAMMING WINDOW
# 5 - BLACKMAN WINDOW
C
C SETTING THE OUTPUT DATA EQUAL TO THE INPUT DATA INITIALLY
C
DO 400 I = 1, N
   DATOUT (I) = DATAIN (I)
400 CONTINUE
C
C IMPLEMENTING THE DESIRED WINDOW
C
WIND = MAXD ( 1, MIND ( 5, WIND ) )
GO TO ( 10, 20, 30, 40, 50 ) WIND
C
C ZERO ORDER WINDOW
C
IF ( L .GE. N ) RETURN
DO 100 I = L + 1, N
   DATOUT (I) = 0.
100 CONTINUE
C
RETURN
C
C FIRST ORDER WINDOW
C
A1 = (L - 1.) / 2.
DO 200 I = 1, L
   W = 1. - ABS (I - 1.) * A1 / A1
   DATOUT (I) = DATOUT (I) + W
200 CONTINUE
C
GO TO 10
C
C HANNING WINDOW
C
A1 = 6.28318 / (L - 1.)
DO 300 I = 1, L
   W = .5 - .5 * COS ( (I-1.) * A1 )
   DATOUT (I) = DATOUT (I) + W
300 CONTINUE
C
GO TO 10
C
C HANNING WINDOW
C
A1 = 6.28318 / (L - 1.)
DO 400 I = 1, L
   W = .5 - .5 * COS ( (I-1.) * A1 )
   DATOUT (I) = DATOUT (I) + W
400 CONTINUE
C
GO TO 10
C
C BLACKMAN WINDOW
C
A1 = 6.28318 / (L - 1.)
A2 = 2. * A1
DO 500 I = 1, L
   W = .42 - .5 * COS ( (I-1.) * A1 ) + .08 * COS ( (I-1.) * A2 )
   DATOUT (I) = DATOUT (I) + W
500 CONTINUE
C
END
END OF FILE
THIS ROUTINE PLOTS INTO A NEW GPD PAGE A GRAPH OF THE X POINTS VS THE Y POINTS. IT ALSO PLACES A SCALE ALONG THE TWO AXES & WRITES THE LABELS ALONG THE CORRECT AXIS.

INPUTS

X ARRAY OF POINTS TO BE PLOTTED
Y ARRAY OF Y VALUES TO BE PLOTTED AGAINST THE X VALUES
XAXIS THE X AXIS LENGTH IN CMS
YAXIS THE Y AXIS LENGTH IN CMS
XLABEL THE X AXIS LABEL
YLABEL THE Y AXIS LABEL
NXCHAR THE NUMBER OF CHARACTERS IN THE X AXIS LABEL
NYCHAR THE NUMBER OF CHARACTERS IN THE Y AXIS LABEL
TITLE THE TITLE OF THE PLOT
NTCHAR THE NUMBER OF CHARACTERS IN THE PLOT TITLE
NTPTS THE NUMBER OF POINTS IN THE X AND Y ARRAYS
XMIN MINIMUM X VALUE
XMAX MAXIMUM X VALUE
YMIN MINIMUM Y VALUE
YMAX MAXIMUM Y VALUE

OUTPUTS

THE ROUTINE OUTPUTS TO A NEW GPD PLOT FILE THE GRAPH.

LOCAL VARIABLES

XRANG THE DIFFERENCE BETWEEN THE X VALUES AT THE ORIGIN AND AT THE END OF THE X AXIS
YRANG THE DIFFERENCE BETWEEN THE Y VALUES AT THE ORIGIN AND AT THE END OF THE Y AXIS
XSTART THE VALUE OF THE ORIGIN IN TERMS OF THE X VALUES
YSTART THE VALUES OF THE ORIGIN IN TERMS OF THE Y VALUES
XVST THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE X AXIS SCALE MARKS.
YVST THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE Y AXIS SCALE MARKS.
XSTEP THE NUMBER OF CMS BETWEEN SUCCESSIVE X AXIS SCALE MARKS
YSTEP THE NUMBER OF CMS BETWEEN SUCCESSIVE Y AXIS SCALE MARKS
XCHAR THE CALCULATED MAXIMUM HEIGHT OF X LABEL CHARACTERS
YCHAR THE CALCULATED MAXIMUM HEIGHT OF Y LABEL CHARACTERS
XCHAR THE SIZE OF THE LABEL CHARACTERS IN CMS
NXLNS THE CALCULATED NUMBER OF LINES NEEDED FOR THE X AXIS LABEL
NYLNS THE CALCULATED NUMBER OF LINES NEEDED FOR THE Y AXIS LABEL
NTLNS THE CALCULATED NUMBER OF LINES NEEDED FOR THE PLOT TITLE.
REAL X(1), Y(1)
CHARACTER*4 XLABEL(1), YLABEL(1), TITLE(1)

SET UP THE SCALE VALUES FOR THE LABELING

----- GET THE RANGE OF VALUES ----- 
XRANG = XMAX - XMIN
YRANG = YMAX - YMIN

----- GET FIRST VALUES ----- 
XSTART = XMIN
YSTART = YMIN

XLAST = XMAX
YLAST = YMAX

----- WORK OUT HOW MANY TICKS TO HAVE ----- 
NXTICS = IFIX(XAXIS/4.0 + 0.5) + 1
NYTICS = IFIX(YAXIS/4.0 + 0.5) + 1

----- GET THE DISTANCE BETWEEN TWO SCALE MARKS ----- 
XSTEP = XAXIS/(NXTICS-1)
YSTEP = YAXIS/(NYTICS-1)

----- GET THE CHANGE OF VALUE BETWEEN TWO SCALE MARKS ----- 
XXST = (XLAST-XSTART)/FLOAT(NXTICS-1)
YYST = (YLAST-YSTART)/FLOAT(NYTICS-1)

----- GET THE FACTORS TO CONVERT THE X & Y POINTS ----- 
XFAC = XAXIS/XRANG
YFAC = YAXIS/YRANG

----- WORK OUT THE PAGE SIZE NEEDED FOR THE PLOT ----- 

----- WHAT SPACE FOR THE LABELS ----- 

IF(NXCHAR .GT. 80) THEN
XCHARH = XAXIS / 80.0
ELSE
XCHARH = 0.5
ENDIF

IF(NYCHAR .GT. 80) THEN
YCHARH = YAXIS / 80.0
ELSE
YCHARH = 0.5
ENDIF

----- SET THE NUMBER OF LINES OF EACH AXIS LABEL ----- 

NXLNS = 1
NLXNS = 1
NLXNS = 1

----- YOU CAN GET HEIGHT OF AXIS LABELS ----- 
CHARH = AMNCH(XCHARH,YCHARH,TCH,1RHI)

----- YOU GET THE PAGE DEFINITION ----- 
CALL PAGEDEF(-1.5+NYLNS*1.5*XCHARH, -1.5+(NXLNS+NLNS)*1.5*XCHARH),
+ (1.0+YAXIS), (1.0+YAXIS) )
SUBROUTINE AXES(NXTICS, NYTICS, XSTEP, YSTEP, XSTART, YSTART, XVST, YVST)

THIS ROUTINE IS DESIGNED TO DRAW TO THE CURRENTLY ACTIVE
PLOT PAGE THE AXES & THE SCALE VALUES.

**INPUTS**

NXTICS  NUMBER OF TICS ON THE X AXIS
NYTICS  NUMBER OF TICS ON THE Y AXIS
XSTEP
YSTEP
XSTART
YSTART
XVST
YVST

**OUTPUT**

THIS ROUTINE WRITES TO THE CURRENTLY ACTIVE PLOT FILE THE CODE
NECESSARY TO PRODUCE THE PLOT OF THE AXES & SCALES

**DECLARATIONS**

CHARACTER*4 YMAT(3) '//.,","",","",",",', CHAR(2)
DATA CHROM, 10.35/

---- SET UP THE SPACE SIZE OF THE TICMARK TO NUMBER GAP ----
SPACE = CHROM

---- SET UP THE TIC SIZE ----
TICSIZ = 0.25*CHAR

---- INITIALISE FOR X ----
X = 0.0
Y = XSTART
CALL FMTI1(XSTEP,fmt)
DO 10 I = 1, NXTICS
ENCOD(4,fmt,CHAR) V
CALL DIEXT(X=+5.0*CHAR), (-SPACE+TICSIZ+CHAR), CHAR, CHAR, 0.0, 0.0)

---- DRAW THE TIC MARK ----
CALL DRAWEX(), -TICSIZ, X, 0.0)

---- SET UP FOR THE NEXT SHOT ----
X = X + XSTEP
Y = Y + YSTEP

CONTINUE
--- DRAW THE AXIS LINE ---

CALL DRAWSEG(X, YSTEP, 0.0, 0.0, 0.0)

--- HOW THE Y AXIS ---

--- INITIALISE ---

Y = 0.0
V = YSTART
CALL FRMTITI(YSTEP, FRMT)

DO 20 I = 1, NTTICS
   ENCODE(S, FRMT, CHAR) V
   CALL DTEXT(S, SPACE + TICSIZ), (Y - YSTEF'), CHAR, CHAR, CHAR + .0, 0.

20 CONTINUE

--- DRAW THE TIC MARKS ---

CALL DRAWSEG(-TICSIZ, Y, 0.0, Y)

--- SET UP FOR THE NEXT SHOT ---

Y = Y + YSTEP
V = V + YSTEF

THAT'S IT

RETURN
END

**************************************************************************

SUBROUTINE FRMTITI(STEP, FRMT)

THIS ROUTINE IS DESIGNED TO GENERATE THE FORMAT CODE TO CONVERT THE VALUES SEPARATED BY STEP INTO CHARACTERS.

INPUT

STEP --- THE CHANGE IN VALUE BETWEEN TWO SUCCESSIVE VALUES TO BE ENCODED

OUTPUT

FRMT --- THE CHARACTER ARRAY GIVING THE FORMAT

DEclarations

************

CHARACTER*4 FRMT(1)

IF (STEP .GE. 10000.00 .OR. STEP .LT. -1000.00) THEN
   --- NUMBER WILL HAVE TO BE IN E FORMAT ---
   FRMT(2) = 'EB.3'

ELSE IF (ABS(STEP) .GT. 1.00) THEN
   --- NUMBER MAY BE IN F FORMAT ---
   FRMT(2) = 'FB.2'

ELSE IF (ABS(STEP) .GT. 0.001) THEN
   --- WILL BE FORCED TO USE E FORMAT ---
   FRMT(2) = 'EN.3'

ELSE
   --- CAN USE MODIFIED F FORMAT ---
   FRMT(2) = 'FB.5'

ENDIF IF

RETURN
END
SUBROUTINE LABEL(XLABEL, YLABEL, TITLE, IXCHAR, NYCHAR, NTCHAR, CHARH)

    THIS ROUTINE WRITES THE X & Y LABELS AND TITLE

    INPUTS

    XLABEL    THE CHARACTER ARRAY CONTAINING THE X LABEL
    YLABEL    THE CHARACTER ARRAY CONTAINING THE Y LABEL
    TITLE     THE CHARACTER ARRAY CONTAINING THE PLOT TITLE
    IXCHAR    THE NUMBER OF CHARACTERS IN THE X LABEL
    NYCHAR    THE NUMBER OF CHARACTERS IN THE Y LABEL
    NTCHAR    THE NUMBER OF CHARACTERS IN THE PLOT TITLE
    CHARH     THE HEIGHT TO DRAW THE CHARACTERS

    OUTPUT

    TO THE CURRENTLY ACTIVE GDP PLOT PAGE THE LABELS ARE DRAWN

    DECLARATIONS

    CHARACTER*4 XLABEL(1), YLABEL(1), TITLE(1)

    DO THE X AXIS LABEL

        Y = -1.2 + CHARH
        CALL DTEXT(0.0, Y, CHARH, XLABEL, 0.0, IXCHAR)

    NOW THE TITLE

        Y = -1.2 + 2.5*CHARH
        CALL DTEXT(0.0, Y, CHARH, TITLE, 0.0, NTCHAR)

    AND THE Y AXIS LABEL

        X = -1.2
        CALL DTEXT(X, 0.0, CHARH, YLABEL, 0.0, NYCHAR)

    RETURN

END

SUBROUTINE CURVE(X, Y, XSTART, YSTART, XFAC, YFAC, NPOINTS)

    THIS ROUTINE DRAWS A LINE GRAPH OF THE X POINTS VS THE Y POINTS

    INPUTS

    X        ARRAY OF X POINTS
    Y        ARRAY OF Y POINTS
    XSTART   STARTING VALUE OF X ARRAY
    YSTART   STARTING VALUE OF Y ARRAY
    XFAC     FACTOR TO CONVERT ABSOLUTE X VALUES TO CMS
    YFAC     FACTOR TO CONVERT ABSOLUTE Y VALUES TO CMS
    NPOINTS  THE NUMBER OF POINTS IN THE X & Y ARRAYS

    TO BE PLOTTED
DECLARATIONS

REAL X(1), Y(1)

GET THE SHOW ON THE ROAD
CALL MOVE((X(1)-XSTART)*XFAC,(Y(1)-YSTART)*YFAC)
DO 10 I = 2,NPNTS
CALL DRAW((X(I)-XSTART)*XFAC,(Y(I)-YSTART)*YFAC)
10 CONTINUE
RETURN
END
END OF FILE
A PROOF ON THE CONVOLUTION RESULTS USED

The proof given here shows that when relating the input $x(t)$ to a system with impulse response $g(t)$ and output $y(t)$ via the convolution integral the exact relation holds if one substitutes the autocorrelation function of the input for the input $x(t)$ and the crosscorrelation function of the input and output for the output $y(t)$.

Defining the autocorrelation of a function $x(t)$ as:

$$
\phi_{xx}(t) = \int_{-\infty}^{\infty} x^*(t) x(t+\tau) \, dt \quad \ldots \text{A4.1}
$$

and the crosscorrelation of two functions $x(t)$ and $y(t)$ as:

$$
\phi_{xy}(\tau) = \int_{-\infty}^{\infty} x^*(t) y(t+\tau) \, dt \quad \ldots \text{A4.2}
$$

If we then state the convolution integral relating $x(t)$, $g(t)$ and $y(t)$:

$$
y(t) = \int_{0}^{\infty} g(\theta) x(t-\theta) \, d\theta \quad \ldots \text{A4.3}
$$

Now replacing $x(t)$ with its autocorrelation function $\phi_{xx}(\tau)$ and $y(t)$ by some unknown function I, we may then write:
\[ I = \int \int g(\theta) \cdot \phi_{xx}(\tau-\theta) d\theta \]

Now substituting for \( \phi_{xx}(\tau-\theta) \) from equation A4.1 we have:

\[ I = \int \int g(\theta) \, x^*(t) . x(t+\tau-\theta) dt \, d\theta \]

rearranging the order of integration

\[ I = \int \int g(\theta) \, x^*(t) . x(t+\tau-\theta) dt \, d\theta \]
\[ \quad = \int x^*(t) \int g(\theta) . x(t+\tau-\theta) d\theta \, dt \quad \ldots \ A4.5 \]

and since we recognise that:

\[ \int g(\theta)x(t+\tau-\theta) d\theta = g(t) . x(t+\tau) = y(t+\tau) \quad \text{by A4.3} \]

Substituting this into A4.5 we have that:

\[ I = \int x^*(t) y(t+\tau) dt = \phi_{xy}(\tau) \quad \text{by A4.2} \]

thus we may rewrite expression A4.3:

\[ \phi_{xy}(\tau) = \int g(\theta) . \phi_{xx}(\tau-\theta) d\theta \]
Figure A5.1a The autocorrelation function.

Figure A5.1b The crosscorrelation function.

Figure A5.1c The transit time distribution.

Figure A5.1d The velocity distribution.

Figure A5.1e The simulation program output using a flat velocity profile.

SIMULA V2.0 17/02/81

PIPE LENGTH = 30 PIPE WIDTH = 20

NUMBER OF SAMPLES = 64 AVERAGING WIDTH = 10

NUMBER OF RECORDS AVERAGED = 100

VELOCITY PROFILE USED IS:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

ACF NORMALIZING FACTOR = 7.7483 CCF NORMALIZING FACTOR = 5.1628
DECONVOLUTION OF CORRELATION FUNCTIONS OBTAINED
FROM THE SIMULATED FLOW SYSTEM

The flow simulation program (SIMULA) was run using stepped and flat velocity profiles. The autocorrelation and crosscorrelation functions thus calculated were then deconvolved in an effort to arrive back with a transit time distribution or velocity distribution which resembles the original distribution used in the simulation of these correlation functions.

The output from the simulation program, the respective correlation functions and deconvolution results are shown here for three different velocity profiles. In Figure A5.1 a flat velocity profile was used in the simulation of the flow and hence the TTD and VD are expected to comprise a single impulse corresponding to this delay. If one examines Figure A5.1c and A5.1d, these impulses are present. In Figure 5.1d however, the velocity distribution has a poor signal to noise ratio. This may be attributed to the finite averaging effects as discussed in Section 3.1.4.

The curves shown in Figure A5.2 were also obtained from a simulation with a flat velocity profile but in this case the flow velocity has been increased. The ACF is narrower than that of Figure A5.1 due to the increase in frequency.
Figure A5.2a The autocorrelation function.

Figure A5.2b The crosscorrelation function.

Figure A5.2c The transit time distribution.

Figure A5.2d The velocity distribution.

SIMULA V2.0 17/02/81

PIPE LENGTH = 50 PIPE WIDTH = 20

NUMBER OF SAMPLES = 64 AVERAGING WIDTH = 10

NUMBER OF RECORDS AVERAGED = 100

VELOCITY PROFILE USED IS:

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

ACF NORMALIZING FACTOR = 7.9484 CCF NORMALIZING FACTOR = 7.3592

Figure A5.2e The simulation program output using a flat velocity profile.
of the flow signals arising from the more rapid flow. The resulting TTD and VD are shown in Figures A5.2c and A5.2d. In Figure A5.3 the deconvolution of correlation function simulated using a stepped velocity profile are given.
Figure A5.3a The autocorrelation function.

Figure A5.3b The crosscorrelation function.

Figure A5.3c The transit time distribution.

Figure A5.3d The velocity distribution.

Figure A5.3e The simulation program output using a stepped velocity profile.
Figure A6.1a Rectangular input function.

Figure A6.1b Rectangular system impulse response.

Figure A6.1c Triangular system output.
FOURIER TRANSFORM DECONVOLUTION IN THE PRESENCE OF NOISE

In an effort to determine the sensitivity of the Fourier transform deconvolution method to noise, several tests were carried out. Two well-known functions were convoluted and this result stored. Noise signals of different amplitudes are then added to this convolution result and the deconvolution of the original function and this noisy function carried out. Results of the deconvolution of triangular and rectangular pulses are shown here as well as those obtained using "test" correlation functions. These "test" correlation functions were computed using the autocorrelation function measured from the flow rig. The crosscorrelation function was found by computing a transit time distribution resulting from a typical parabolic velocity profile and convolving this TTD with the ACF.

The rectangular functions used in the first tests are shown in Figures A6.1a and A6.1b. The convolution of these two pulses is given in Figure A6.1c. Various levels of noise are added to this triangular waveform and the deconvolution of the rectangular function (in Figure A6.1a) and this noisy triangular function carried out. The expected result is the waveform given in Figure A6.1b.
Figure A6.2a System output + noise.

Figure A6.2b Deconvolved impulse response without windowing.

Figure A6.2c Deconvolved impulse response using Hamming windowing.
In Figures A6.2a and A6.3a the noise signals are added to the triangular function. The deconvolution results shown in Figures A6.2b and A6.3b have been computed without applying any windowing to the intermediate deconvolution results. The results shown in Figures A6.2c and A6.3c have been calculated by applying Hamming windows to the Fourier transform of the autocorrelation and crosscorrelation functions before and after dividing these Fourier transforms.

The application of the data window definitely reduces the noise present in the deconvolved function. The results demonstrate the sensitivity of the deconvolution result to noise added to the waveform to be deconvolved.

A second test was carried out where the functions represent more closely typical autocorrelation and crosscorrelation functions. Examples are given in Figures A6.4a and A6.4b. The TTD used to compute the CCF is given in Figure A6.4c. The deconvolution is thus expected to return a TTD similar to that shown in Figure A6.4c.

With no added noise, the deconvolution returns the expected TTD. Extremely low noise signals which were added to the CCF are shown in Figures A6.5b, A6.6b and A6.7b for comparison with the original CCFs. The correlation functions with the noise signals added, are shown in Figures A6.5a and A6.6a and A6.7a. Hamming windowing was
Figure A6.3a System output + noise.

Figure A6.3b Deconvolved impulse response without windowing.

Figure A6.3c Deconvolved impulse response using Hamming windowing.
applied to the intermediate deconvolution results. The TTDs obtained by deconvolving these correlation functions are shown in Figures A6.5c, A6.6c and A6.7c.

These results show the typical correlation functions to be much more sensitive than the rectangular functions to added noise when deconvolving.
Figure A6.4a The ACF measured from the flow rig.

Figure A6.4b The CCF obtained from the convolution of the ACF and the TTD.

Figure A6.4c The TTD calculated from a parabolic velocity profile.
Figure A6.5a  CCF + noise (.001)

Figure A6.5b  The noise added to the CCF

Figure A6.5c  The TTD obtained from the deconvolution of the ACF and the CCF + noise (.001)
Figure A6.6a CCF + noise (.0001)

Figure A6.6b The noise added to the CCF

Figure A6.6c The TTD obtained from the deconvolution of the ACF and the CCF + noise (.0001)
Figure A6.7a The CCF + noise (.00001)

Figure A6.7b The noise added to the CCF

Figure A6.7c The TTD obtained from the deconvolution of the ACF and the CCF + noise (.00001)