# A NEW ULTIMATE LIMIT STATE APPROACH TO THE DESIGN OF PRESTRESSED CONCRETE BEAMS

BY

GRANT ROBERT HALLAM

A thesis submitted in partial fulfilment of the requirements for the Degree of Master of Science in the Faculty of Engineering, University of Cape Town

Department of Civil Engineering
UNIVERSITY OF CAPE TOWN

January 1990

## DECLARATION OF CANDIDATE

I, Grant Robert Hallam, hereby declare that this thesis is my own work and that it has not been submitted for a degree at another university.

Signed by candidate

Signature Removed

January 1990

# TABLE OF CONTENTS


TERMS OF REFERENCE

SYNOPSIS

SYMBOLS USED

CHAPTER 1 INTRODUCTION 1-01

CHAPTER 2 THE DESIGN APPROACH TO PRESTRESSED CONCRETE BEAMS 2-02

2.1 Present Design Approach 2-01
2.1.1 A Brief History 2-01
2.1.2 Details of the Present Design Approach 2-03
2.1.3 Load-balancing Method 2-09

2.2 Proposed Approach to Prestress Design 2-11

CHAPTER 3 GENERAL THEORY FOR BEAM DEFORMATIONS 3-01

3.1 Curvature 3-01
3.1.1 Curvature in the Elastic Zone 3-01
3.1.2 Determination of the Cracking Moment and Curvature 3-02
3.1.3 Curvature After Cracking 3-03
3.1.4 Correction of Tension Stiffening 3-05

3.2 Calculation of Deformation by Integration Principles 3-09
3.2.1 Statically Determinate Beams 3-10
3.2.2 Statically Indeterminate Beams 3-10

CHAPTER 4 DETAILED PROCEDURE TO DETERMINE DEFORMATIONS 4-01

4.1 Input of Data 4-02
4.1.1 Material Properties 4-02
4.1.2 Element Cross-section 4-02
4.1.3 Other Beam Data 4-03

4.2 Determination of Curvature at a Cross-section 4-04
4.2.1 Determination of Cracking Moment 4-04
4.2.2 Determination of Moment and Curvature 4-05
4.2.3 Correction of Tension Stiffening 4-07

4.3 Determination of Deformations 4-07
4.3.1 Cantilever Beams 4-07
4.3.2 Simply Supported Beam 4-09
4.3.3 Fully Fixed Beam 4-09

CHAPTER 5 DISCUSSION OF RESULTS 5-01

5.1 Experimental Results 5-01
5.1.1 Results of Beams B1 and B2 5-01
5.1.2 Results of Beams C1 and C2 5-02
5.1.3 Results of D1 5-02
5.1.4 Discussion of Results 5-03

5.2 The Effects of Tension Stiffening 5-03
5.2.1 Discussion of Example 5-03
5.2.2 The Effect on Moment Redistribution 5-04
5.2.3 Effect on the Investigation 5-04

5.3 Discussion of the Investigation 5-04
5.3.1 Results for T-Beam 5-05
5.3.2 Effect of Reinforcing Steel 5-06
5.3.3 Results for I-Beam 5-06
5.3.4 Discussion of Results 5-07
5.3.5 Equation to Determine Redistribution 5-07
5.3.5 Use of Design Material Factors 5-08

CHAPTER 6 CONCLUSION 6-01

REFERENCES

ACKNOWLEDGEMENTS

APPENDIX A MATHEMATICAL MODELS USED IN PROGRAM A-01

A.1 Simpson's Rule A-01
A.2 Newton-Raphson Method A-02

APPENDIX B PROGRAM LISTING B-01

APPENDIX C RESULTS OF TESTING C-01

APPENDIX D EXAMPLE FOR TENSION STIFFENING D-01

APPENDIX E FULLY FIXED T-BEAM EXAMPLE E-01

APPENDIX F EXAMPLE FOR A FULLY FIXED I-BEAM F-01

# TERMS OF REFERENCE

This half-thesis was required by the Department of Civil Engineering, UCT, as partial fulfilment of the requirements for the Degree of Master of Science, Civil Engineering.

Under the supervision of Associate Professor R,D, Kratz, an uncomplicated design approach for prestressed concrete structures had to be devised which would enable the prestress profiles and magnitude of prestress to be determined by considering the Ultimate Limit State only, ie. disregarding allowable stress limits in the design approach. The Serviceability Limit State considerations, ie. state of cracking and deflection limits, would only be used to check the design.

Particular attention had to be paid to the physical design parameters such as ductility and bending moment redistribution limits as part of the design procedure.

A rigorous plastic analysis computer program had to be developed in order to study prestressed beam behaviour and verify certain design parameters.

# SYNOPSIS

The present approach to the design of prestressed beams is antiquated and time consuming. Neither SLS or ULS requirements are satisfied directly. There is a need for a new approach using plastic principles to design the prestressing requirements at ULS considering a whole span at a time, with checks made for SLS requirements afterwards.

For a plastic design, the designer would need to know the limits of the bending moment redistribution for the beam under consideration. An equation is therefore necessary to assist the designer in this regard. Such an equation should take into account the cross-section shape of the beam and the prestress to reinforcing steel ratios. Many examples would have to be investigated using a rigorous plastic analysis to formulate such an equation.

A computer program has been written as part of this thesis to perform such a rigorous analysis. It's accuracy has been evaluated by comparison with laboratory test beam results. The comparison was favourable, although more results would have to be compared to establish the accuracy that could be expected.

A few examples were investigated and the observations noted. Two extreme cases were examined, those of a T-beam and I-beam. The redistribution of bending moments was similar in both cases, but the percentage of the central moment capacity reached was considerably less for the T-beam. For both beams it was impossible to reach the ultimate capacity at the centre before the strain capacity at the supports was exhausted.

It was concluded that a plastic design procedure should be introduced, based on the ULS capacity of a span at a time. SLS requirements could then be checked afterwards. This approach would provide a design procedure corresponding to those used for other forms of modern structural design and would be much quicker, safer, more accurate and less costly that the existing design procedures.

## SYMBOLS USED

$A_C$ Area of concrete in compression

$A_{PS}$ Area of prestressing steel

$A_R$ Area of reinforcing steel

b Breadth of a concrete cross-section

$C_C$ Compressive force in the concrete

d Effect of depth of the tensile steel

E Modulus of Elasticity

$e_p$ Eccentricity of the line of pressure

$e_s$ Eccentricity of the tendon profile

$f_{a\ min}$ is the minimum allowable stress in the concrete in compliance with the class of the member and the grade of concrete

$f_{a\ max}$ is the maximum allowable stress in the concrete in compliance with the class of the member and the grade of concrete

$F_C$ Total force in the concrete

$F_S$ Force in the steel

$f_{ctm}$ Modulus of rupture of concrete

$f_{pi}$ Strain in the tendon due to prestressing

h Depth of cross-section

I Second moment of area of cross-section

$k_1$ Coefficient of effect of section shape on redistribution

$k_2$ Coefficient for effect of ratio of prestress to reinforcing

L Length of span

$\ell$ Length of that part of a span under consideration

M Bending moment applied to the beam

$M_{cr}$ Cracking moment for a cross-section

$M_d$ Bending moment due to dead load

$M_{i\ min}/M_{i\ max}$ Mimimum and maximum bending moments which could act on a cross-section

$M_{ps}$ Bending moment on a cross-section due to prestress eccentricity at that section

P Prestress force at transfer

$P_e$ Effective Prestress force (ie prestress force after all losses)

r Radius of curvature

red Redistribution of bending moments

$T_C$ Tensile force in concrete

$T_s$ Tensile force in steel

$x$ Depth of the concrete in compression

$z$ Elastic section modulus

$\delta$ Deflection at a point

$\epsilon_c$ Strain in the extreme concrete compression fibre

$\epsilon_{ci}$ Strain in the concrete due to the prestress force

$\epsilon_{ct}$ Strain at which the concrete begins to crack

$\epsilon_{c1}$ Strain in the concrete calculated for an uncracked section

$\epsilon_{c2}$ Strain in the concrete calculated for a cracked section

$\epsilon_s$ Strain in the steel

$\epsilon_{s1}$ Strain in the steel calculated for an uncracked section

$\epsilon_{s2}$ Strain in the steel calculated for a cracked section

$\phi$ Curvature at a cross-section

$\phi_{cr}$ Cracking curvature for a cross-section

$\sigma_c$ Stress in the extreme concrete compression fibre

$\sigma_{pi}$ Compressive stress in the concrete due to the prestress force

$\sigma_{sr}$ Stress in a steel at cracking of extreme concrete fibre calculated on the basis of a cracked section

$\sigma_{s2}$ Stress in the reinforcement at a cracked section under the combination of actions considered

$\theta$ Rotation of an element

1,2 Used as subscripts, refer to the bottom and top of a beam cross-section respectively in serviceability calculations

# CHAPTER 1 INTRODUCTION

The method of design for prestressed concrete structures has changed little over the past 40 to 50 years. Although it is well known and documented, the approach is antiquated in light of recent developments in the philosophy of structural analysis and design. Although this approach is systematic, it is extremely time consuming and satisfies neither the Ultimate Limit State (ULS) or Servicability Limit State (SLS) requirements directly. It is based on an allowable stress approach at working load level, with many equations to be satisfied at critical sections. Much of this work is totally unnecessary. ULS and SLS requirements are checked at critical sections after all the allowable stress equations have been met.

The prestress requirements could instead be found using a plastic design procedure to ensure the ULS capacity of the structure on the basis of a span at a time. The SLS requirements could subsequently be checked for cracking and deflections in a manner corresponding to similar SLS checks presently required for reinforced concrete members, structural steel sections and composite sections. This proposed method results in a much quicker and more accurate design procedure. Chapter 2 describes the existing design procedure and then puts forward the proposed approach.

In order to evaluate the amount of prestressed (and reinforcing) steel along a span of an indeterminate structure it is essential to know the maximum amount of redistribution which can take place in that span for a given type of beam cross-section. A distribution equation, taking into account the applicable beam parameters, could be used to guide the designer.

In order to use a plastic procedure for the design of a prestressed beam, it would be necessary to know this extent of redistribution of bending moments that could be allowed for the structure under consideration. A computer program was thus developed as part of this thesis to investigate the behaviour of prestressed concrete beams when they become plastic. The program

uses a rigorous plastic analysis to determine the amount of redistribution within the beam. The theory used in the development of the program is discussed in Chapter 3, while the actual development of the program itself and the application of the theory is discussed in Chapter 4.

The accuracy of the program was assessed by comparing results obtained from a few laboratory experiments, conducted as part of a B.Sc thesis under supervision of the author, with results obtained from similar beams using the computer program. These results are given in Chapter 5, along with additional theoretical examples using the program. These demonstrate the usefulness of a rigorous analytical approach and give an idea of the results that can be expected for various beams.

Further examples and laboratory verification should be done in order to gain a more accurate idea of beam behaviour under plastic conditions so that an equation can be developed to guide the designer with respect to the amount of redistribution that can be safely allowed. This equation would have to include the effects of the concrete cross-section shape and prestress to reinforced steel ratios expected.

# CHAPTER 2 THE DESIGN APPROACH TO PRESTRESSED CONCRETE BEAMS.

The approach to the design of prestressed concrete beams is extremely well documented and used in most codes of practice world wide. As opposed to reinforced concrete beams, the design of prestressed concrete beams is generally governed by the stress criteria in service or at transfer, rather than by their ultimate strengths. The ultimate strengths of critical sections are checked after the allowable stress equations have been satisfied. The elastic theory is thus very relevant in prestressed concrete design.

This approach is extremely time consuming, especially for continuous beams with a high degree of indeterminacy. It also satisfies neither the Ultimate Limit State nor the Serviceability Limit State requirements directly. As a result, a new approach is proposed based on the ultimate design of a beam system span by span, with serviceability limit requirements checked afterwards. This is also the trend in other modern design procedures (eg. reinforced concrete, structural steel etc.). This chapter will first discuss the existing method and will then put forward the proposed design approach.



## 2.1 Present Design Approach

### 2.1.1 A Brief History

Reinforced concrete came into widespread use as a structural material just before the start of the present century. It was during the 1940's that the idea of prestressed concrete was developed and put to practical use. It provided a way of loading the structure internally to counteract a portion of the externally applied loads and dead load by stressing with high strength steel, thereby placing the concrete in compression. The advantages were that the entire cross-section became effective in resisting the applied moment, resulting in greatly reduced deflections compared with reinforced concrete. Part of the shear

could be carried by the curved tendons. The diagonal tension was reduced, and thus much lighter sections could be used to carry the same applied load. Much longer spans could also be used. Also, as most of the concrete was in compression, cracks were absent during service type loading.

Prestressing led to the activation of the whole cross-section to carry the loads, and the stress distribution over the section due to both bending and shear were well known. Prestressed concrete structures were thus designed for the SLS with permissible stresses in the concrete and steel. As cracking was avoided, the role of reinforcement steel was greatly reduced, and was almost non existent in many structures.

The importance of safety against failure in the design of structures was soon realised and lead to the Comité Euro-International du Béton's (C.E.B) approach to reinforced concrete design on the basis of ultimate limit state with checks for serviceability limit state as introduced in the 1950's. Prestressed concrete structural design, however, was still based on a serviceability limit state approach, with checks for ultimate limit state.



Although prestressing was a method in which most of the concrete was under compression, such that cracks did not occur under normal loading, some specialists felt that some cracking should be allowed. If a crack appeared under a temporary load, it would close if the load decreased. Such a beam would be more economical than a fully prestressed concrete beam. This was termed partial prestressing, and could be achieved by reducing the steel stresses or by using a number of unstressed wires in pretensioned prestressed concrete in order to limit the initial compressive stresses in the precompressed tensile zone, without reducing the failure load of the structure.

In 1956, Swiss engineers introduced this method into bridge design. Under service conditions the stresses were checked on the assumption of uncracked sections and the stresses were also checked on the assumption of a reinforced concrete section with a cracked tensile zone. The use of partially prestressed

concrete has been subsequently developed by Swiss engineers and has been used in Switzerland ever since. Other countries, however, have debated the issue, and common acceptance has not yet emerged. This is largely due to the fact that there is no generally accepted approach and practising designers are unaware of the advantages. A further problem is that the design and dimensioning of partially prestressed concrete structures are relatively complicated in comparison to normal reinforced or prestressed structures due to the additional design parameters.

### 2.1.2 Details of the Present Design Approach

The present design approach, as discussed earlier, is governed by the satisfaction of allowable stress equations at critical sections. Designs are usually based on conditions in service, but the concrete stresses at transfer must be checked as well. Transfer is the operation whereby the prestressed forces from the tendons are transfered to the concrete section.

The design approach is the basis of (1) the South African code SABS 0100 Part I, and is well described and explained in books such as those by Kong and Evans (2) and Hurst (3). The design approach for continuous beams, as described by Kong and Evans, is summarized below. Members are divided into three classes for SLS. For class 1 members, no tensile stresses are permitted, while for class 2 and 3 members tensile stresses are permitted. For class 2 members these stresses must be kept sufficiently low so that no visible cracks occur. For class 3 members cracks are permitted, but must not exceed 0,1 mm for aggressive environments and 0,2 mm for normal conditions. Except where considering the ULS of collapse, the members are analysed and designed as uncracked members so that the ordinary elastic beam theory applies.

For the design of a beam, the maximum and minimum bending moments which would act at selected critical design cross-sections are determined from combinations of loads expected to act on the beam. This is based on an elastic analysis. Using these moments, $M_{i\ max}$ and $M_{i\ min}$, the minimum elastic section modulus values, Z, can be determined from equation (7) on page 2.07. A

cross-section should be selected using the minimum Z values as a guide. If a section is chosen with a greater than minimum Z value the depth of the permissible tendon zone is increased. This will simplify the design of the tendon profile.

Having the cross-section, the moment due to dead load, $M_d$, can be found. The minimum prestressing force $P_{e\ min}$, can then be computed from equation (13) on Page 2.07. A suitable force $P_e$ can then be selected. If $P_e$ is larger than $P_{e\ min}$ there will be more freedom in choosing a tendon profile.

The permissible zone for the line of pressure should then be plotted, using equations (57) to (60). If the zone is too large the prestress force or cross-section size should be reduced. If the limits of the zones cross at any location the prestress force or section size should be enlarged.

A tendon profile should then be chosen within the permissible zone. It should be noted that the support reactions of a continuous beam cause secondary moments. The sum of the primary and secondary moments is called the resulting moment. At any section of a continuous beam, the effect of the prestressing force $P_e$ and the resulting moment $M_{ps}$ is equivalent to a force $P_e$ acting at an eccentricity $e_p$, which is called the line of pressure. A tendon profile which is coincident with the line of pressure is called concordant. This means that it does not produce secondary moments or support reactions. Equations (53) to (56) are used to calculate the stresses.



If the tendon is non-concordant then the line of pressure must be determined from equation (61). If the line of pressure lies within the permissible zone, the profile can be used. If not, it must be moved until it satisfies these requirements, which can be tedious.

The stresses at transfer must then be checked and the loss of prestress calculated at all the design cross-sections. The

ultimate flexural strength and ultimate shear resistance at critical sections must then be checked. End blocks are then designed, if necessary, and deflections checked. The design must be revised if the design criteria of any of these checks are not met.

## PRESTRESSED CONCRETE DESIGN EQUATIONS.

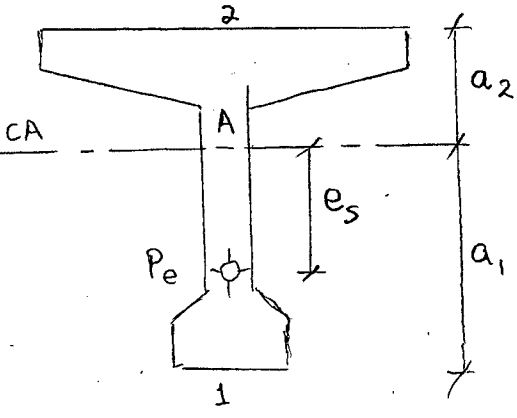
$$\text{Eqn. (3)} \quad \frac{P_e}{A} + \frac{(P_e\, e_s - M_{i\max} - M_d)}{Z_1} \geq f_{a\min_1}$$

$$\text{Eqn. (4)} \quad \frac{P_e}{A} + \frac{(P_e\, e_s - M_{i\min} - M_d)}{Z_1} \leq f_{a\max_1}$$

$$\text{Eqn. (5)} \quad \frac{P_e}{A} + \frac{(-P_e\, e_s + M_{i\max} + M_d)}{Z_2} \leq f_{a\max_2}$$

$$\text{Eqn. (6)} \quad \frac{P_e}{A} + \frac{(-P_e\, e_s + M_{i\min} + M_d)}{Z_2} \geq f_{a\min_2}$$

$$Z_2 = \frac{I_{CA}}{a_2} \qquad Z_1 = \frac{I_{CA}}{a_1}$$




---

from Eqn. (3)

$$P_e \geq \frac{(f_{a\min} Z_1 + M_{i\max} + M_d)\, A}{Z_1 + A.e_s}$$

from Eqn. (4)

$$P_e \leq \frac{(f_{a\max} Z_1 + M_{i\min} + M_d)\, A}{Z_1 + A.e_s}$$

from Eqn. (5)

$$P_e \leq \frac{(f_{a\max} Z_2 - M_{i\max} - M_d)\, A}{Z_2 - A.e_s}$$

from (eqn. (6)

$$P_e \geq \frac{(f_{a\min} Z_2 - M_{i\min} - M_d)\, A}{Z_2 - A.e_s}$$

**NOTE**

**The inequality ≥ changes to ≤ (and vice versa) whenever the sign or denominator is negative.**

satisfying <u>both</u> Eqns. (3) & (6)

$$P_{e\min} = \frac{[f_{a\min}(Z_1 + Z_2) + M_r]\,A}{Z_1 + Z_2} \qquad .....(13)$$

corresponding

$$e_s = \frac{Z_2 M_{i\max} + Z1 M_{i\min} + (Z_1 + Z_2)\,M_d}{[f_{a\min}(Z_1 + Z_2) + M_r]\,A} \qquad .....(14)$$

Satisfying both Eqns. (4) & (5)

$$P_{e\max} = \frac{[f_{a\max}(Z_1 + Z_2) - M_r]\,A}{Z_1 + Z_2} \qquad .....(13a)$$

corresponding

$$e_s = \frac{Z_2 M_{i\min} + Z_1 M_{i\max} + (Z_1 + Z_2)\,M_d}{[f_{a\max}(Z_1 + Z_2) - M_r]\,A} \qquad .....(14a)$$

---

<u>Minimum Section Size</u>

$$\frac{Z_1}{Z_2} \geq \frac{M_{i\max} - M_{i\min}}{f_{a\max} - f_{a\min}} = \frac{M_{range}}{f_{range}} \qquad .....(7)$$

satisfying all equations (3) (4) (5) and (6).

<u>Permissible $e_s$ zone SLS</u>:

$$e_s \geq \frac{M_{i\max} + M_d}{P_e} - \frac{Z_1}{A} + \frac{Z_1 f_{a\min}}{P_e} \qquad .....(15)$$

$$e_s \geq \frac{M_{i\max} + M_d}{P_e} + \frac{Z_2}{A} - \frac{Z_2 f_{a\max}}{P_e} \qquad .....(16)$$

$$e_s \leq \frac{M_{i\min} + M_d}{P_e} - \frac{Z_1}{A} + \frac{Z_1 f_{a\max}}{P_e} \qquad .....(17)$$

$$e_s \leq \frac{M_{i\min} + M_d}{P_e} + \frac{Z_2}{A} - \frac{Z_2 f_{a\min}}{P_e} \qquad .....(18)$$

Stresses at transfer:

$$\frac{P}{A} + \frac{(Pe_s - M_d)}{Z_1} \geq f_{a\min t_1} \qquad \text{.....(27)}$$

$$\frac{P}{A} + \frac{(Pe_s - M_d)}{Z_1} \leq f_{a\min t_1} \qquad \text{.....(28)}$$

$$\frac{P}{A} + \frac{(-Pe_s + M_d)}{Z_2} \leq f_{a\min t_2} \qquad \text{.....(29)}$$

$$\frac{P}{A} + \frac{(-Pe_s + M_d)}{Z_2} \geq f_{a\min t_2} \qquad \text{.....(30)}$$

Statically indeterminate prestressed concrete structures:

Replace $e_s$ in all equations by $e_p$ = eccentricity of the line of pressure.

$$\frac{P_e}{A} + \frac{P_e e_p}{Z_1} - \frac{M_{i\max} + M_d}{Z_1} \geq f_{a\min} \qquad \text{.....(53)}$$

$$\frac{P_e}{A} + \frac{P_e e_p}{Z_1} - \frac{M_{i\min} + M_d}{Z_1} \leq f_{a\max} \qquad \text{.....(54)}$$

$$\frac{P_e}{A} - \frac{P_e e_p}{Z_2} + \frac{M_{i\min} + M_d}{Z_2} \leq f_{a\max} \qquad \text{.....(55)}$$

$$\frac{P_e}{A} - \frac{P_e e_p}{Z_2} + \frac{M_{i\min} + M_d}{Z_2} \geq f_{a\min} \qquad \text{.....(56)}$$

Permissible Pressure Zone

$$e_p \geq \frac{M_{i\max} + M_d}{P_e} - \frac{Z_1}{A} + \frac{Z_1 f_{a\min}}{P_e} \qquad .....(57)$$

$$e_p \geq \frac{M_{i\max} + M_d}{P_e} + \frac{Z_2}{A} - \frac{Z_2 f_{a\max}}{P_e} \qquad .....(58)$$

$$e_p \leq \frac{M_{i\min} + M_d}{P_e} - \frac{Z_1}{A} + \frac{Z_1 f_{a\max}}{P_e} \qquad .....(59)$$

$$e_p \leq \frac{M_{i\min} + M_d}{P_e} + \frac{Z_2}{A} - \frac{Z_2 f_{a\min}}{P_e} \qquad .....(60)$$

**equation (14) becomes:**

$$e_p = \frac{Z_2 M_{i\max} + Z_1 M_{i\min} + (Z_1 + Z_2) M_d}{[f_{a\min}(Z_1 + Z_2) + M_r] A} \qquad .....(61)$$



### 2.1.3 Load-balancing Method.

The design of prestressed concrete beams by the load-balancing method is an alternative approach, and is described by Lin (4). In this approach, prestressed concrete is seen primarily as an attempt to balance a portion of the load on the structure. The approach is similar for statically indeterminate beams, especially for preliminary designs. A draped cable produces a distributed upward load, which is used to balance a portion of the applied loads. The loading under which there will be no deflection anywhere along the beam is thus known, and the net deflection produced under any other condition of loading is simply computed by treating the extra load as acting on an elastic beam. If the effective prestress balances the sustained loading, the beam will remain perfectly level regardless of the modulus of elasticity or the flexural creep of concrete.

A continuous beam under the balanced action between the vertical component of the prestress and the applied external load has a uniform stress $\sigma$ across any section of the beam:

$$\sigma = \frac{P_i}{A_c}$$

For any change from the balanced load condition, ordinary elastic analysis can be applied to the extra load to obtain the moment M at any section, and the resulting stresses computed from the formula:

$$\sigma = \frac{M y}{I}$$

Thus, after load balancing, the analysis of prestressed continuous beams is reduced to the analysis of a non-stressed continuous beam. As this analysis will apply only to the unbalanced portion of the load, any inaccuracies in the method will be relatively insignificant.

To determine the prestress required to balance a uniform load, the most economical location of the cable should first be found. This is one with maximum sag so that the least amount of prestress will be required to balance the load. The prestress force $P_i$ required to balance the load is then:

$$P_i = W \times r$$

(distributed load × radius of curvature of cable profile)

and the stresses produced over the supports due to any additional load can be determined.

The load to be balanced depends on each individual case. The dead load can be completely balanced although it is not necessary. If the live load to be carried by the structure is high compared to the dead load, it may be necessary to balance some of the live load as well. When evaluating the amount of live load to be balanced by prestressing, the real load should be considered and not the design live load. If the live load

is permanent, then a larger portion can be balanced than if the load is temporary.

The load balancing method is being complicated due to the assessment of time dependant prestress losses, ie.with passage of time different amounts of loading are being balanced. It is therefore also a method which is useful as a rough guide but cumbersome to use for a final design operation.

## 2.2 Proposed Approach to Prestress Design

Instead of continuing with the existing design method, which is both cumbersome and satisfies neither the ultimate nor serviceability limit states directly, a new approach is proposed which is based primarily on the ultimate limit state, with subsequent checks made for serviceability requirements.

It is envisaged that beam systems be designed span by span for ULS requirements using collapse principles (plastic hinges) to determine cross-section size and shape and prestress location and requirements. Once the ULS requirements are satisfied, the structure would then be checked for SLS requirements such as cracking and deformation limits.

The proposed design procedure would be to first select a suitable cross-section, either from experience or using the span/depth limitations in Table 8 and 9 of the code, SABS 0100 Part 1, multiplied by 1,5 as a guide. The slenderness can be assessed using section 3.3.1.3 of the code. The construction joints must then be planned and hence the points along the span from which the jacking will be done.

The next step is to identify the critical sections along the beam for the ULS of flexure. A plastic analysis of the structure, similar to that used for steel structures, using the applicable ULS loading must then be performed using the critical sections as the location of plastic hinges. For continuous beams this plastic analysis can be done span by span, with the plastic hinges required for collapse placed at the points of maximum bending moment. The ratios of plastic

moments of resistance, negative to positive bending moments, must be carefully chosen not to exceed the maximum allowable percentage redistribution of bending moments. Redistribution of bending moments from elastic to plastic states must be limited because of implied ductility achievablity. An equation should be developed to guide the designer on how much redistribution can be expected and allowed. These implications are discussed later in this thesis.

The prestress requirements for the ULS of flexure at the critical points of flexure should then be calculated. A suitable prestress system can then be selected. The prestress losses should then be estimated. The prestressing requirements should be planned such that the positive bending moments are covered by prestress alone and the negative bending moment requirements are covered by the prestress plus some reinforcement. The type and number of cables required will affect the maximum eccentricities at each section and may thus necessitate more than one cycle of design calculations. The ductility at the support hinges should be checked such that it complies with the assumed percentage redistribution of bending moments at these points. If the ductility is insufficient, the plastic analysis should be repeated with a smaller amount of redistribution.

A cable profile should then be decided on based on the above results. The corresponding prestress and load deflections should then be calculated. This would be done by an elastic analysis performed on a computer. The allowable limits should be checked and the profile adjusted if necessary.

The prestress losses should then be evaluated at the critical sections and at the anchorages. If the effective prestress differs by more than 10% from the initial assumptions made, those steps should then be repeated.

The SLS of cracking at the critical sections should then be checked and reinforcement added along the tensile face of the member if necessary. The shear reinforcement can then be calculated and the endblock design performed for posttensioned

beams. This procedure involves the least amount of unnecessary repetition of work and produces a safe and cost effective structure.

# CHAPTER 3 GENERAL THEORY FOR BEAM DEFORMATIONS

In order to make definite recommendations for the ductility limitations and resulting allowable bending moment redistribution coefficients which may be applied in a general design procedure as proposed in this thesis, it is essential to investigate these phenomena for structures with different cross-section shapes. In order to predict results which can be compared with laboratory investigations a rigorous analysis procedure must be developed. The general theory is developed in this chapter while the more detailed procedure is covered in Chapter 4.

The object of this chapter is to show and explain the methods used to find a deformation at any point on a prestressed concrete beam with given dimensions, prestressing, loading and state of plasticity. These methods were used in the development of the computer program described in this thesis.

Firstly, it will be explained how to obtain the moment-curvature relationship for a given cross-section, and then, using this relationship, the attainment of the required deformation, be it deflection or rotation, for a particular beam.

The sign convention used below assigns a positive sign to compressive stresses and strains and to sagging bending moments. Positive cable eccentricity is above the centroidal axis and will result in a positive moment.



## 3.1 Curvature

### 3.1.1 Curvature in the Elastic Zone

For a given applied moment, insufficient to cause cracking in the section, the compressive concrete force balances the tensile concrete and untensioned steel forces and the prestress steel

force. The curvature for a cross-section within the elastic zone is defined as:

$$\phi = 1/r = M/EI = (\epsilon_c - \epsilon_s)/d \quad \text{or} \quad \epsilon_c/x$$

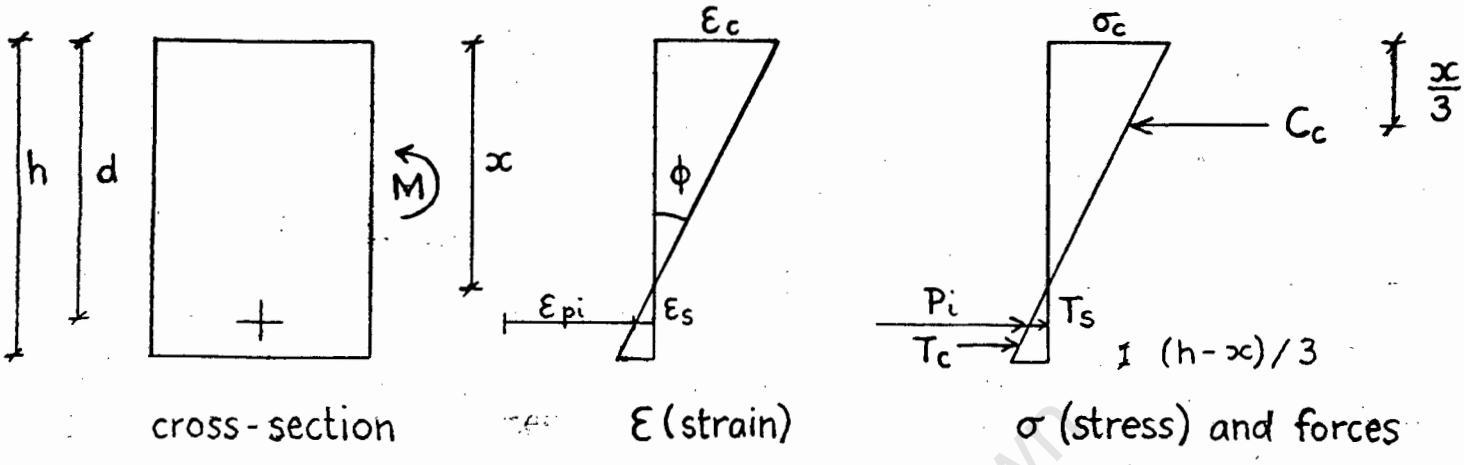
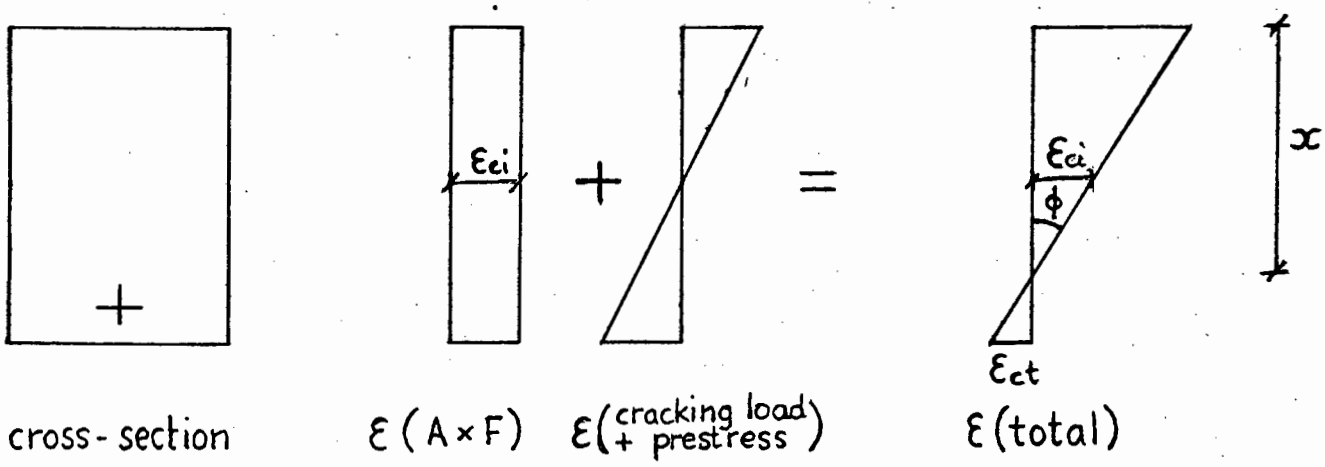



**Fig 3.1.1 Diagram to show definition of curvature**

### 3.1.2 Determination of the cracking moment and curvature

To find the curvature and thus moment at cracking, the modulus of rupture of the concrete ($f_{ctm}$) should be known. The corresponding strain ($\epsilon_{ct}$) can be found by dividing fctm by the Elastic modulus of the concrete. The force due to prestessing is found by multiplying the relevant prestress, $f_{pi}$, by the area of the tendons. This force is then divided by the area of concrete to give the compressive stress in the concrete due to the prestressing. The corresponding axial compressive strain in the concrete is then found by dividing the concrete stress by the Elastic modulus of the concrete. The curvature at cracking is then the tensile cracking strain subtracted from the concrete compressive strain, all divided by half the depth, as shown diagrammatically and mathematically below. The cracking moment is the cracking curvature multiplied by EI. This means that the concrete is assumed to behave elastically over the whole section up to cracking of the tensile fibre.




$$P_i = f_{pi} \times A_{ps}$$
$$\sigma_{pi} = P_i / A_c$$
$$\epsilon_{ci} = \sigma_{pi} / E_c$$
$$\phi_{cr} = (\epsilon_{ci} - \epsilon_{ct}) / (h/2)$$
$$M_{cr} = EI \times \phi_{cr}$$

**Fig 3.1.2 Procedure for finding cracking curvature and moment**

In the case of an eccentric prestress tendon, a bending moment will be caused by the force exerted by the tendon, and the $M_{cr}$ will therefore be:

$$M_{cr} = (\phi_{cr} \times EI) + M_{ps},$$

where $M_{ps}$ is moment due to the tendon eccentricity at that section.

### 3.1.3 Curvature after cracking

To find the moment-curvature relationship for a section after cracking, the tensile strength of the concrete is ignored as it has begun to crack. The relationship described above no longer applies as the concrete no longer behaves elastically. The compressive concrete force is thus balanced by the prestress

force and the additional tensile force in the steel. The curvature is defined as:

$$\phi = (\epsilon_c - \epsilon_s)/d$$

for a certain applied moment. Figure 3.1.3 shows this diagrammatically.

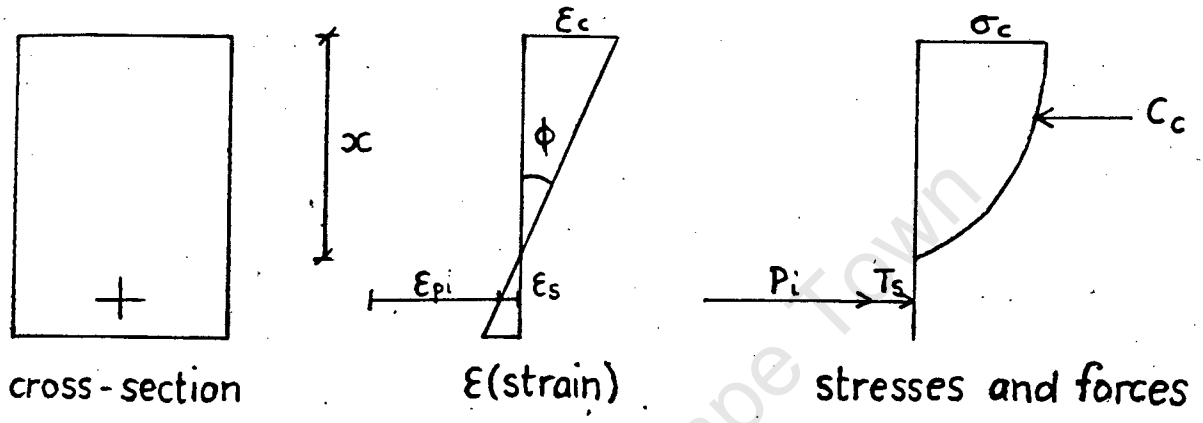



Fig 3.1.3 Diagram for cracked cross-section

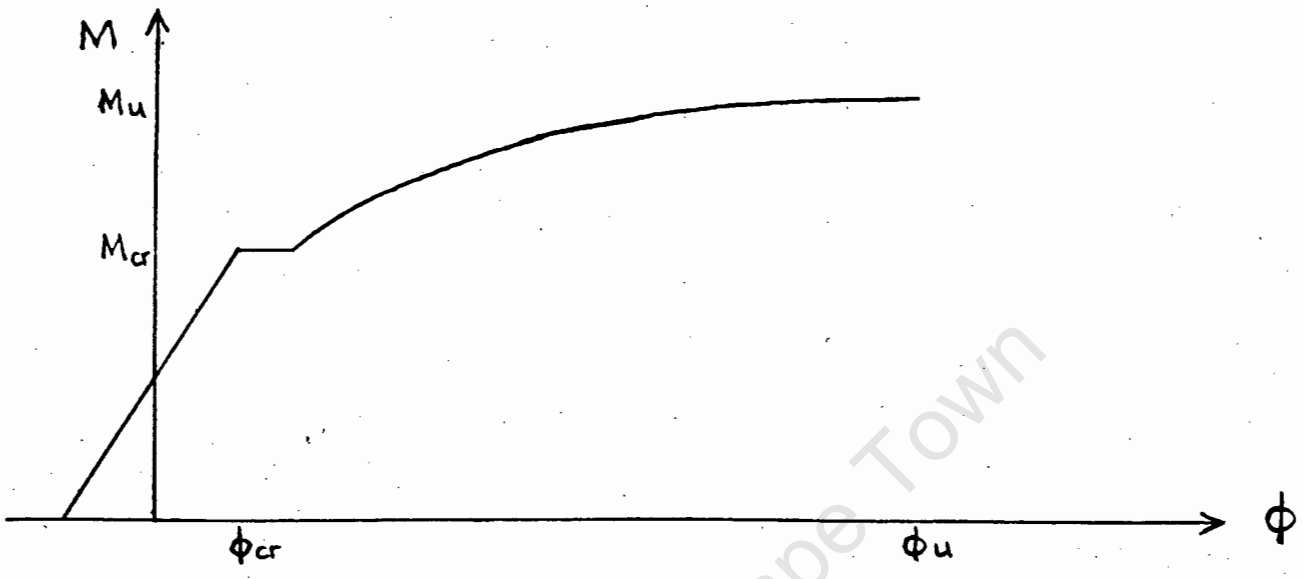





**Fig 3.1.4** **Plot of moment-curvature relationship for a typical section**

### 3.1.4 Correction for Tension Stiffening

The method for finding the curvature at a section after cracking, as described in section 3.1.3 above, is applicable only at that section where the crack occurs. In the zones between cracks, the behaviour is intermediate between the two states described above. The C.E.B. Manual 'Cracking and Deformations' (5) refers to the uncracked section described in section 3.1.1 as State I, while the cracked section described in section 3.1.3 is referred to as State II-Naked.

The reason for these intermediate states is the process of tension stiffening. Until the tensile strength of the concrete

is reached, the beam will be in State I. The first crack will form at the weakest section when the tensile strength of the concrete is reached. At the point where the concrete cracks, the reinforcement will take up all the tension previously carried by the concrete in tension. The steel will therefore undergo a sudden increase in stress, and thus strain. Due to this increased stress the bond strength between the steel and concrete may be exceeded resulting in a certain amount of slippage. The crack will then open up due to the differential movement between the steel and concrete. It is assumed that the slippage is equal on each side of the crack in order that strain compatibility may be applied at the crack

Due to the bond between the concrete and steel, the concrete resists the extension of the steel. The bond transfers tensile force from the steel to the concrete. The bond may be broken at the crack but intact further from the crack, and thus the tensile force transferred to the concrete by the bond increases away from the crack. Close to the crack the sections thus behave somewhere between State I and State II. At a distance further from the first crack compatibility of strains between the steel and concrete is re-established and the section will behave as a homogeneous section (State I). It will only be at or beyond this distance that the next crack forms.

Fig 3.1.5 shows the formation of cracks. The stresses in the bottom fibre of the concrete are plotted to demonstrate the effect of tension stiffening. Fig 3.1.5 (a) shows a section along the beam. Assuming a constant bending moment along the beam, the stress in the bottom fibre of the concrete is constant along the beam until the first crack occurs, as shown by the broken line in Fig 3.1.5 (b). As the load, and thus the bending moment increases, so too does the stress in the concrete. Since the concrete tensile strength is variable, the first crack will occur at the weakest point when the cracking stress at that point is reached. As the crack occurs, all tensile stresses in the concrete are transferred to the steel, and the stress along the beam is now as shown by the solid line in Fig 3.1.5 (b).

As the moment increases, so too does the stress in the concrete, and the next crack will occur at the next weakest section, provided it is far enough from the first crack that the bond strength has not been effected by it. The straight broken line in Fig 3.1.5 (c) shows the cracking stress at the first crack, the solid line shows the additional stress just before formation of the second crack, and the other broken line shows stress after formation of the second crack.

When modeling this effect, however, the concrete must be assumed to be uniform, and therefore the cracking stress, $f_{ctm}$, is therefore constant throughout the beam. Cracks are thus assumed to occur at the same time at a certain distance S apart.

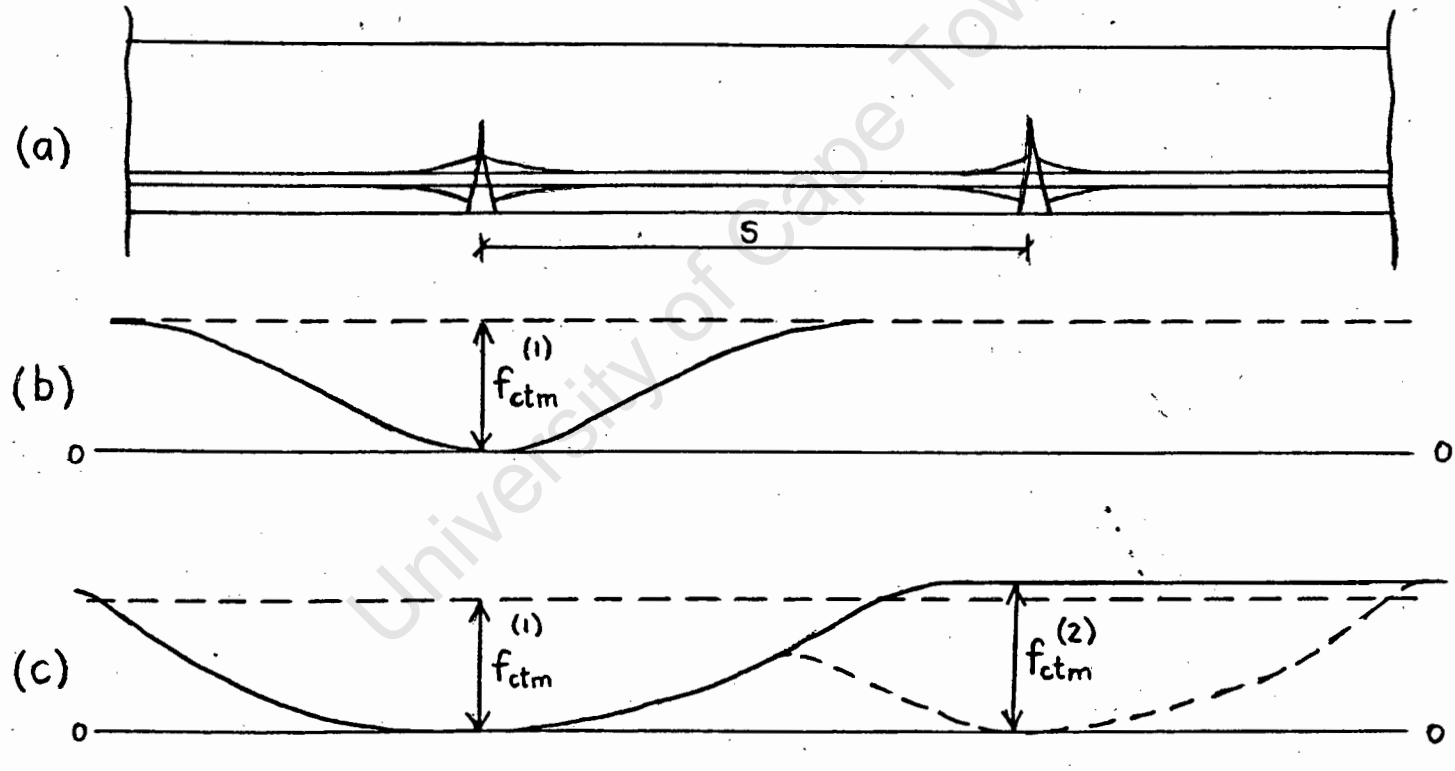





**Fig 3.1.5** **Demonstration of the principal of tension stiffening**

A model has been developed in the CEB manual referred to, which gives an equation for finding the average intermediate strains

in the cracked zone. The model divides the length $\ell$ of the element under consideration into two parts; one in State I (uncracked), the other in State II - naked (fully cracked). Length $\ell 1$ is the lenght of the uncracked part, while $\ell 2$ is the length of the fully cracked part. The equation for the average strain in the tension steel is as follows:

$$\epsilon_{sm} = \Delta\ell/\ell = (\Delta\ell 1 + \Delta\ell 2)/\ell = (\ell 1\epsilon_{s1} + \ell 2\epsilon_{s2})/\ell$$

or,

$$\epsilon_{sm} = (1 - \zeta)\epsilon_{s1} + \zeta\epsilon_{s2}$$

where $(1 - \zeta) = \ell 1/\ell$ and $\zeta = \ell 2/\ell$.

Similarly, the average strain in the concrete at the extreme compressive fibre is:

$$\epsilon_{cm} = (1 - \zeta)\epsilon_{c1} + \zeta\epsilon_{c2}.$$

The coefficient $\zeta$ is defined as:

$$\zeta = 1 - (a1a2 \cdot (\sigma_{sr}/\sigma_{s2})^2) = 1 - (a1a2 \cdot (M_{cr}/M)^2)$$

and $\zeta = 0$ for $\sigma_{s2} < \sigma_{sr}$ or $M < M_{cr}$

where $M_{cr}$ is the cracking moment and a1 and a2 are coefficients characterizing the bond quality of the steel and the influence of the duration of application or of repetition of loading respectively. Suggested values are as follows:

a1 = 1.0 for high bonding (deformed bars)
= 0.5 for smooth bars

a2 = 1.0 for first loading
= 0.5 for long term loads or for a large number of cycles of load

The factor a1 for prestressing steel would be close to 0.5 as the bond strength is low, while for high yield deformed reinforcing steel, the bond strength is high, and so a factor of about 1.0 would be used. For a combination of prestress and reinforcing steel, the factor would depend upon the ratio of area of prestressed to reinforced steel. The steel with the greater area would take more of the stress. The distance of the steel from the centroid would also have an effect on the factor. Steel closer to the centroid would have less effect than steel further away as the strains are smaller and their effect has a smaller leverarm.

The factor a2 would be 1 for checking the results of a laboratory test, where the beam is loaded to failure, but for a design analysis, 0.5 would be used as the beam would be subjected to long term loading or repetative loading.

From the average concrete and steel strains, the curvature can easily be determined by

$$\phi = (\epsilon_{cm} - \epsilon_{sm})/d,$$

which will give the average curvature for the applied moment.

Appendix D deals with an example showing the effects of tension stiffening.

## 3.2 Calculation of deformation by integration principles

The deflection/angular rotation at any point of a member can be obtained by integration of the curvature along the member and applying the principle of virtual work.

From the loading on the beam, a bending moment diagram can be found, and thus the curvature at any section can be found from the moment-curvature relationship for that section, as explained in section 3.1. This procedure is dealt with separately for statically determinate and indeterminate beams.

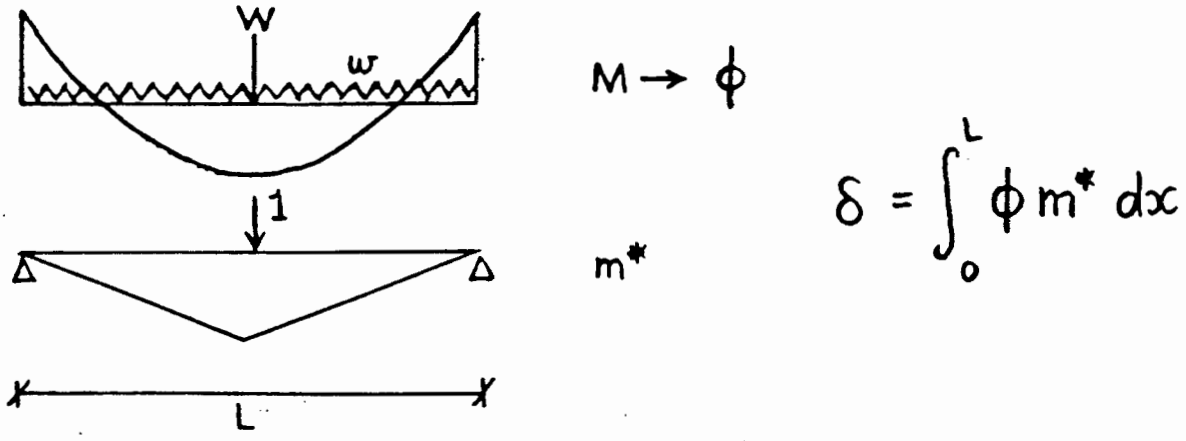



Fig 3.2.1 Principle of virtual work

### 3.2.1 Statically determinate beams

For statically determinate beams, only one integration cycle is required to calculate the deformation. The integration is done as indicated in Fig 3.2.1. (on a statically indeterminate beam)

### 3.2.2 Statically indeterminate beams

This is dealt with by way of a fully fixed beam. An unknown amount of redistribution of bending moments takes place in statically indeterminate structures when portions of the beam crack or become plastic. The beam becomes soft in this cracked, or plastic, zone, and therefore a greater portion of the increasing load is taken up by the elastic ,or uncracked, zone. The procedure to obtain deformations then becomes an iterative one, as the amount of moment redistribution must first be found. For a fully fixed beam the rotation at the supports should always be zero. Therefore to find the amount of moment redistribution, a bending moment diagram giving zero end rotation should be found. Starting with the elastic bending moment diagram, the member end rotation should be found, using the principle of virtual work, as shown in Fig 3.2.1, with an applied unit virtual end moment instead of a unit virtual transverse force. The datum axis of the bending moment diagram must then be raised by a constant amount until a redistributed moment diagram giving zero rotation at the member ends can be found. A suitable iterative

procedure will satisfy this requirement. The required midspan deflection can then be found using this redistributed bending moment diagram as indicated in Fig 3.2.1.

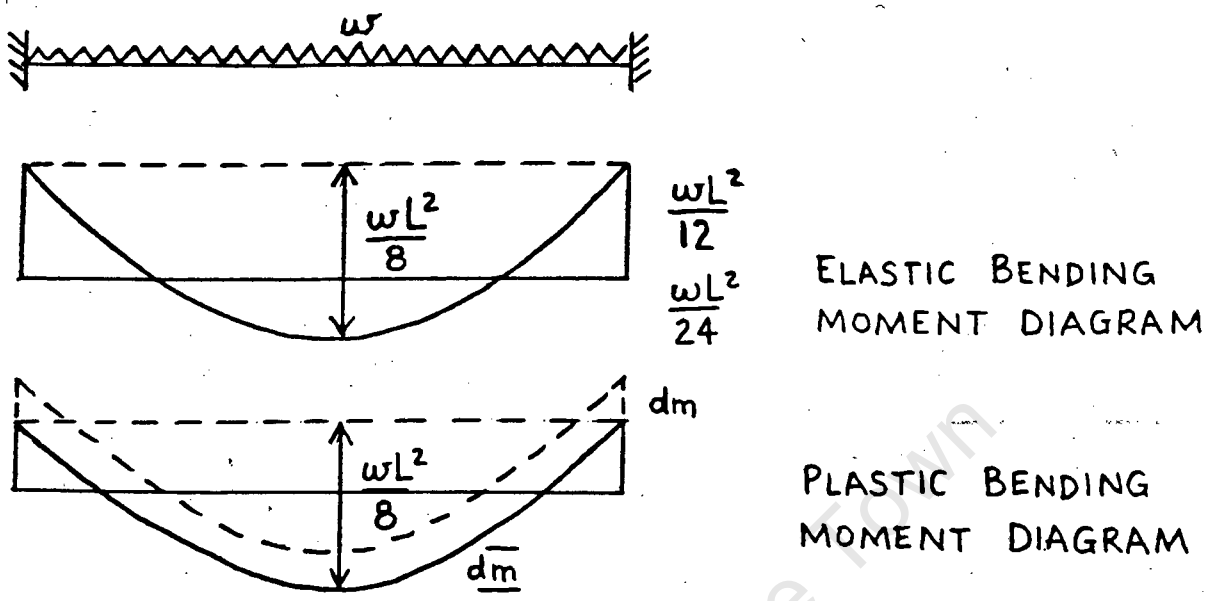



**Fig 3.2.2 Effect of plasticity on the bending moment diagram**

Fig 3.2.2 shows a fully fixed beam with a uniformly distributed load, w. An elastic analysis gives end moments $-wL^2/12$ and a moment at the centre span of $wL^2/24$. The difference is thus $wL^2/8$. If cracking occurs at any point, that zone will be termed plastic, and a certain amount of moment redistribution will occur. The bending moment diagram will be lowered by a constant dm, as shown, with the difference between the hogging and sagging moments remaining $wL^2/8$.

# CHAPTER 4 DETAILED PROCEDURE TO DETERMINE DEFORMATIONS

The procedure described in chapter three is unsuitable for hand calculation and a computer program to perform such calculations has thus been developed as part of this thesis. The development of the program will be discussed in this chapter and the procedures and routines used will be explained. A listing of the program is included in Appendix B.

The program was developed in stages. During the investigation cantilever beams with straight tendon profiles were examined first, and thus initially only one cross section and a single bending moment diagram were used. Once this concept was fully understood, a simply supported beam was investigated, which required several cross-sections due to the draped cable profile, but was still statically determinate. Finally a fully built in (statically indeterminate) beam was investigated, which required several cross sections, as well as a variable bending moment diagram, due to the bending moment redistribution with onset of plasticity.

The program requires the initial choice of beam type, either cantilever, simply supported or fully fixed, and then calls the relevant controlling routine. Each such routine requires the definition of the stress/strain relationships for the materials to be used, the beam cross-sections, steel area and prestress requirements, and tendon profile equations. Once each beam has been defined in this manner, certain routines are carried out to determine the required deformations.

In 4.1, it will be explained how the data is entered in the program. 4.2 will then show how the curvature is obtained for a given moment at a given section, and 4.3 will show how this is used to find the required deformations and, in the case of the fully fixed beam, the redistribution of bending moments.

## 4.1 Input of Data

The three routines, CANTIL, SIMSUP and FULFIX control the calculations, depending on the type of beam, which is chosen at the start of the program. These routines then call the following routines in order to control data entry.

### 4.1.1 Material Properties

The properties of the materials used in the beam are input into a separate data routine before the program is run. These include the number of material types defined and the stress-strain relationships of each material. As many material types as required can be defined.

The stress-strain relationship is set up such that a certain number of stresses and their respective strains can be defined in the data routine. The number of points defined depends on the accuracy required by the user, and as many points as necessary can be defined. For any point in between those defined a linear interpolation is performed. The routine MATINP calls the stress/strain relationships already defined, or alternatively it requires new relationships to be entered, depending on the users choice. MATINP is the first routine called by the controlling routines.



### 4.1.2 Element Cross-section

The controlling routine then calls the routine ELINP, which defines the cross-section used for the beam under consideration. It is done by splitting the concrete cross-section into rectangular and triangular elements. An arbitrary reference line is chosen as the datum axis. This is best chosen as passing through the base of the lowest element. The rectangular elements are defined by the number 1; 2 represents upward pointing triangles and 3 downward pointing triangles. An example showing how a section is divided is shown in Fig 4.1.1. Elements should be entered in such an order that the bottom fibre of the first element is the lowest, and the top fibre of the last is the

highest. This facilitates the easy calculation of the total depth of the section. The material type, depth and breadth of each section is entered, as well as the distance from the datum axis to the bottom of each element.

Routine ELINP also requires that the number of prestress tendons be entered, as well as the number of separate divisions along the length of the beam, the length from the support to the end of each division, and the area of each tendon in each division. As the simply supported and fully fixed beams concidered are symmetrical, only half the length need be concidered. A number of divisions can be specified, each with different areas of steel, in order to facilitate the use of additional untensioned reinforcing steel over part of the beam length. This reinforcing steel is entered as a tendon with zero initial prestrain. In a division where no such steel exists, the area can be defined as zero.

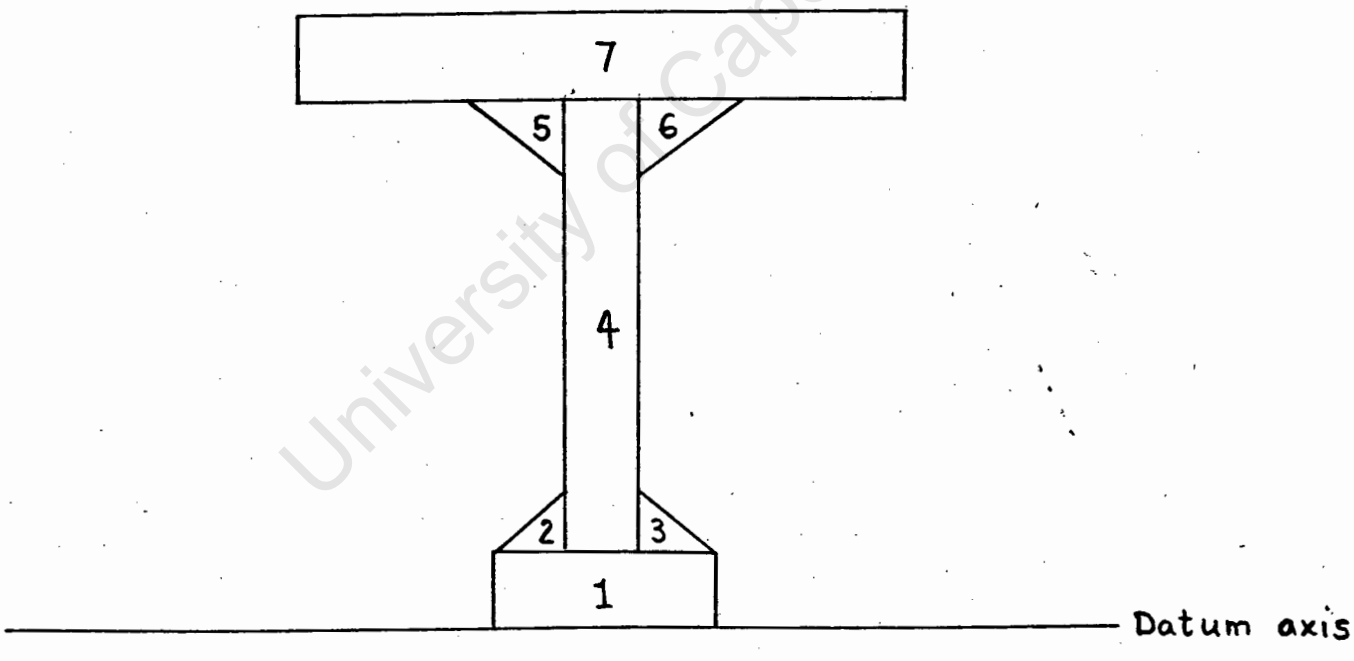





**Fig 4.1.1 Subdivision of cross-section**

#### 4.1.3 Other Beam Data

The controlling routines then require that the initial prestrain in each tendon be entered, as well as the length of the beam, the tendon profile quadratic equations, and the dead and live loads.

## 4.2 Determination of Curvature at a Cross-section

At each section under consideration the cross-section is defined and it is required to find the curvature corresponding to the moment at that section. Section 3.1 described how to obtain a moment-curvature relationship for a certain cross-section. This section will show how to obtain the curvature corresponding to a given moment for a given cross-section computationally.

The first step is to assume a certain strain in the top concrete fibre and the depth of the neutral axis, and then to balance the internal cross-sectional forces and find the corresponding moment capacity of the cross-section under consideration. In this program an initial concrete strain is set at 0.0015. The moment can then be found and compared to the required moment at that section. The Newton- Raphson iterative method is then used to set the strain to a new value, which should give a moment closer to that required. The iteration procedure is continued until a strain is obtained which gives a sufficiently accurate moment. This iterative procedure is shown in Appendix A. The iteration is done in routine DESIGN. The method used to balance the forces and find the moment is shown below. The program deals with positive and negative moments separately. This is done to exchange 'greater than' and 'less than' conditions and to allow for other differences.



### 4.2.1 Determination of Cracking Moment

Routine DESIGN checks to determine whether or not the section is cracked for the given moment. This is done by calling routine CRACKM, which determines the cracking moment and curvature for the section under concideration. This routine uses the method discussed in 3.1.2. If the required moment is less than the cracking moment, the moment-curvature relationship is elastic, and therefore the moment and curvature can be determined without the iteration described above. The curvature is just the moment divided by EI. If the required moment is greater than the cracking moment, then the iterative procedure must be applied.

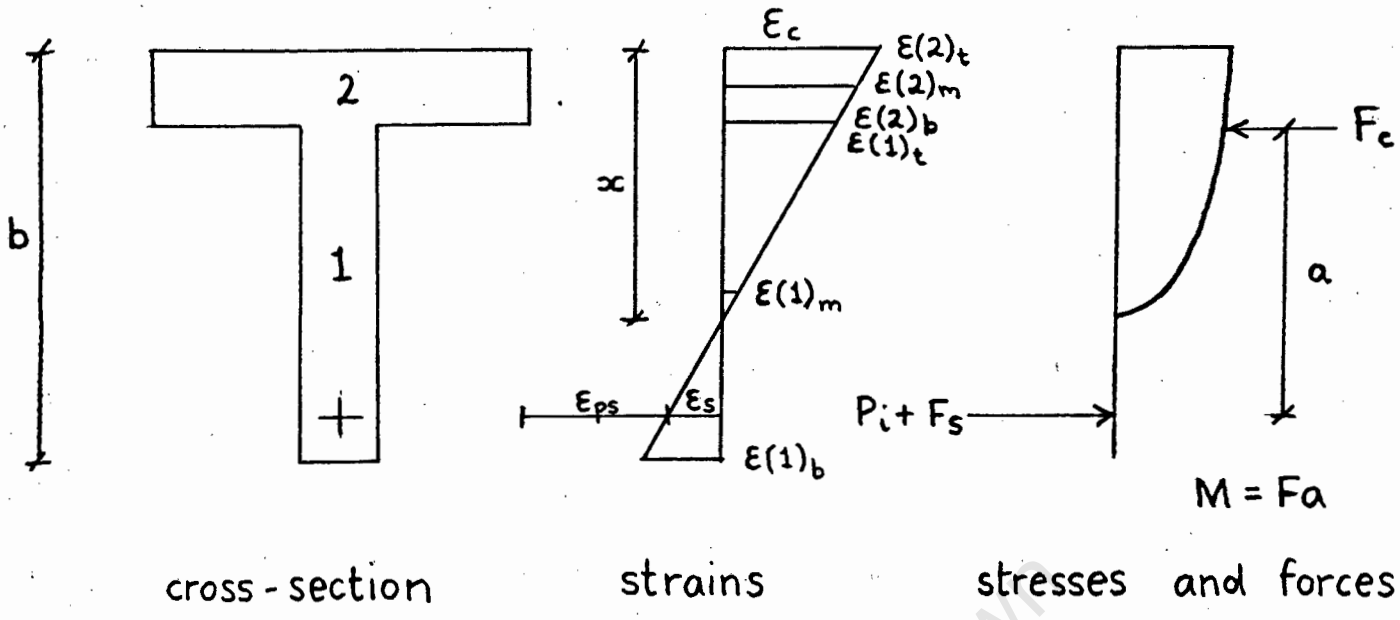
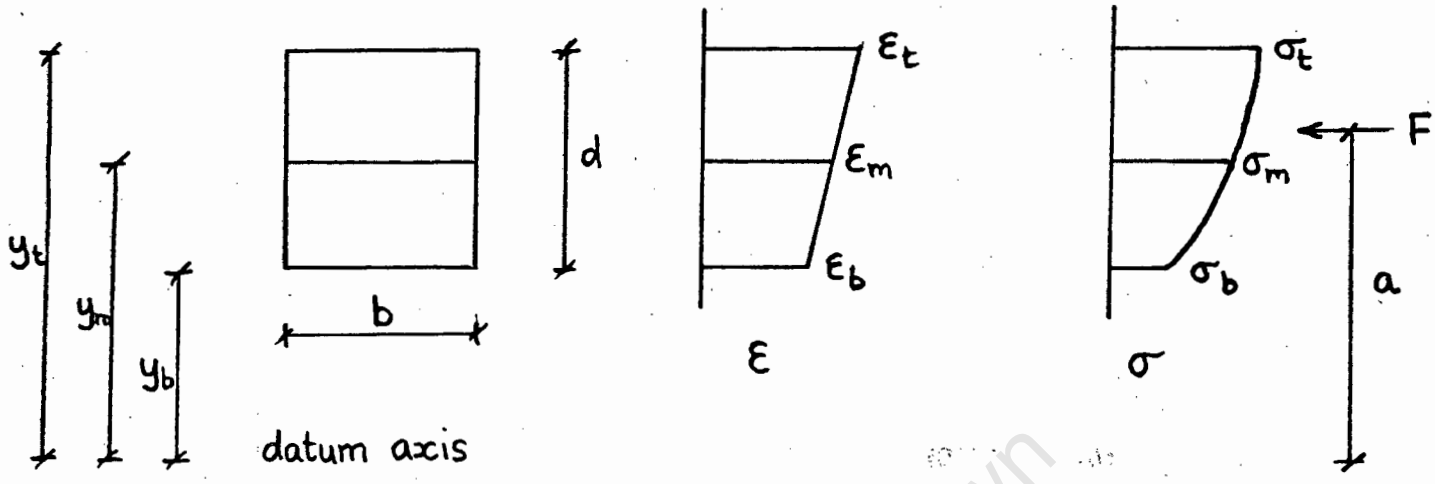



**Fig 4.2.1** Calculation of strains, stresses, forces and moments

#### 4.2.2 Determination of Moment and Curvature

For cracked sections, routine ANALYS controls the balancing of forces and the determination of the moment and curvature for the section under consideration with a certain strain in the extreme compression fibre. The strain in this fibre is set and the value of x, the distance from this fibre to the point of zero strain, is initially guessed as being half the depth of the section. This is done in the routine GETSTR. For each cross-sectional element, the strain in the top, middle and bottom fibre is determined, as the strain varies linearly down the section. This is done in the GETETA routine. Routine GETSIG finds the respective stresses from the material properties. As this method is used to find the moment-curvature relationship for the section after cracking only, the tensile stresses in the concrete are ignored. The program will therefore find the point of zero stress, and then re-adjust to make this the bottom of the concrete section for the purposes of calculation. In routine FORCE, the stress at each of the defined points is multiplied by the breadth of the element at that point and using Simpson's integration equation, the force on each element is found. The moment is found by multiplying the

distance from the datum axis to the fibre concerned to the stress times breadth, as for force, and again applying Simpson's equation. The method is shown graphically in Fig 4.2.2.




$$F = (d \cdot b)/6 \cdot (\sigma_t + 4\sigma_m + \sigma_b)$$

$$M = (d \cdot b)/6 \cdot (\sigma_t \cdot y_t + 4\sigma_m \cdot y_m + \sigma_b \cdot y_b)$$

**Fig 4.2.2 The application of Simpson's equation to find force and moment on each element**

The forces and moments for each element are then both summed to find the total compressive force due to the concrete, and it's moment. The tensile force in the steel is found simply by getting the stress from the known strain using the stress-strain relationship for the steel. It is then multiplied by the area to find the force, which is multiplied by the distance from the base axis, to get the moment due to that steel. The strain is determined in routine STETA, and the stresses, forces and moments in routine STSIG. It is required that the steel and concrete forces should balance, and so the value of x is changed until the forces balance. The Newton-Rhapson method, described in Appendix A, is used to find a new x value and the process is repeated with this new x value, the strain in the top concrete fibre being held constant. This procedure is continued until a value for x is found such that the forces are balanced. The moment is then known and the curvature is $\epsilon_c/x$. This therefore represents one point on the moment-curvature relationship plot.

In this case, the Newton-Rhapson iteration process minimizes the equation F = f(x), where F is the sum of the cross-sectional extensional forces, which must be zero, and x is the distance from the extreme concrete compression fibre to the centroid, which is the variable on which the balancing of forces depends.

#### 4.2.3 Correction for Tension Stiffening

The principle of tension stiffening was described in 3.1.4. Once the curvature corresponding to the required moment for the cross-section is found, the effect of tension stiffening can be corrected for if required by calling the routine TENSTIF. The model described in 3.1.4 is used. As the program is being used for an investigation, the factors a1 and a2 were used to give the least amount of redistribution of bending moments as this would be conservative. For a design, these factors would have to be changed. An example is given in Appendix D showing the effect of tension stiffening.



### 4.3 Determination of Deformations

The next step makes use of the method described in 4.2 to find the deflection at a given point on a specified structure. The requirements and procedures for cantilevers, simply supported and fully fixed beams are different, and shall therefore be discussed separately.

#### 4.3.1 Cantilever Beams

For cantilever beams, only one iteration cycle is required per load increment. If there is a plastic zone, it's location is found. It is first checked that the beam does reach cracking at some section under the given load conditions. The first section to reach cracking will be that with the highest moment, in this case at the support. As the tendon profile is straight, the cross-section at any point along the length of the beam is the same.

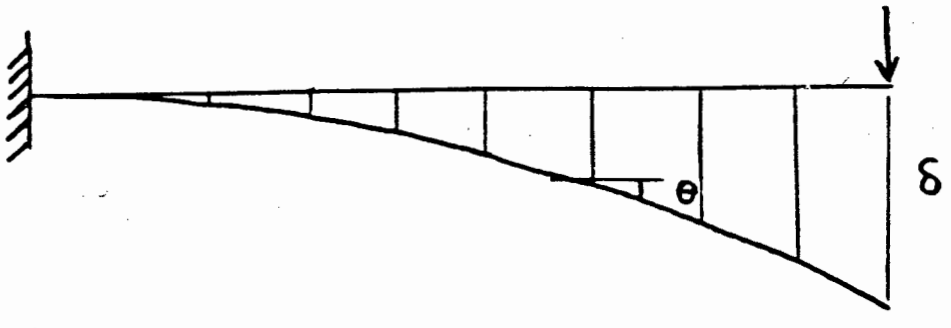
The cracking moment for the section at the fixed end is then found using the routine CRACKM, and if it is greater than the moment at that section due to the loading on the beam, then cracking does not occur, and the element is elastic throughout it's length. If it is less, then cracking occurs and the region of the plastic zone must be found.

This is done using the Newton-Raphson iteration process, with M(cr) - M(applied) as the function to be reduced to zero and the length from the fixed end to the section under consideration as the variable. The length is initially guessed as being zero, and iterated until the distance is found which gives the correct region of the plastic zone.
This could have been done by applying the following:

$$\text{If } M_{sup} > M_{cr} \text{ then } x_{cr} = L\sqrt{M_{cr} / M_{sup}}$$

$$= \text{distance from tip of cantilever to } M = M_{cr}$$

Once this region has been found, the beam is then split into the plastic (cracked) region and the elastic (uncracked) region. Each region is divided into eight equal lengths, and, using the routine DESIGN described earlier, the curvature is found at the relevant point from the applied moment diagram. Simpson's rule for integration is then used to find the deflection for the beam. The contribution from the elastic and plastic sections are determined separately due to the discontinuity in the moment-curvature relationship plot. The deflection is given by the equation as shown in Fig 4.3.1.



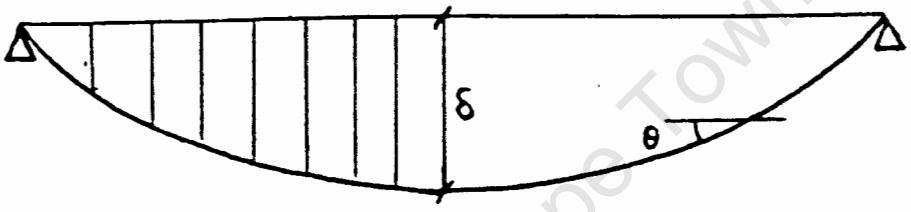

$$\delta = d/3(m_1\phi_1+4m_e\phi_e+2m_o\phi_o+m_9\phi_9)$$

e ≡ even ordinates
o ≡ odd ordinates

**Fig 4.3.1** **Simpson's rule to find deflection for a cantilever**

### 4.3.2 Simply Supported Beam

The procedure for simply supported beams is similar to that for cantilevers. The tendon profile is draped, and thus each cross-section will be different, but it is still statically determinate, and thus the procedure is similar to that described above. Obviously the maximum moment will now be at the centre of the beam, and this is where cracking will first occur. The location of the plastic zone is found as before, except the initial guess is taken as being at the centre of the beam. It is assumed that the beam and it's loading is symmetrical, and thus only half the beam need be considered for integration. The equation used is shown in Fig 4.3.2.




$$\delta = 2d/3(m_1\phi_1+4m_e\phi_e+2m_o\phi_o+m_9\phi_9)$$

**Fig 4.3.2** Simpson's rule to find deflection for a simply supported beam



### 4.3.3 Fully Fixed Beam

The procedure for a statically indeterminate beam is more complicated as the redistribution of moments must be taken into account. A fully fixed beam is assumed. The rotation at the fixed ends should be zero, and thus this rotation should first be ensured. The elastic moment diagram is first assumed, and the position of the plastic zones (there can be more than one in this case) is determined in a similar manner to that for determinate beams. Once their extent has been found the beam is divided into the two plastic and one elastic zones, each of which is divided into eight, and the beam end rotation is then determined in a manner similar to that described above. The method is shown diagramatically in Fig 4.3.3.

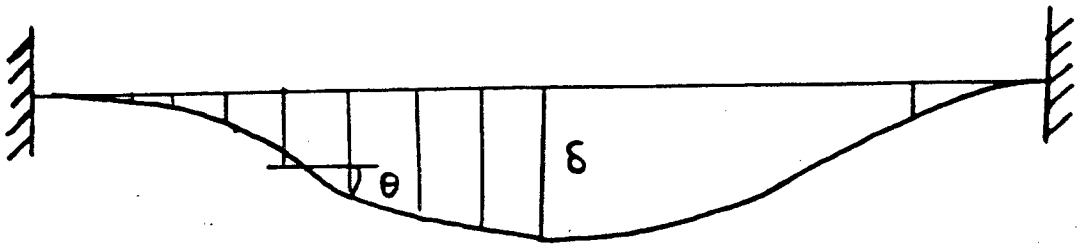



Fig 4.3.3 Simpson's rule to find end rotation for a fully fixed beam



If cracking has not yet begun, the rotation will be zero, in which case the deflection can be calculated in a similar manner. If cracking is present, then an iteration procedure must be carried out to adjust the moment diagram until the rotation becomes zero. The Newton-Raphson method is again used for this purpose, with the beam end rotation to be minimized to zero, and the moment being the variable. The datum axis of the bending moment diagram will be raised by a constant to allow for the redistribution, as explained in Section 3.2.2. The uniform increase is referred to as dm. A suitable value for dm will be found which results in zero end rotation, and once this has been found the deflection can then be evaluated in the same manner using the redistributed moment diagram.

# CHAPTER 5 DISCUSSION OF RESULTS

This chapter will discuss the results of several examples investigated using the computer program discussed in the previous chapter. First results obtained experimentally are compared to the results obtained using the program in order to validate the procedure and applied theory. The effect of tension stiffening on the ductility of a section and its effect on the bending moment redistribution will then be discussed. Then the behaviour of a few fully built in beams is investigated.

The effects of various factors on the bending moment redistribution will then be discussed using examples. A comparison will be made between the use of prestressing and reinforcing steel, showing their effects on the bending moment redistribution. The effect of the cross-section shape will also be investigated by looking at two extreme cases. The same examples will be investigated using design material factors on the material stress-strain relationships.



## 5.1 Experimental Results

Five simply supported beams were constructed and tested to determine moment-curvature relationships as part of the requirements for a B.Sc (Civ Eng) thesis. The results of the tests are shown in Appendix C, and are compared to the results obtained from the computer program. The beams are referred to as beams B1, B2, C1, C2 and D1. They each had rectangular cross-sections with the first two having 0,15% prestress steel, the second two having 0,25% prestress steel and the final one having 0,4% prestress steel.

### 5.1.1 Results of Beams B1 and B2.

The results of beams B1 and B2 are shown in Fig. C.4 in Appendix C. The curve for B1 is shown in black, B2 is shown in green and the theoretical curves are shown in red. The upper curve has

tension stiffening taken into account, while the lower curve does not. The curve taking the tension stiffening effect into account is extremely close to the test results, except at the higher curvatures, where the theoretical curve dips slightly below the experimental curves. As this is the area in which the beam is reaching its ultimate capacity, the beam is starting to deform rapidly. A slight increase in load will lead to a large increase in curvature, and hence deflection. It is thus very difficult to determine an exact moment-curvature relationship over this range and it is thus unlikely that the test results are extremely accurate in this region.

### 5.1.2 Results of C1 and C2

The results of the beams C1 and C2 are shown in Fig C.5 in Appendix C. The same colour coding is used. The curves for these two beams do not correspond as accurately as in the previous case. Here the curve ignoring the effects of tension stiffening corresponds closely to the test results, with the curve including the effect of tension stiffening being too high just after cracking. However, the accuracy of the determination of the concrete strength in this case is doubtful. It was determined as being 45 MPa, but was probably less, which would reduce the theoretical curve slightly.

### 5.1.3 Results of D1

The results of beam D1 are shown in Fig. C.6 in Appendix C. The theoretical results in this case are much higher than the test result. The accuracy of the test result is extremely doubtful, however. The beam is identical to C1 and C2 except for the increased steel area. The concrete strengths at testing were supposedly the same. If this were the case then beam D1 should be stronger due to the extra steel, however the moment-curvature plot is very similar to those for C1 and C2, with the latter part of the curve dropping below the curves of C1 and C2. It is thus likely that the concrete strength was much lower than indicated.

### 5.1.4 Discussion of Results

The above results are thus inconclusive in proving the accuracy of the program results, but indicate sufficient accuracy to be able to continue with the investigations. Great care was taken in the preparation and testing of beams B1 and B2, and in the determination of their concrete strengths, which was done under personal supervision. The accuracy of the other beams, however, is doubtful as the cube strengths were not representative due to curing anomalies.

To check the exact accuracy of the program a large number of beams with different cross-section shapes and material properties would have to be tested. It is very important that accurate stress-strain relationships for the materials involved are known, as these relationships form the basis of the program. Further inaccuracies could occur in the determination of the applied prestress force, especially in determining the losses.



## 5.2 The Effect of Tension-Stiffening

The example shown in Appendix D is used to illustrate the effect of tension-stiffening on the ductility of a section, and hence the bending moments after redistribution along a beam.

### 5.2.1 Discussion of Example

Fig. D.1 shows the moment-curvature relationships for the cross-section described in Appendix D. This is a typical moment-curvature plot and shows the effect of tension-stiffening on such a relationship. As described in section 3.1.4, an interpolation model is used to determine the curve for the tension-stiffening effect. As can be seen, the tension-stiffening curve is above that for which the effect is ignored. It's effect is thus to reduce the ductility of the section. The stiffness is greater at each section when tension-stiffening is taken into account since the curvature is less for a given moment diagram.

### 5.2.2 Effect on Moment Redistribution

Due to greater stiffness, the redistribution of bending moments will be smaller if the effect of tension-stiffening is accounted for. It should be noted, however, that the two curves shown in Fig. D.1 are two extreme cases. The lower curve represents the case in which the effect of tension-stiffening is ignored, while the upper curve represents the maximum effect of tension-stiffening observed for first time loading of the beam. For long term or repetitive loading the amount of tension-stiffening will produce intermediate results to those shown, and thus the bending moment redistribution will be intermediate.

### 5.2.3 Effect on the Investigation

The examples shown in Appendix E and F have been investigated ignoring the effect of tension-stiffening. The reason was that the extent of bending moment redistribution was being investigated and it was required to find the greatest possible amount of bending moment redistribution. If the exact behaviour of the beams were required a means of determining the exact amount of tension-stiffening would have to be determined experimentally. Instead, for these investigations the effect is ignored in the computational procedure, but its effects are noted.



## 5.3 Discussion of Investigation

The examples given in Appendix E and F were used to investigate the amount of bending moment redistribution that can be expected from various beams. The effect of the cross-section shape was investigated by considering two extreme shapes, an I-beam and a T-beam. The results for the T-beam are shown in Appendix E and those for the I-beam in Appendix F. The effect of untensioned reinforcement steel was also compared to that of prestressed steel by replacing half of the prestressing steel with reinforcement steel over the support area in both examples. The results are discussed below.

### 5.3.1 Results for T-Beam

The T-beam is an extreme case as the maximum positive moment capacity is considerably larger than the maximum negative moment capacity due to the compression flange. The concrete in the flange has a considerably large area, and thus for a negative moment the concrete in the flange is in tension and most of the web will be in compression. The flange thus provides no resistance to negative moment. However, if the moment is positive the concrete in the flange will be in compression, thereby providing considerable resistance to the moment.

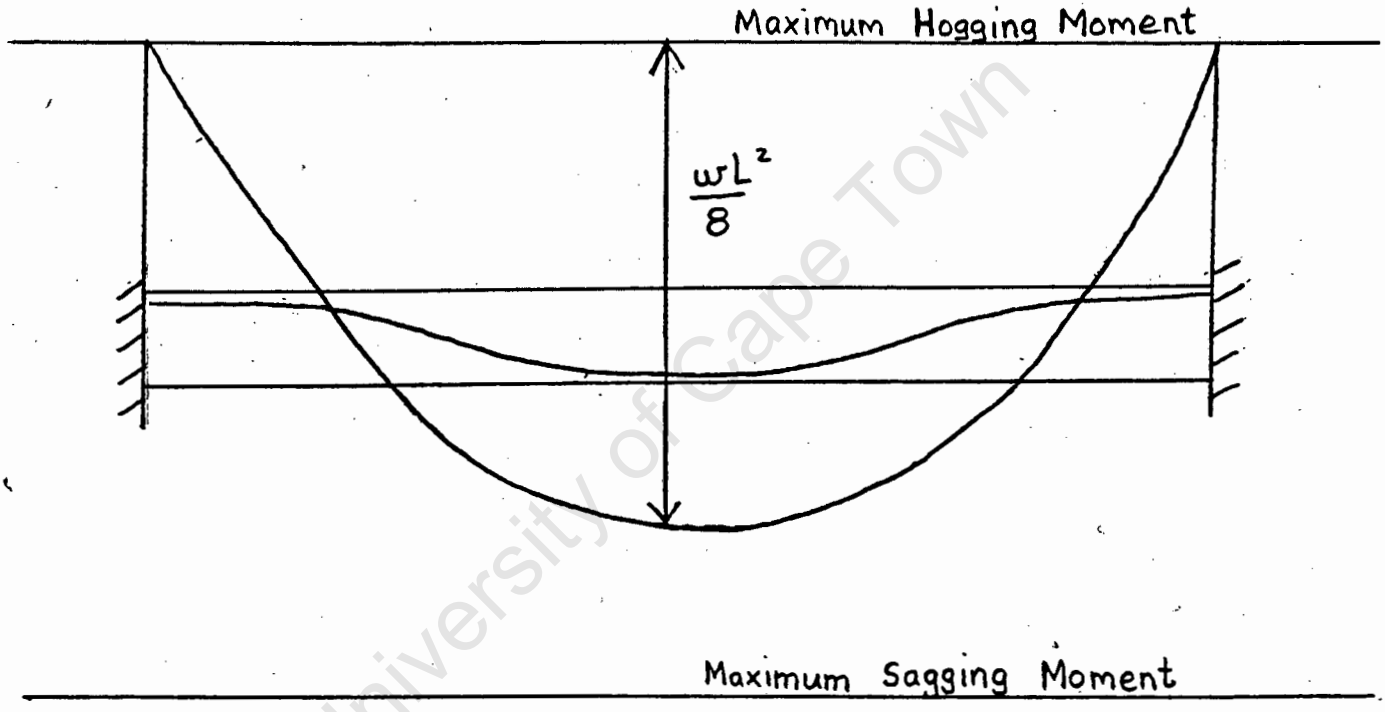





**Fig. 5.3.1 Ultimate Moment Range for T-Beam**

In order to obtain a collapse mechanism, it would be necessary to obtain plastic hinges at the supports and at the centre. In order for the centre region to reach its ultimate capacity a considerable amount of redistribution of bending moment is required. Appendix E(a) shows the results of a plastic analysis on the given beam. The actual bending moment redistribution which can be achieved is found to be only 17,132%. Only 70,66% of the moment capacity at the centre was reached before the strain capacity at the supports was exhausted. The strain capacity is exhausted at the supports in the steel and concrete

almost simultaneously in this case. If the effect of tension-stiffening were to be taken into account, the redistribution would have been even less. In this case the beam would rupture and fall down at the supports before reaching a collapse mechanism. To avoid this the ductility at the supports would have to be increased considerably.

### 5.3.2 Effect of Reinforcing Steel

In order to increase the ductility it was decided to substitute reinforcing steel for a portion of the prestressing steel at the supports. Half of the area of prestressing steel was replaced by untensioned reinforcing steel over the first 10 m from the supports. The area of steel replaced was such that the ultimate strength of each section was the same as before.

The result of this is given in Appendix E(b). It was found that the bending moment redistribution increased slightly to 17,227%. 73,46% of the central capacity was achieved before the strain capacity at the supports was exhausted. It can therefore be seen that the reinforcing steel provides slightly more ductility, but the increase is not significantly larger.

### 5.3.3 Results for I-Beam

The I-beam is the other extreme case as the maximum positive and negative moment capacities are almost equal. There are flanges on each side and the beam section is thus symmetrical. For a uniformly distributed load the elastic moment at the supports is $-WL^2/12$ and $WL^2/24$ at the centre. The difference is thus $WL^2/8$. If the maximum positive and negative moments are equal, the moments at the support and centre would have to be $WL^2/16$ each. This would require a bending moment redistribution of 25%. The example given in Appendix F(a) considers an I-beam with reinforcement steel replacing half the prestress steel as in the T-beam discussed in 5.3.2. In this case the extent of redistribution of bending moments is found to be only 17,18%. 89,57% of the central capacity was reached when the strain capacity at the supports was exhausted. In this case the strain

capacity in the steel at the supports was exhausted while the strain in the bottom concrete fibre only reached 0,096%. In the case of the T-beam, both concrete and steel strain capacities were exhausted almost simultaneously.

### 5.3.4 Discussion of Results

From the results discussed above it can be seen that it is very difficult to reach the ultimate capacity at the centre of any fully fixed beam, thereby causing a collapse mechanism, unless additional untensioned reinforcement is added around the supports in order to increase the section capacity in this area. The addition of reinforcing steel to prestressed steel is termed partial prestressing and the amount of moment redistribution required to utilise the full central capacity of the beam would be reduced.

### 5.3.5 Equation to Determine Redistribution

It would therefore be desirable to develop an equation which would give the maximum amount of bending moment redistribution that can be expected from a certain type of beam. The equation would have to take into account the shape of the cross-section, as well as the ratios of reinforcing to prestress steel. The equation could be of the form:

$$0 \leq red \leq 1 - k_1\left(0{,}8 - k_2 \frac{A_R}{A_{PS} + A_R}\right)$$

where red is the redistribution of bending moments such that

$$red = 1 - \frac{M\,(after\ redistribution)}{M\,(before\ redistribution)}$$

for the moments at the supports, and where $k_1$ is of the order of

1,10 for symmetrical sections

and 1,05 for T-sections

and M is the bending moment at the support

and $k_2$ is of the order of 0,05. $A_R$ is the reinforcing steel area at the supports and $A_{PS}$ is the prestress steel area at the supports. Several examples would have to be studied in order to accurately formulate such an equation.

### 5.3.6 Use of Design Material Factors

The same examples discussed above were investigated using design material factors of 1,15 on the material stress-strain curves. It was found that the bending moment redistribution in all cases was increased slightly.

In the case of the T-beam with no untensioned steel, the extent of redistribution was found to be 18,906%, with 80,88% of the central moment capacity reached. These results are shown in Appendix E(c). Appendix E(d) gives the results for the example with untensioned reinforcement. The extent of redistribution was 19,472%, with 81,58% of the central capacity reached. For the I-beam, the redistribution was 17,1%, which was the same as when the real material properties were used. The central capacity utilised was 85,69%. These results show that the use of material factors results in a less conservative design than the use of actual (average) material properties. This is undesirable since design values should always be more conservative.

# CHAPTER 6 CONCLUSION

The existing design approach to prestressed concrete beams is antiquated and time consuming, satisfying neither the ULS or SLS requirements directly

A design approach using plastic principles should be implemented, considering the ULS capacity of a span at a time. Checks for SLS requirements would be done after the initial design.

The maximum amount of bending moment redistribution that can be achieved from a certain beam would have to be known by the designer. An equation giving the maximum amount of redistribution for a certain beam would have to be developed. This would have to include the effect of the beam cross-section shape and the prestress to reinforced steel ratio. Many examples would have to be investigated and verified experimentally in order to develop such an equation.

The computer program developed in this thesis was used to find the extent of redistribution of bending moments for a T-beam and I-beam example. It was found that the shape of the cross-section did not greatly affect the amount of redistribution, but the percentage of the central moment capacity reached was considerably less for the T-beam. For both beams it was impossible to reach the ultimate moment capacity at the centre before the strain capacity in the supports was exhausted.

The strain capacity at the supports was exhausted almost simultaneously in both the concrete and steel for the T-beam, while the steel reached its capacity long before the concrete in the case of the I-beam. The effect of replacing half of the prestress steel at the supports with an amount of reinforced steel, giving an equal section capacity, had only a small effect on the redistribution. To attain the central capacity it would therefore be necessary to place extra reinforcing steel at the supports to increase the capacity in that area. This is equivalent to reducing the planned amount of redistribution.

The effect of using design material factors was investigated and found to give less conservative results than using average material properties. This is undesirable as design values should be more conservative. Real (average) values should therefore be used for analysis.

The examples investigated ignored the effect of tension stiffening as this effect produces less ductility. The purpose of these investigations was to find the maximum expected moment redistribution, and thus, if a realistic value was required, this effect would have to be included.

A design approach using a plastic design procedure to ensure the ULS capacity of the structure on the basis of a span at a time should thus be put into practice for prestressed concrete beams. The proposed approach would correspond to the methods used in other forms of structural design, and would provide a much quicker, safer, more accurate and less costly method of design than that presently in use.

# REFERENCES


1. South African Code of Practice SABS 0100, Part I - 1980
2. Kong, F.K. and Evans, R.H., Reinforced and Prestressed Concrete, 3rd edition, Van Nostrand Reinhold (UK) Co. Ltd, London, 1987.
3. Hurst, M.K., Prestressed Concrete Design, Chapman and Hall, London, 1988.
4. Lin, T.Y., Design of Prestressed Concrete Structures, 2nd edition, John Wiley and Sons, New York, 1963
5. C.E.B. Design Manual on Cracking and Deformations, prepared by Comité Euro-International du Beton (C.E.B.), École Polytechnique Fédérale de Lausanne, Lausanne, 1983


## OTHER BOOKS USED


6. Bruggeling, A.S.G., Structural Concrete Science into Practice. In Heron vol. 32, 1987, №2.
7. Thomas, G.B., and Finney, R.L., Calculus and Analytical Geometry, Fifth edition, Addison-Wesley, Massachusetts, 1979.
8. Prestressed Concrete Design notes for the course CIV401F at the University of Cape Town, compiled by R.D. Kratz, Cape Town, 1989
9. Mollett, C.J., Ductility requirements in Prestressed Beams, B.Sc (Civ.Eng.) thesis № 11,1989

# ACKNOWLEDGEMENTS

I wish to thank my supervisor, Assoc. Prof R.D. Kratz for his assistance and guidance. He was always available for consultation and his ideas helped to make this thesis possible.

I would also like to thank the following people:

- Calvin Mollett for his experimental work
- The Laboratory assistance for their help
- Andy Scholtz of The Computer Room for his excellent typing service
- Jenni Biggs for the constant moral support.

Finally I would like to thank the CSIR for their financial assistance.

# APPENDIX A MATHEMATICAL MODELS USED IN PROGRAM.

## A.1 SIMPSON'S RULE

Simpson's Rule is a model used to solve an integral, and is used in the program to solve virtual work integrals. To find the integral of a value between x = a and x = b, the length between a and b is divided into equal intervals of width h. Each separate piece of the curve covering an x-subinterval of width 2h, is approximated by an arc of a parabola through its ends and its midpoint. The areas under the parabolic arcs are then added to give Simpson's Rule, which is:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[y_o + 4y_1 + y_2 + 4y_3 + 2y_4 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n]$$

where the y's are the values of y =f(x) at the points $x_0$ = a, $x_1$ = a+h, $x_2$ = a+2h, ... , $x_n$ = a + nh = b corresponding to a subdivision of the interval a ≤ x ≤ b into n equal subintervals of width h = (b-a)/n. The number of n must be even.

Fig. A.1 shows the division of a typical curve:

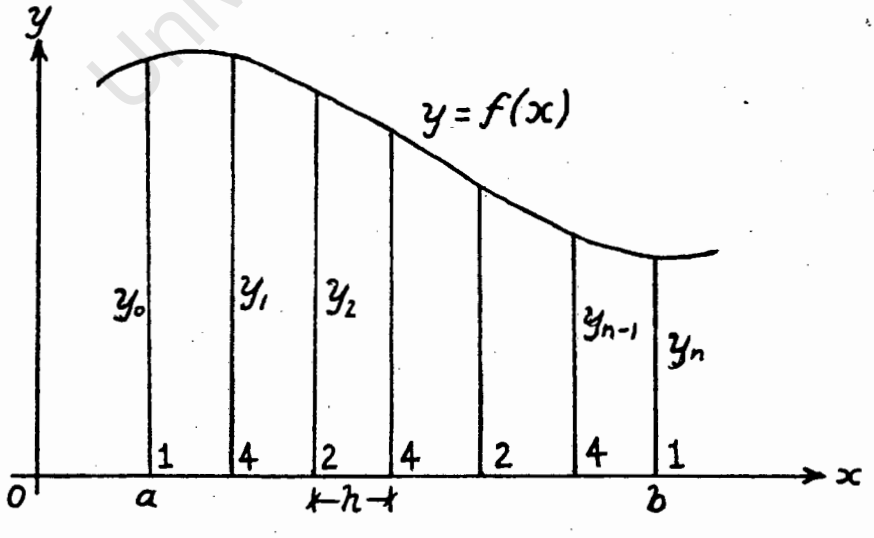



Fig A.1 Division of a curve for Simpson's Rule

In the case of a virtual work interval, where:

$$\delta = \int \phi m^* dx$$

the curves of $\phi$ and $m_*$ are multiplied together and Simpson's rule can be applied to the resulting curve.

## A.2 NEWTON-RAPHSON METHOD

The Newton-Raphson method is an iterative method used to solve high order equations. It is used often in the program. To solve a high order equation f(x) = 0, a first approximation must be guessed. The first guess will be $x_1$, and the value of $f(x_1)$ must be found. A second, more accurate approximation can then be obtained from the equation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where $f'(x_n)$ is the derivative of f at $x_n$. The second approximation can then be used to obtain a third, and so on, until an accurate enough approximation is obtained.

# APPENDIX B - PROGRAM LISTING

```
c-----------------------------------------------------------------
c     Programmer: Grant Hallam
c     Program   : DEFLEC.FOR
c     Date      : 1989
c     Purpose   : To find deflection at a point along a symmetrical
c                 beam, either a cantilever, simply supported or
c                 fully fixed. The program uses the plastic range
c                 of the beam and , for fully fixed beams, determines
c                 the amount of redistribution.
c     Language  : Fortran 77
c-----------------------------------------------------------------
c
      integer l,k
      integer icode
      character*1 inchr
c
c--- set output requirements
      open(unit=5,file='terminal')
      open(unit=6,file='printer')
c
      k = 0
      l = 0
5     continue
c--- get controlling routine
      call BEAMIN(l)
50    continue
      l = l + 1
c--- check for another example
      write(5,1004)
1004  format('Would you like another example? (y/n)')
      call KBDGET (inchr)
      icode=ichar(inchr)
      if(icode.eq.121)then
        goto 5
      else if(icode.eq.89)then
        goto 5
      else if(icode.eq.110)then
        goto 100
      else if(icode.eq.78)then
        goto 100
      else
        goto 50
      end if
100   continue
      stop
      end
```

```
      subroutine DESIGN(l,nmt,npt,ne,ms)
c-----------------------------------------------------------------------
c     Purpose:     To perform a design analysis. Given the moment and
c                  section properties, the corresponding curvature for
c                  that section will be determined.
c     Definitions:
c                  ms - moment to find corresponding curvature
c                  curv - curvature corresponding to ms
c                  ecu - strain in topmost or bottommost concrete fibre
c                  m - moment corresponding to given ecu
c                  nmt - number of material types
c                  npt - number of prestress tendons
c                  ne - number of elements in cross-section
c                  itb -     = 1 for possitive moment
c                            = 2 for negative moment
c     Details of routine: A value of .0015 is chosen for ecu and
c     the forces are balanced and the moment found using the ANALYS
c     routine. Using the Newton-Rhapson iterative method, ecu is
c     changed until the moment matches that required.
c-----------------------------------------------------------------------
c
      integer l,mat,npt,nmt,ne,k,itb,mesp,iph
      real mc,m,mur,area,yp,uml,umdxl,mdm,ai,ms
      real ei,phicr,mcr,mcri,phis,etasi
      real phi,ymax,elb,elt,h,ecu,curv
      real ecul,ecu2,ml,m2,mdxl,mcrn
      dimension etasi(10)
      dimension mc(20)
      dimension mat(10),area(10),yp(10)
      common /br/ phi,ymax,elb,elt,h,ecu
      common /bi/ etasi
      common /bp/ mat,area,yp
      common /cu/ curv
      common /m/ mc,m
      common /mur/ mur
      common /mesp/ mesp
      common /mcr/ mcr,mcrn
      common /ph/ iph
c
      mesp = 0
      if(ms.ge.0)then
        itb = 1
      else
        itb = 2
      end if
c--- check if moment within elastic range
      call CRACKM(npt,phicr,ei,mcri)
      if(itb.eq.1)then
        if(mcri.ge.0)then
          if(ms.le.mcr)then
            curv = (ms - mcri) / ei
            write(6,*) 'moment =',ms
            write(6,*) 'curvature =',curv
            goto 100
          end if
```

```
        else
          if(ms.ge.mcrn)then
            curv = (ms - mcri) / ei
            write(6,*) 'moment =',ms
            write(6,*) 'curvature =',curv
            goto 100
          end if
        end if
      else if(itb.eq.2)then
        if(mcri.lt.0)then
          if(ms.ge.mcrn)then
            curv = (ms - mcri) / ei
            write(6,*) 'moment =',ms
            write(6,*) 'curvature =',curv
            goto 100
          end if
        else
          if(ms.le.mcr)then
            curv = (ms - mcri) / ei
            write(6,*) 'moment =',ms
            write(6,*) 'curvature =',curv
            goto 100
          end if
        end if
      end if
c--- check that moment doesn't exceed section capacity
      iph = 1
      call ANALYS(nmt,npt,ne,1,4,itb,mcri)
      if(itb.eq.1)then
        if(m.lt.ms)then
          write(5,*) 'Moment at section exceeds section capacity'
          mesp = 1
          goto 100
        end if
      else if(itb.eq.2)then
        if(m.gt.ms)then
          write(5,*) 'Moment at section exceeds section capacity'
          mesp = 1
          goto 100
        end if
      end if
c--- moment within plastic range
c--- now find curvature using Newton-Rhapson iteration
      call ANALYS(nmt,npt,ne,1,2,itb,mcri)
      if(iph.eq.0)then
        ecu = ecu + .00005
        iph = 1
        goto 80
      end if
50    continue
      iph = 1
      ecu1 = ecu
      m1 = m
      if(ms-m1.le..0001)then
        if(ms-m1.ge.-.0001)then
          call ANALYS(nmt,npt,ne,1,6,itb,mcri)
          goto 100
        end if
      end if
```

```
      call ANALYS(nmt,npt,ne,1,3,itb,mcri)
      ecu2 = ecu
      m2 = m
      mdx1 = (m1 - m2)/.00001
      ecu = ecu1 - ((ms - m1)/mdx1)
      if(ecu.le.0)then
        ecu = ecu1 - .0005
      else if(ecu.gt..0035)then
        ecu = ecu1 + .0005
      end if
80    continue
      call ANALYS(nmt,npt,ne,1,5,itb,mcri)
      if(iph.eq.0)then
        ecu = ecu + .0001
        iph = 1
        goto 80
      end if
      goto 50
100   continue
      return
      end
c
      subroutine ANALYS(nmt,npt,ne,1,k,itb,mcri)
c-----------------------------------------------------------------
c     Purpose:     To perform the analysis. This routine is called
c                  by the DESIGN routine to determine the moment
c                  and curvature for a specified value of ecu.
c     Definitions: k - indicator for value of ecu
c                  mcri - moment due to prestressing only
c-----------------------------------------------------------------
c
      integer nmt,ne,npt,1,k,itb,its,iph
      real x1,fc,f,mc,m,phi,ymax,e1b,e1t,h,ecu,curv,etas,mcri
      dimension fc(20),mc(20)
      dimension etas(10)
      common /bs/ etas
      common /f/ fc,f
      common /m/ mc,m
      common /br/ phi,ymax,e1b,e1t,h,ecu
      common /cu/ curv
      common /its/ its
      common /ph/ iph
c
      if(k.eq.2)then
        goto 5
      else if(k.eq.3)then
        goto 5
      else if(k.eq.4)then
        goto 5
      else if(k.eq.5)then
        goto 5
      else if(k.eq.6)then
        goto 5
      end if
c--- get input for material types
      call MATINP(nmt,1)
c--- get input for elements
      call ELINP(ne,npt,1)
```

```
5     continue
c--- get strains in elements
      call GETSTR(ne,x1,k,itb)
c--- get strains in tendons
      call STETA(npt,x1)
c--- get stresses in elements
      call GETSIG(ne)
c--- get stresses in tendons
      call STSIG(npt)
c--- get force
      call FORCES(ne)
c--- print results
c     call VALUES(ne,nmt,npt,1)
c--- use Newton-Rhapson method to get final x for f = 0
      do 10 i = 1,20
        call DELTAX(ne,x1,npt,nmt,itb,k)
        if(iph.eq.0)then
          goto 100
        end if
c--- check if f within acceptable limits
        if(f.le.0.0001)then
          if(f.ge.-.0001)then
            goto 20
          end if
        end if
10    continue
20    continue
      if(k.eq.4)then
        goto 100
      else if(k.eq.2)then
        goto 100
      else if(k.eq.3)then
        goto 100
      else if(k.eq.5)then
        goto 100
      end if
c--- write final moment-curvature values to printer
      write(6,*) 'ecu =',ecu
      do 30 i = 1,npt
        write(6,1001) i,etas(i)
1001  format('etas(',i1,') = ',f8.6)
30    continue
c     write(6,*) 'etas =',etas(1)
      write(6,*) 'x =',x1
      write(6,*) 'm =',m
      if(itb.eq.1)then
        curv = ecu / x1
      else if(itb.eq.2)then
        curv = -ecu / (h - x1)
      end if
      write(6,*) 'curvature =',curv
c--- if tension stiffening applies
      if(its.eq.1)then
        call TENSTIF(itb,npt,mcri)
      end if
100   continue
      return
      end
```

```
      subroutine TENSTIF(itb,npt,mcri)
c-----------------------------------------------------------------------
c     Purpose:     To correct the curvature value for tension stiffening
c-----------------------------------------------------------------------
c
      integer itb,mat,npt,matype
      real m,ecu,etas,etasi,mcr,mcrn,h,ymax,etas2,etasm,ybar
      real etas1,etac1,etac2,etacm,seclen,curv,area,yp,pi,iy,ei
      real ec,ic,fctm,y,b,d,var1,var2,var3,l1,l2,pit,mcri,art
      dimension etas(10),mc(20)
      dimension etasi(10),area(10),yp(10),mat(10)
      dimension pi(10)
      dimension y(20),b(20),d(20)
      dimension eltype(20),matype(20)
      dimension ec(5),fctm(5)
      common /b/ eltype,matype,y,b,d
      common /bi/ etasi
      common /bs/ etas
      common /br/ phi,ymax,e1b,e1t,h,ecu
      common /bp/ mat,area,yp
      common /cu/ curv
      common /m/ mc,m
      common /mcr/ mcr,mcrn
      common /sec/ seclen
      common /pi/ pi
      common /cent/ ybar,iy,art
      common /ef/ ec,fctm
c
      pit = 0
      do 20 i = 1,npt
        pit = pi(i) + pit
20    continue
      ei = ec(matype(1)) * iy
      if(itb.eq.1)then
        l1 = ( ( mcr / m ) ** 2 ) * seclen
        l2 = ( 1 - ( (mcr / m) ** 2 ) ) * seclen
        var1 = pit / ( art * ec(matype(1)) )
        var2 = ( (m - mcri) / ei ) * ( ybar - yp(1) )
        etas1 = var1 - var2
        etas2 = etasi(1) + etas(1)
        etasm = ((l1 * etas1) + (l2 * etas2)) / seclen
        var3 = ( (m - mcri) / ei ) * ( ymax - ybar )
        etac1 = var1 + var3
        etac2 = ecu
        etacm = ((l1 * etac1) + (l2 * etac2)) / seclen
        curv = (etacm - etasm) / (ymax - yp(1))
c--- write moment-curvature values to printer
        write(6,*) 'length of section =',seclen
        write(6,*) 'ec =',etacm
        write(6,*) 'es =',etasm
        write(6,*) 'm =',m
        write(6,*) 'curvature =',curv
      else
        l1 = ( ( mcrn / m ) ** 2 ) * seclen
        l2 = ( 1 - ( (mcrn / m) ** 2 ) ) * seclen
        var1 = pit / ( art * ec(matype(1)) )
        var2 = ( (m - mcri) / ei ) * ( ybar - yp(1) )
```

```
        etas1 = var1 - var2
        etas2 = etasi(1) + etas(1)
        etasm = ((l1 * etas1) + (l2 * etas2)) / seclen
        var3 = ( (m - mcri) / ei ) * ( ybar - y(1) )
        etac1 = var1 - var3
        etac2 = ecu
        etacm = ((l1 * etac1) + (l2 * etac2)) / seclen
        curv = -(etacm - etasm) / (yp(1) - y(1))
c--- write moment-curvature values to printer
        write(6,*) 'length of section =',seclen
        write(6,*) 'ec =',etacm
        write(6,*) 'es =',etasm
        write(6,*) 'm =',m
        write(6,*) 'curvature =',curv
      end if
      return
      end
c
      subroutine CRACKM(npt,phicr,ei,mcri)
c-----------------------------------------------------------------------
c     Purpose    : To find the cracking moment and curvature
c     Definitions: mcr - positive cracking moment
c                  mcrn - negative cracking moment
c                  phicr - positive cracking curvature
c                  phicrn - negative cracking curvature
c                  etact - cracking strain in concrete
c-----------------------------------------------------------------------
c
      integer mat,npt,eltype,matype
      real ic,fctm,etasi,ei,etact,fpi,pi,area,yp,ec,es
      real sigpi,etaci,etci,b,d,phicr,mcr,mcri,mcra,mci
      real phicrn,mcrn,ybar,iy,art,ymax,mcran
      dimension etasi(10),area(10),yp(10),mat(10)
      dimension y(20),b(20),d(20)
      dimension eltype(20),matype(20)
      dimension fpi(10),pi(10),sigpi(10),etaci(10),mci(10)
      dimension ec(5),fctm(5)
      common /b/ eltype,matype,y,b,d
      common /bi/ etasi
      common /bp/ mat,area,yp
      common /ef/ ec,fctm
      common /mcr/ mcr,mcrn
      common /pi/ pi
      common /cent/ ybar,iy,art
      common /br/ phi,ymax,e1b,e1t,h,ecu
c
      etact = fctm(matype(1))/ec(matype(1))
      ei = ec(matype(1)) * iy
      etci = 0
      mcri = 0
      do 10 i = 1,npt
        fpi(i) = etasi(i) * ec(mat(i))
        pi(i) = fpi(i) * area(i)
        sigpi(i) = pi(i) / art
        etaci(i) = sigpi(i) / ec(matype(1))
        etci = etaci(i) + etci
        mci(i) = (ybar - yp(i)) * pi(i)
        mcri = mci(i) + mcri
10    continue
```

```
      phicr = (etci + etact) / (ybar - y(1))
      phicrn = (etci + etact) / (ybar - ymax)
      mcra = ei * phicr
      mcran = ei * phicrn
      mcr = mcri + mcra
      mcrn = mcri + mcran
      return
      end
c
      subroutine SECPROP(ne)
c-----------------------------------------------------------------------
c     Purpose    : To find the centroidal axis and moment of inertia
c                  for the given cross-section.
c     Definitions: iy - moment of inertia (I)
c                  ybar - distance to centroidal axis
c-----------------------------------------------------------------------
c
      integer ne,eltype,matype
      real yb,ar,iyi,y,b,d,ybb,ybt,iy,ybar,art
      dimension eltype(20),matype(20)
      dimension y(20),b(20),d(20)
      dimension yb(20),ar(20),iyi(20)
      common /b/ eltype,matype,y,b,d
      common /cent/ ybar,iy,art
c
      ybt = 0
      ybb = 0
      art = 0
c--- find section properties of individual elements
      do 10 i = 1,ne
        if(eltype(i).eq.1)then
          yb(i) = y(i) + (d(i)/2)
          ar(i) = b(i) * d(i)
          iyi(i) = b(i) * (d(i)**3) / 12
        else if(eltype(i).eq.2)then
          yb(i) = y(i) + (d(i)/3)
          ar(i) = b(i) * d(i) / 2
          iyi(i) = b(i) * (d(i)**3) / 36
        else
          yb(i) = y(i) + (2*d(i)/3)
          ar(i) = b(i) * d(i) / 2
          iyi(i) = b(i) * (d(i)**3) / 36
        end if
        ybt = (ar(i) * yb(i)) + ybt
        ybb = ar(i) + ybb
c--- find the total area of the x-section
        art = ar(i) + art
10    continue
c--- find distance to centroidal axis of entire x-section
      ybar = ybt / ybb
c--- find moment of inertia (I) of entire x-section
      iy = 0
      do 20 i = 1,ne
        iy = iyi(i) + (ar(i) * ((ybar - yb(i))**2)) + iy
20    continue
      return
      end
```

```
1005  format('Do you wish to (a) change all material properties,')
1006  format('                (b) change some material properties,')
1007  format('             or (c) leave all properties as is?')
        call KBDGET (inchr)
        icode = ichar(inchr)
        if(icode.eq.97)then
          goto 15
        else if(icode.eq.65)then
          goto 15
        else if(icode.eq.98)then
          k = 2
          goto 35
        else if(icode.eq.66)then
          k = 2
          goto 35
        else if(icode.eq.99)then
          goto 100
        else if(icode.eq.67)then
          goto 100
        else
          goto 5
        end if
      end if
c--- for own material data...
15    continue
c--- get number of material types
      write(5,1001)
1001  format('Enter number of material types')
      read(*,*) nmt
35    continue
c--- get details of material types
      do 10 i = 1, nmt
        if(k.eq.2)then
45        continue
          write(5,1008) i
1008  format('Change values for material type',i2,'(y/n)?')
          call KBDGET(inchr)
          icode = ichar(inchr)
          if(icode.eq.121)then
            goto 55
          else if(icode.eq.89)then
            goto 55
          else if(icode.eq.110)then
            goto 65
          else if(icode.eq.78)then
            goto 65
          else
            goto 45
          end if
        end if
55      continue
        write(5,1002) i
        read(*,*) np(i)
        do 20 j = 1, np(i)
          write(5,1003) j
          read(*,*) strain(i,j)
          write(5,1004) j
          read(*,*) stress(i,j)
20      continue
```

```
65      continue
10    continue
1002  format('Enter number of points to define material type ',i1)
1003  format('Enter strain of point ',i1)
1004  format('Enter stress of point ',i1)
100   continue
      return
      end
c
c
      subroutine ELINP(ne,npt,l)
c-----------------------------------------------------------------------
c     Purpose    : To input information about elements.
c                  Called by ANALYS or BEAMIN.
c     Define element types: - type 1 - rectangle
c                           - type 2 - triangle pointing upward
c                           - type 3 - triangle pointing downward
c     Definitions: eltype(i) - no. of element type for element i
c                  matype(i) - no. of material type for element i
c                  y(i),b(i),d(i) - distance from base to lowest point,
c                                   bredth and depth respectively for
c                                   element i
c                  ne - no. of element types
c     Note : Elements should be numbered in such a way that the first
c            element should be that with the lowest base and the last
c            should be that with the highest top.
c-----------------------------------------------------------------------
c
      integer i,j,ne,npt,l
      integer eltype,matype,mat
      real y,b,d,area,yp,mur,ymax
      integer icode
      character*1 inchr
      dimension eltype(20),matype(20),mat(10),area(10),yp(10)
      dimension y(20),b(20),d(20)
      common /b/ eltype,matype,y,b,d
      common /bp/ mat,area,yp
      common /mur/ mur
      common /br/ phi,ymax,e1b,e1t,h,ecu
c
c--- for second run
      if(l.ne.0)then
5       continue
        write(5,1013)
        write(5,1017)
        write(5,1018)
1013  format('Do you wish to change (a) element properties')
1017  format('                      (b) tendon properties')
1018  format('                  or (c) leave all values?')
        call KBDGET (inchr)
        icode = ichar(inchr)
        if(icode.eq.97)then
          goto 15
        else if(icode.eq.65)then
          goto 15
        else if(icode.eq.98)then
          goto 25
        else if(icode.eq.66)then
          goto 25
```

```
        else if(icode.eq.99)then
          goto 100
        else if(icode.eq.67)then
          goto 100
        else
          goto 5
        end if
      end if
15    continue
c--- get details of element types
      write(5,1001)
1001  format(/,'Enter number of elements')
      read(*,*) ne
      do 10 i = 1, ne
        write(5,1002) i
        write(5,1003)
        read(*,*) eltype(i)
        write(5,1004)
        read(*,*) matype(i)
        write(5,1005)
        read(*,*) y(i)
        write(5,1006)
        read(*,*) b(i)
        write(5,1007)
        read(*,*) d(i)
10    continue
      ymax = y(ne) + d(ne)
      call SECPROP(ne)
1002  format('For element ',i2)
1003  format('...enter number for element type')
1004  format('...enter number for material type')
1005  format('...enter distance from axis to lowest point')
1006  format('...enter bredth of element')
1007  format('...enter height of element')
25    continue
c--- get details of prestress tendons
      write(5,1008)
1008  format(/,'Enter number of prestress tendons')
      read(*,*) npt
      do 20 i = 1,npt
        write(5,1009) i
        write(5,1010)
        read(*,*) mat(i)
        write(5,1011)
        read(*,*) area(i)
20    continue
1009  format('For tendon ',i2)
1010  format('...enter number for material type')
1011  format('...enter area of steel')
1012  format('...enter distance from base axis to centre')
1016  format('For tendon...')
100   continue
      return
      end
```

```
      subroutine GETSTR(ne,x1,k,itb)
c-----------------------------------------------------------------------
c     Purpose    : To find the strains in the concrete. A value for ecu
c                  is defined and x1 is initially chosen as half height.
c                  This (x1) is changed by the other routines using the
c                  Newton-Rhapson iteration. This routine calculates the
c                  curvature for the specified ecu and x1 and calls the
c                  routine GETETA to find the strains in the concrete.
c     Definitions: x1 - distance from top fibre to zero strain,
c                       set initially to half height
c                  h - height of section
c                  phi - curvature
c-----------------------------------------------------------------------
c
      integer ne,k,itb
      real etat,etam,etab
      real y,b,d,phi,x1,e1t,e1b,h,ymax,ecu
      integer eltype,matype
      dimension eltype(20),matype(20)
      dimension y(20),b(20),d(20)
      dimension etat(20),etam(20),etab(20)
      common /b/ eltype,matype,y,b,d
      common /c/ etat,etam,etab
      common /br/ phi,ymax,e1b,e1t,h,ecu
c
c--- for Newton-Rhapson iteration
      if(k.eq.3)then
        ecu = ecu + .00001
      else if(k.eq.1)then
c--- get strain in top fibre
        write(5,1001)
1001  format(/,'Enter strain in top fibre')
        read(*,*) ecu
c--- set ecu for first run
      else if(k.eq.2)then
        ecu = .0015
c--- set ecu = .0035 to find max section moment
      else if(k.eq.4)then
        ecu = .0035
      end if
      e1t = 0
      e1b = 0
50    continue
c--- set first x value to half depth of section
      h = ymax - y(1)
      x1 = h/2
c--- calculate angle phi
      if(itb.eq.1)then
        phi = ecu / x1
      else if(itb.eq.2)then
        phi = -ecu / (h - x1)
      end if
c--- find strains at top, middle and bottom
      call GETETA(x1,ne,phi,ymax,e1t,e1b,h)
      return
      end
```

```
      subroutine GETETA(x,ne,phi,ymax,elt,elb,h)
c-----------------------------------------------------------------------
c     Purpose    : To find strains at top, middle and bottom of each
c                  eliment in the cross-section.
c     Definitions: etat,etam,etab - the top, middle and bottom strains
c                                   for each elament
c-----------------------------------------------------------------------
c
      integer i,ne,eltype,matype
      real x,phi,ymax,yt,d,etat,etam,etab
      real elt,elb,h,etalt,etalm,etalb,y,b
      dimension eltype(20),matype(20),yt(20),etat(20),etam(20),etab(20)
      dimension y(20),b(20),d(20)
      common /b/ eltype,matype,y,b,d
      common /c/ etat,etam,etab
      common /d/ yt
c
c--- find strains for each element
      do 10 i = 1,ne
        yt(i) = y(i) + d(i)
        z = ymax - yt(i)
        etalt = elt + ((elb - elt)*(ymax - yt(i))/h)
        etat(i) = (phi * (x - z)) + etalt
        etalm = elt + ((elb - elt)*(ymax - yt(i) + (d(i)/2))/h)
        etam(i) = phi * (x - z - (d(i)/2)) + etalm
        etalb = elt + ((elb - elt)*(ymax - y(i))/h)
        etab(i) = phi * (x - (ymax - y(i))) + etalb
10    continue
      return
      end
c
      subroutine STETA(npt,x)
c-----------------------------------------------------------------------
c     Purpose    : To find the strain in the prestress tendons.
c     Definitions: etast - steel strains due to applied moment
c                  etas - total steel strain inc. prestrain
c-----------------------------------------------------------------------
      integer npt,mat,i,eltype,matype
      real area,yp,etasi,z,etast,etas,y,b,d,phi,x1,ymax,x,h
      dimension y(20),b(20),d(20),eltype(20),matype(20)
      dimension mat(10),area(10),yp(10),etasi(10),etast(10),etas(10)
      common /bp/ mat,area,yp
      common /b/ eltype,matype,y,b,d
      common /br/ phi,ymax,elb,elt,h,ecu
      common /bs/ etas
      common /bi/ etasi

5     continue
      do 10 i = 1,npt
        z = ymax - yp(i)
        etast(i) =  phi * (x - z)
        if(etast(i).lt.-.01)then
          write(5,*) 'for tendon',i
          write(5,*) 'steel strain is greater than 1%'
        end if
        etas(i) = etast(i) - etasi(i)
10    continue
      return
      end
```

```
      subroutine STSIG(npt)
c-----------------------------------------------------------------
c     Purpose    : To get the stresses in the tendons.
c     Definitions: fst - force in the tendon
c                  mst - moment due to tendon force
c                  sigs - stress in steel
c                  yp - distance from base to tendon
c-----------------------------------------------------------------
c
      integer npt,mat
      real etas,area,yp,sigs,fst,fcst,mcst,mst
      dimension etas(10),sigs(10),fcst(10),mcst(10)
      dimension mat(10),area(10),yp(10)
      common /bp/ mat,area,yp
      common /bs/ etas
      common /bg/ sigs
      common /fs/ fcst,fst
      common /ms/ mcst,mst
c
      fst = 0
      mst = 0
      do 10 i = 1,npt
        call CONVRT(etas(i),mat(i),sigs(i))
        fcst(i) = sigs(i) * area(i)
        fst = fst + fcst(i)
        mcst(i) = sigs(i) * area(i) * yp(i)
        mst = mst + mcst(i)
10    continue
      return
      end
c
      subroutine GETSIG(ne)
c-----------------------------------------------------------------
c     Purpose    : To get the stresses from the strains in the concrete.
c     Definitions: sigt,sigm,sigb - stresses in concrete at top, middle
c                                   and bottom
c-----------------------------------------------------------------
c
      integer eltype,matype,ne,i,j,k,np,iw,itw
      real etat,etam,etab,v,w,z,etaw,etawm,vm
      real sigt,sigm,sigb,sigw
      real stress,strain
      real y,b,d,yt
      dimension np(5)
      dimension stress(5,10),strain(5,10)
      dimension eltype(20),matype(20)
      dimension y(20),b(20),d(20),yt(20)
      dimension etat(20),etam(20),etab(20)
      dimension sigt(20),sigm(20),sigb(20)
      common /a/ np,stress,strain
      common /b/ eltype,matype,y,b,d
      common /c/ etat,etam,etab
      common /d/ yt
      common /e/ sigt,sigm,sigb
      common /ew/ sigw,iw,v,sigwm,vm,itw
c
      sigw = 0
      iw = 0
      itw = 0
```

```
c--- for top strain larger than bottom strain
      if(etat(ne).ge.etab(1))then
        do 10, i = 1,ne
c--- for stresses below limit
          if(etat(i).le.strain(matype(i),1))then
            sigt(i) = 0
            sigm(i) = 0
            sigb(i) = 0
c--- for stresses within given limits
          else if(etab(i).ge.strain(matype(i),1))then
            call CONVRT(etat(i),matype(i),sigt(i))
            call CONVRT(etam(i),matype(i),sigm(i))
            call CONVRT(etab(i),matype(i),sigb(i))
c--- for bottom stress below limit
c--- must find point of zero strain
          else
            z = (etat(ne)-etab(1))/(etat(ne)-strain(matype(i),1))
            w = (yt(ne) - y(1))/z
            v = yt(ne) - w
            iw = i
            itw = 1
            sigw = stress(matype(i),1)
            etaw = strain(matype(i),1)
            etawm = etaw + ((etat(i) - etaw)/2)
            call CONVRT(etawm,matype(i),sigwm)
            vm = v + ((yt(i) - v)/2)
            call CONVRT(etat(i),matype(i),sigt(i))
            sigb(i) = 0
            if(etam(i).lt.strain(matype(i),1))then
              sigm(i) = 0
            else
              call CONVRT(etam(i),matype(i),sigm(i))
            end if
          end if
10      continue
c--- for top strain less than bottom strain
       else
        do 20, i = 1,ne
c--- for stresses below limit
          if(etab(i).le.strain(matype(i),1))then
            sigt(i) = 0
            sigm(i) = 0
            sigb(i) = 0
c--- for stresses within limits
          else if(etat(i).ge.strain(matype(i),1))then
            call CONVRT(etat(i),matype(i),sigt(i))
            call CONVRT(etam(i),matype(i),sigm(i))
            call CONVRT(etab(i),matype(i),sigb(i))
c--- for top stress below limit
c--- must find point of zero strain
          else
            z = (etab(1)-etat(ne))/(etab(1)-strain(matype(i),1))
            w = (yt(ne) - y(1))*(z - 1)/z
            v = yt(ne) - w
            iw = i
            itw = 2
            sigw = stress(matype(i),1)
            etaw = strain(matype(i),1)
            etawm = etaw + ((etab(i) - etaw)/2)
```

```
            call CONVRT(etawm,matype(i),sigwm)
            vm = y(i) + ((v - y(i))/2)
            call CONVRT(etab(i),matype(i),sigb(i))
            sigt(i) = 0
            if(etam(i).lt.strain(matype(i),1))then
              sigm(i) = 0
            else
              call CONVRT(etam(i),matype(i),sigm(i))
            end if
          end if
20      continue
      end if
      return
      end
c
      subroutine CONVRT(str,mt,sig)
c-----------------------------------------------------------------------
c     Purpose    : To convert strains to stresses by interpolation
c                  from property stress-strain values given above.
c-----------------------------------------------------------------------
c
      integer np,mt
      real stress,strain,p1,p2,r1,r2,s,sig,str
      dimension np(5)
      dimension stress(5,10),strain(5,10)
      common /a/ np,stress,strain
c
      do 10 i = 1,np(mt)
        if(i+1.eq.np(mt))then
c--- check if strain not too big
          if(str.gt.strain(mt,np(mt)))then
            str = strain(mt,np(mt))
            write(5,*) 'WARNING -Strain exceeds max value'
            write(5,*) '         Value has been changed'
          end if
        end if
        if(str.le.strain(mt,i+1))then
          p1 = strain(mt,i+1)
          r1 = strain(mt,i)
          s = (str - r1)/(p1 - r1)
          p2 = stress(mt,i+1)
          r2 = stress(mt,i)
          sig = r2 + (s*(p2 - r2))
          goto 100
        end if
10    continue
100   continue
      return
      end
c
```

```
      subroutine FORCES(ne)
c-----------------------------------------------------------------------
c     Purpose    : To determine the forces on the section.
c     Definitions: brt,brm,brb - bredths of element at top, middle and
c                                bottom
c                  fct,fcm,fcb - forces per unit length at top, middle
c                                and bottom
c                  mct,mcm,mcb - moments due to above forces
c                  fc,mc - force on each element and its respective
c                          moment
c                  f,m - total force and moment on section from concrete
c                        and steel
c     Details of routine: This routine uses Simpson's rule to find the
c     total force and moment on the section. It first multiplies the
c     concrete stresses at top, middle and bottom of each element by the
c     bredth of the section at that point, and then uses Simpson's rule
c     to find the total force and moment on each section. These forces
c     and moments are added together, along with the steel forces and
c     moments to obtain the total effect on the cross-section.
c-----------------------------------------------------------------------
c
      integer eltype,matype,ne,iw,itw
      real y,b,d,m,mcst,mc,mct,mcm,mcb,yt,mst
      real sigt,sigm,sigb
      real brt,brm,brb,fct,fcm,fcb,fc,f,fcst,fst,fw
      real sigw,v,sigwm,vm,mw
      dimension y(20),b(20),d(20),yt(20)
      dimension eltype(20),matype(20)
      dimension sigt(20),sigm(20),sigb(20)
      dimension brt(20),brm(20),brb(20)
      dimension fct(20),fcm(20),fcb(20),fc(20)
      dimension fcst(10)
      dimension mcst(10),mc(20),mct(20),mcm(20),mcb(20)
      common /b/ eltype,matype,y,b,d
      common /d/ yt
      common /e/ sigt,sigm,sigb
      common /ew/ sigw,iw,v,sigwm,vm,itw
      common /f/ fc,f
      common /fs/ fcst,fst
      common /m/ mc,m
      common /ms/ mcst,mst
c
      f = 0
      m = 0
      fw = 0
      mw = 0
      do 10 i = 1,ne
c--- check if stresses start above base
        if(iw.ne.0)then
          if(i.eq.iw)then
            call FORCEW(i,fw,mw)
            fc(i) = fw
            mc(i) = mw
            goto 50
          end if
        end if
```

```
c--- find bredths at top, bottom and middle of each section
        if(eltype(i).eq.1)then
          brt(i) = b(i)
          brm(i) = b(i)
          brb(i) = b(i)
        else if(eltype(i).eq.2)then
          brt(i) = 0
          brm(i) = b(i)/2
          brb(i) = b(i)
        else
          brt(i) = b(i)
          brm(i) = b(i)/2
          brb(i) = 0
        end if
c--- find multiple of bredth and stress
        fct(i) = brt(i) * sigt(i)
        fcm(i) = brm(i) * sigm(i)
        fcb(i) = brb(i) * sigb(i)
c--- find multiple of bredth,stress and dist. from base
        mct(i) = fct(i) * yt(i)
        mcm(i) = fcm(i) * (y(i) + ((yt(i) - y(i))/2))
        mcb(i) = fcb(i) * y(i)
c--- apply Simpson's rule to find force and moment
        fc(i) = d(i)*(fct(i) + fcb(i) + (4*fcm(i)))/6
        mc(i) = d(i)*(mct(i) + mcb(i) + (4*mcm(i)))/6
50      continue
c--- add forces consecutively
        f = f + fc(i)
        m = m + mc(i)
10    continue
      f = f + fst
      m = m + mst
      return
      end
c
      subroutine FORCEW(i,fw,mw)
c-----------------------------------------------------------------------
c     Purpose    : To find force in element iw where that element is
c                  such that the stress is zero at some point within it.
c-----------------------------------------------------------------------
c
      integer iw,itw
      integer eltype,matype
      real bb,bt,bw,bm,sbft,sbfb,sbfm,sbfw
      real sbmt,sbmb,sbmm,sbmw
      real y,b,d
      real sigw,v,sigwm,vm,mw,fw
      real sigt,sigm,sigb,yt
      dimension sigt(20),sigm(20),sigb(20)
      dimension y(20),b(20),d(20)
      dimension eltype(20),matype(20)
      dimension yt(20)
      common /b/ eltype,matype,y,b,d
      common /d/ yt
      common /ew/ sigw,iw,v,sigwm,vm,itw
      common /e/ sigt,sigm,sigb
c
```

```
if(itw.eq.1)then
  if(eltype(i).eq.1)then
    bt = b(i)
    bm = b(i)
    bw = b(i)
  else if(eltype(i).eq.2)then
    bt = 0
    bm = b(i) * ((yt(i) - vm)/(yt(i) - y(i)))
    bw = b(i) * ((yt(i) - v)/(yt(i) - y(i)))
  else
    bt = b(i)
    bm = b(i) * ((vm - y(i))/(yt(i) - y(i)))
    bw = b(i) * ((v - y(i))/(yt(i) - y(i)))
  end if
  sbft = bt * sigt(i)
  sbfm = bm * sigwm
  sbfw = bw * sigw
  fw = (yt(i) - v) * (sbft + sbfw + (4*sbfm))/6
  sbmt = sbft * yt(i)
  sbmm = sbfm * vm
  sbmw = sbfw * v
  mw = (yt(i) - v) * (sbmt + sbmw + (4*sbmm))/6
else
  if(eltype(i).eq.1)then
    bb = b(i)
    bm = b(i)
    bw = b(i)
  else if(eltype(i).eq.2)then
    bb = b(i)
    bm = b(i) * ((yt(i) - vm)/(yt(i) - y(i)))
    bw = b(i) * ((yt(i) - v)/(yt(i) - y(i)))
  else
    bb = 0
    bm = b(i) * ((vm - y(i))/(yt(i) - y(i)))
    bw = b(i) * ((v - y(i))/(yt(i) - y(i)))
  end if
  sbfb = bb * sigb(i)
  sbfm = bm * sigwm
  sbfw = bw * sigw
  fw = (v - y(i)) * (sbfb + sbfw + (4*sbfm))/6
  sbmb = sbfb * y(i)
  sbmm = sbfm * vm
  sbmw = sbfw * v
  mw = (v - y(i)) * (sbmb + sbmw + (4*sbmm))/6
end if
return
end
```

```
      subroutine DELTAX(ne,x1,npt,nmt,itb,k)
c-----------------------------------------------------------------------
c     Purpose    : To find force for f(x+dx), which is part of the
c                  iterative process. x + dx refers to the distance from
c                  the top fibre to the point of zero strain. In this
c                  iteration x is the variable in finding the required
c                  force.
c-----------------------------------------------------------------------
c
      integer ne,npt,nmt,mat,itb
      real phi,x1,ymax,e1b,e1t,f1,h,m1
      real fc,f,xdx,etasi,etas,area,yp,mc,m
      dimension fc(20),mc(20)
      dimension etasi(10),etas(10),etast(10)
      dimension mat(10),area(10),yp(10)
      common /bp/ mat,area,yp
      common /br/ phi,ymax,e1b,e1t,h,ecu
      common /bi/ etasi
      common /bs/ etas
      common /f/ fc,f
      common /fi/ f1
      common /m/ mc,m
      common /mi/ m1
c
      f1 = f
      m1 = m
      f = 0
      m = 0
c--- set x to x + dx
      xdx = x1 + .01
c--- calculate new phi
      if(itb.eq.1)then
        phi = ecu / xdx
      else if(itb.eq.2)then
        phi = -ecu / (h - xdx)
      end if
      call GETETA(xdx,ne,phi,ymax,e1t,e1b,h)
c--- find strains in tendons
      do 10 i = 1,npt
        z = ymax - yp(i)
        etast(i) = phi * (xdx - z)
        etas(i) = etast(i) - etasi(i)
10    continue
      call GETSIG(ne)
      call STSIG(npt)
      call FORCES(ne)
      call TEST2(ne,x1,npt,nmt,itb,k)
      return
      end
```

```
      subroutine TEST2(ne,x1,npt,nmt,itb,k)
c-----------------------------------------------------------------------
c     Purpose    : To find force for f(x2), where x2 is the next value
c                  for x in the iteration.
c-----------------------------------------------------------------------
      integer ne,npt,nmt,mat,itb,iph
      real phi,x1,ymax,e1b,e1t,f1,fdx,x2
      real fc,f,xdx,etasi,etas,area,yp,h,mc,m,m1
      dimension fc(20),mc(20),etasi(10),etas(10),etast(10)
      dimension mat(10),area(10),yp(10)
      common /bp/ mat,area,yp
      common /br/ phi,ymax,e1b,e1t,h,ecu
      common /bi/ etasi
      common /bs/ etas
      common /f/ fc,f
      common /fi/ f1
      common /m/ mc,m
      common /mi/ m1
      common /ph/ iph
c
c--- set x to x2
      fdx = (f - f1) / .01
      if(fdx.eq.0)then
        iph = 0
        goto 100
      else
        x2 = x1 - (f1 / fdx)
      end if
      if(itb.eq.2)then
        if(x2.ge.ymax)then
          x2 = 1.25 * x1
        end if
      else
        if(x2.le.0)then
          x2 = .75 * x1
        end if
      end if
c     end if
c--- calculate new phi
      if(itb.eq.1)then
        phi = ecu / x2
      else if(itb.eq.2)then
        phi = -ecu / (h - x2)
      end if
      call GETETA(x2,ne,phi,ymax,e1t,e1b,h)
c--- find strains in tendons
      do 10 i = 1,npt
        z = ymax - yp(i)
        etast(i) = phi * (x2 - z)
        etas(i) = etast(i) - etasi(i)
10    continue
      f = 0
      m = 0
      call GETSIG(ne)
      call STSIG(npt)
      call FORCES(ne)
      x1 = x2
100   continue
      return
      end
```

```
      subroutine BEAMIN(l)
c-----------------------------------------------------------------------
c     Purpose    : To specify the type of beam and call the controlling
c                  routine for that beam type.
c-----------------------------------------------------------------------
c
      integer icode,l
      character*1 inchr
c
5     continue
c--- choose type of beam
      write(5,1010)
      write(5,1011)
      write(5,1012)
1010  format('Is the beam (a) a cantilever,')
1011  format('            (b) simply supported,')
1012  format('         or (c) fully fixed?')
      call KBDGET (inchr)
      icode = ichar(inchr)
      if(icode.eq.97)then
        write(5,*) 'Cantilever chosen'
        call CANTIL(l,ne,npt)
      else if(icode.eq.65)then
        write(5,*) 'Cantilever chosen'
        call CANTIL(l,ne,npt)
      else if(icode.eq.98)then
        write(5,*) 'Simply suported beam chosen'
        call SIMSUP(l,ne,npt)
      else if(icode.eq.66)then
        write(5,*) 'Simply suported beam chosen'
        call SIMSUP(l,ne,npt)
      else if(icode.eq.99)then
        write(5,*) 'Fully fixed beam chosen'
        call FULFIX(l,ne,npt)
      else if(icode.eq.67)then
        write(5,*) 'Fully fixed beam chosen'
        call FULFIX(l,ne,npt)
      else
        goto 5
      end if
      return
      end
```

```
      subroutine CANTIL(l,ne,npt)
c-----------------------------------------------------------------------
c     Purpose    : To find the deflection at the end of a prestressed
c                  cantilever with specified load, dead load, tendon
c                  profile and beam dimensions.
c     Definitions: its - code for tension stiffening  1 - yes
c                                                     2 - no
c                  defle,deflp - end deflections due to the elastic and
c                                plastic zones respectively
c                  defl - total end deflection
c                  x1 - distance from support to end of plastic zone
c-----------------------------------------------------------------------
c
      integer l,nmt,npt,ne,i,mat,ip,mesp,ilt
      integer icode,its
      character*1 inchr
      real leng,dtp,dl,ll,etasi,yp,area,mmax,curv,lm
      real mcr,mcrn,mcri,ei,phicr,ms,x1,xp,xq,xr,xn,seclen
      real cup,jp,dp,cu,jj,d,xm,defle,deflp,defl,simp1,simp2
      dimension etasi(10),area(10),yp(10),mat(10)
      dimension dtp(10)
      dimension  cu(10),jj(10),cup(10),jp(10)
      common /bi/ etasi
      common /bp/ mat,area,yp
      common /cu/ curv
      common /mesp/ mesp
      common /mcr/ mcr,mcrn
      common /sec/ seclen
      common /its/ its
      common /cant/ leng,dtp,dl
c
5     continue
c--- check whether or not tension stiffening is required
      write(5,1030)
1030  format('Should tension stiffening be concidered? (y/n)')
      call KBDGET (inchr)
      icode=ichar(inchr)
      if(icode.eq.121)then
        its = 1
      else if(icode.eq.89)then
        its = 1
      else if(icode.eq.110)then
        its = 2
      else if(icode.eq.78)then
        its = 2
      else
        goto 5
      end if
      ip = 0
      seclen = 0
c--- get material types
      call MATINP(nmt,l)
c--- get cross-sectional values
      call ELINP(ne,npt,l)
```

```
c--- for second run, check to change values
      if(l.ne.0)then
7       continue
        write(5,1061)
        write(5,1063)
1061  format('Do you wish to (a) change other beam properties,')
1063  format('            or (b) leave all beam properties as is?')
        call KBDGET (inchr)
        icode = ichar(inchr)
        if(icode.eq.97)then
          goto 8
        else if(icode.eq.65)then
          goto 8
        else if(icode.eq.98)then
          goto 25
        else if(icode.eq.66)then
          goto 25
        else
          goto 7
        end if
      end if
8     continue
c--- get initial prestrain in tendons
      do 10 i = 1,npt
        write(5,1001) i
        write(5,1002)
1001  format('For prestress tendon ',i2)
1002  format('...enter initial prestrain')
        read(*,*) etasi(i)
10    continue
c--- get input for beam dimension
      write(5,1003)
      read(*,*) leng
1003  format('Enter length of cantilever')
      do 20 i = 1,npt
        write(5,1001) i
        write(5,1005)
        read(*,*) dtp(i)
1005  format('...enter height from base axis to cable.')
20    continue
c--- get loading values
      write(5,1006)
      read(*,*) dl
1006  format('Enter dead load (Mn/m)')
25    continue
      write(5,1008)
      write(5,1009)
c--- check for end load or moment
1008  format('Enter (a) for end load or')
1009  format('      (b) for end moment')
      call KBDGET (inchr)
      icode = ichar(inchr)
      if(icode.eq.97)then
        ilt = 1
      else if(icode.eq.65)then
        ilt = 1
      else if(icode.eq.98)then
        ilt = 2
```

```
      else if(icode.eq.66)then
        ilt = 2
      else
        goto 25
      end if
      if(ilt.eq.1)then
        write(5,1007)
        read(*,*) ll
      else if(ilt.eq.2)then
        write(5,1010)
        read(*,*) lm
      end if
1007  format('Enter live load at end (Mn)')
1010  format('Enter live moment at end (Mnm) (-ve moment -ve)')
c--- find if plastic hinge exists
c--- get max negative moment
      if(ilt.eq.1)then
        mmax = -((ll * leng) + (dl * (leng **2) /2))
      else
        mmax = lm
      end if
      do 30 i = 1,npt
        yp(i) = dtp(i)
30    continue
      call CRACKM(npt,phicr,ei,mcri)
c--- if max moment is less than cracking moment
      if(mcrn.le.mmax)then
        write(5,*) 'Mcr =',mcrn
        write(5,*) 'Mmax =',mmax
        write(5,*) 'Cracking moment doesnt occur'
        x1 = 0
        deflp = 0
        ip = 1
        goto 35
      end if
c--- find location of plastic hinge
      if(ilt.eq.1)then
        xr = (dl * leng) + ll
        xp = (xr**2)-(2*dl*(mcrn+(ll*leng)+(dl*(leng**2)/2)))
        xq = xp**.5
        if(xq.gt.xr)then
          x1 = (xr + xq) / dl
        else if(xq.eq.xr)then
          x1 = (xr + xq) / dl
          if(x1.gt.1)then
            x1 = (xr - xq) / dl
          end if
        else
          x1 = (xr - xq) / dl
        end if
      else
        x1 = leng
        defle = 0
        write(6,*) 'Deflection (elastic) =',defle
        goto 55
      end if
```

```
c--- divide beam into plastic and elastic section
c--- first find deflection for elastic section
c--- divide section into four equal parts
35    continue
      d = (leng - x1)/4
      xn = x1
      do 50 i = 1,5
        do 40 j = 1,npt
          yp(j) = dtp(j)
40      continue
        if(ilt.eq.1)then
          ms = -((ll*(leng-xn)) + (dl*((leng-xn)**2)/2))
          if(i.eq.1)then
            ms = ms + .001
          else if(i.eq.5)then
            ms = -.001
          end if
        else
          ms = lm
        end if
c--- find curvature for given x-section and moment
        call DESIGN(1,nmt,npt,ne,ms)
        cu(i) = curv
        jj(i) = -(leng - x1 - (d*(i-1))) * cu(i)
        xn = xn + d
50    continue
c--- find deflection due to elastic range
      defle = (d/3)*(jj(1)+jj(5)+(4*jj(2))+(2*jj(3))+(4*jj(4)))
      write(6,*) 'Deflection (elastic) =',defle
      if(ip.eq.1)then
        goto 80
      end if
c--- find deflection for plastic section
c--- divide section into eight equal parts
55    continue
      dp = x1 / 8
      xm = 0
c--- find moment and curvature at each x-section
      do 70 i = 1,9
        do 60 j = 1,npt
          yp(j) = dtp(j)
60      continue
        if(ilt.eq.1)then
          ms = -((ll*(leng-xm)) + (dl*((leng-xm)**2)/2))
          if(i.eq.9)then
            ms = ms - .001
            seclen = dp / 2
          else if(i.eq.1)then
            seclen = dp / 2
          else
            seclen = dp
          end if
        else
          ms = lm
          if(i.eq.1)then
            seclen = dp/2
          else if(i.eq.9)then
            seclen = dp/2
```

```
          else
            seclen = dp
          end if
        end if
        write(6,*) ms,xm
c--- find curvature for given x-section and moment
        call DESIGN(l,nmt,npt,ne,ms)
        if(mesp.eq.1)then
          write(6,*) 'Moment exceeds section properties'
          goto 100
        end if
        cup(i) = curv
        jp(i) = -(leng - (dp*(i-1))) * cup(i)
        write(6,*) xm,cup(i),jp(i)
        xm = xm + dp
70    continue
c--- calculate deflection due to plastic section
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      deflp = (dp/3)*(simp1 + simp2)
80    continue
      write(6,*) 'Deflection (plastic) =',deflp
      defl = defle + deflp
      write(6,*) 'Deflection (total) =',defl
100   continue
      return
      end
```

```
      subroutine SIMSUP(l,ne,npt)
c-----------------------------------------------------------------------
c     Purpose    : To find the deflection at the centre of a simply
c                  supported prestressed beam with specified load, dead
c                  load, tendon profile and beam dimensions.
c-----------------------------------------------------------------------
c
      integer nmt,l,ne,npt,i,mat,ip,mesp
      integer icode,ilt,its,in,inp,j,ij
      character*1 inchr
      real etasi,area,yp,leng,lenip,ll,dl,ls,yp1,yp2,yp3
      real atp,btp,ctp,atpi,btpi,ctpi,d,dp
      real mcr,mcrn,mcri,ei,phicr,x1,mmax,curv
      real x2,m1,xdx,delm,deldm,mdx1,m2,simp1,simp2
      real xn,xm,ms,cu,cup,jj,jp,defle,deflp,defl,seclen
      dimension etasi(10),area(10),yp(10),mat(10)
      dimension atp(3,10),btp(3,10),ctp(3,10),lenip(10,3)
c      dimension atpi(10),btpi(10),ctpi(10)
      dimension  cu(10),jj(10),cup(10),jp(10),in(10)
      common /bp/ mat,area,yp
      common /bi/ etasi
      common /cu/ curv
      common /mesp/ mesp
      common /mcr/ mcr,mcrn
      common /sec/ seclen
      common /its/ its
      common /simp/ leng,in,lenip,atp,btp,ctp,dl
c
5     continue
c--- check for tension stiffening
      write(5,1030)
1030  format('Should tension stiffening be concidered? (y/n)')
      call KBDGET (inchr)
      icode=ichar(inchr)
      if(icode.eq.121)then
        its = 1
      else if(icode.eq.89)then
        its = 1
      else if(icode.eq.110)then
        its = 2
      else if(icode.eq.78)then
        its = 2
      else
        goto 5
      end if
      ip = 0
      seclen = 0
c--- get material types
      call MATINP(nmt,l)
c--- get cross-sectional values
      call ELINP(ne,npt,l)
c--- for second run check to change properties
      if(l.ne.0)then
7       continue
        write(5,1061)
        write(5,1063)
1061  format('Do you wish to (a) change other beam properties,')
1063  format('            or (b) leave all beam properties as is?')
```

```
        call KBDGET (inchr)
        icode = ichar(inchr)
        if(icode.eq.97)then
          goto 8
        else if(icode.eq.65)then
          goto 8
        else if(icode.eq.98)then
          goto 25
        else if(icode.eq.66)then
          goto 25
        else
          goto 7
        end if
      end if
8     continue
c--- get initial prestrain in tendons
      do 10 i = 1,npt
        write(5,1001) i
        write(5,1002)
1001  format('For prestress tendon ',i2)
1002  format('...enter initial prestrain')
        read(*,*) etasi(i)
10    continue
c--- get input for beam dimension
      write(5,1003)
      read(*,*) leng
1003  format('Enter length of beam')
c--- get tendon profile equations
      do 20 i = 1,npt
        write(5,1001) i
        write(5,1050)
        read(*,*) in(i)
        do 15 j = 1,in(i)
          write(5,1051) j
          write(5,1052)
          read(*,*) lenip(i,j)
          write(5,1053)
          write(5,1054)
          read(*,*) atp(j,i),btp(j,i),ctp(j,i)
1050  format('...how many equations define the tendon profile?')
1051  format('for equation ',i1)
1052  format('...enter length to end of equation (from beam end)')
1053  format('For quadratic equation ax2 + bx + c...')
1054  format('...enter values for a, b and c respectively.')
15      continue
20    continue
c--- get loading values
      write(5,1009)
      read(*,*) dl
1009  format('Enter dead load (Mn/m)')
25    continue
      write(5,1018)
      write(5,1019)
1018  format('Enter (a) for central point load or')
1019  format('      (b) for two spaced point loads')
      call KBDGET (inchr)
```

```
      icode = ichar(inchr)
      if(icode.eq.97)then
        ilt = 1
      else if(icode.eq.65)then
        ilt = 1
      else if(icode.eq.98)then
        ilt = 2
      else if(icode.eq.66)then
        ilt = 2
      else
        goto 25
      end if
      if(ilt.eq.1)then
        write(5,1010)
        read(*,*) ll
        write(6,*) 'live load =',ll
1010  format('Enter live load at centre (Mn)')
      else
        write(5,1020)
        read(*,*) ll
        write(6,*) 'live load =',ll
1020  format('Enter total live load (Mn)')
        write(5,1021)
        read(*,*) ls
1021  format('Enter space between two load points (m)')
      end if
c--- find if plastic hinge exists
      if(ilt.eq.1)then
        mmax = ((ll * leng /4) + (dl * (leng **2) /8))
      else
        mmax = ((ll * (leng-ls) /4) + (dl * (leng **2) /8))
      end if
      do 30 i = 1,npt
        yp1 = (atp(in(i),i)*((leng/2)**2))
        yp2 = (btp(in(i),i)*leng/2)
        yp3 = ctp(in(i),i)
        yp(i) = yp1 + yp2 + yp3
30    continue
      call CRACKM(npt,phicr,ei,mcri)
c--- if max moment is less than cracking moment
      if(mcr.ge.mmax)then
        write(5,*) 'Mcr =',mcr
        write(5,*) 'Mmax =',mmax
        write(5,*) 'Cracking moment doesnt occur'
        x1 = leng/2
        deflp = 0
        ip = 1
        goto 58
      end if
c--- find location of plastic hinge
c--- first guess at 3/8 span
      x1 = 3 * leng / 8
38    continue
      do 40 i = 1,npt
        do 45 j = 1,in(i)
          if(x1.le.lenip(i,j))then
            inp = j
            goto 48
```

```
          else if(x1.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 48
          end if
45      continue
48      continue
        yp(i) = (atp(inp,i)*(x1**2))+(btp(inp,i)*x1)+ctp(inp,i)
40    continue
      call CRACKM(npt,phicr,ei,mcri)
      x2 = x1
      if(ilt.eq.1)then
        m1 = (ll*x1/2) + (dl*x1*(leng-x1)/2)
      else
        if(x1.le.((leng-ls)/2))then
          m1 = (ll*x1/2) + (dl*x1*(leng-x1)/2)
        else
          m1 = (ll*(leng-ls)/4) + (dl*x1*(leng-x1)/2)
        end if
      end if
      delm = mcr - m1
      if(delm.le..0001)then
        if(delm.ge.-.0001)then
          goto 55
        end if
      end if
c--- for second part of iteration process
      x1 = x2 + .1
      do 50 i = 1,npt
        do 52 j = 1,in(i)
          if(x1.le.lenip(i,j))then
            inp = j
            goto 53
          else if(x1.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 53
          end if
52      continue
53      continue
        yp(i) = (atp(inp,i)*(x1**2))+(btp(inp,i)*x1)+ctp(inp,i)
50    continue
      call CRACKM(npt,phicr,ei,mcri)
      xdx = x1
      if(ilt.eq.1)then
        m2 = (ll*x1/2) + (dl*x1*(leng-x1)/2)
      else
        if(x1.le.((leng-ls)/2))then
          m2 = (ll*x1/2) + (dl*x1*(leng-x1)/2)
        else
          m2 = (ll*(leng-ls)/4) + (dl*x1*(leng-x1)/2)
        end if
      end if
      deldm = mcr - m2
      mdx1 = (deldm - delm)/.1
```

```
c--- get next guess for location
      x1 = x2 - (delm/mdx1)
      if(x1.gt.(leng/2))then
        x1 = x2 + .1
      else if(x1.le.0)then
        x1 = x2 - .08
      end if
      write(5,*) 'delm =',delm
      write(5,*) 'm1 =',m1
      write(5,*) 'x=',x1
c--- repeat iterative process
      goto 38
55    continue
      write(5,*) 'xf =',x1
      write(5,*) 'mcr =',mcr
c--- divide beam into plastic and elastic section
c--- first find deflection for elastic section
c--- divide section into four equal parts
58    continue
      d = x1 / 4
      xn = 0
      do 70 i = 1,5
        do 60 j = 1,npt
          do 65 ij = 1,in(j)
            if(xn.le.lenip(j,ij))then
              inp = ij
              goto 66
            else if(xn.gt.lenip(j,in(j)))then
              inp = in(j)
              goto 66
            end if
65        continue
66        continue
          yp(j) = (atp(inp,j)*(xn**2))+(btp(inp,j)*xn)+ctp(inp,j)
60      continue
        if(ilt.eq.1)then
          ms = (ll*xn/2) + (dl*xn*(leng-x1)/2)
        else
          if(xn.le.((leng-ls)/2))then
            ms = (ll*xn/2) + (dl*xn*(leng-x1)/2)
          else
            ms = (ll*(leng-ls)/4) + (dl*xn*(leng-x1)/2)
          end if
        end if
        if(i.eq.1)then
          ms = .001
        else if(i.eq.5)then
          ms = ms - .001
        end if
        call DESIGN(1,nmt,npt,ne,ms)
        cu(i) = curv
        jj(i) = (xn / 2) * cu(i)
        xn = xn + d
70    continue
c--- calculate deflection due to elastic zones
      defle = (2*d/3)*(jj(1)+jj(5)+(4*(jj(2)+jj(4)))+(2*jj(3)))
      write(6,*) 'Deflection (elastic) =',defle
```

```
      if(ip.eq.1)then
        goto 90
      end if
c--- find deflection for plastic section
c--- divide section into eight equal parts
      dp = ((leng/2) - x1) / 8
      xm = x1
      do 80 i = 1,9
        do 75 j = 1,npt
          do 72 ij = 1,in(j)
            if(xm.le.lenip(j,ij))then
              inp = ij
              goto 73
            else if(xm.gt.lenip(j,in(j)))then
              inp = in(j)
              goto 73
            end if
72        continue
73        continue
          yp(j) = (atp(inp,j)*(xm**2))+(btp(inp,j)*xm)+ctp(inp,j)
75      continue
        if(ilt.eq.1)then
          ms = (ll*xm/2) + (dl*xm*(leng-x1)/2)
        else
          if(xm.le.((leng-ls)/2))then
            ms = (ll*xm/2) + (dl*xm*(leng-x1)/2)
          else
            ms = (ll*(leng-ls)/4) + (dl*xm*(leng-x1)/2)
          end if
        end if
        if(i.eq.1)then
          ms = ms + .001
          seclen = dp / 2
        else if(i.eq.9)then
          seclen = dp / 2
        else
          seclen = dp
        end if
        call DESIGN(1,nmt,npt,ne,ms)
        if(mesp.eq.1)then
          write(6,*) 'Moment exceeds section properties'
          goto 100
        end if
        cup(i) = curv
        jp(i) = (xm / 2) * cup(i)
        xm = xm + dp
80    continue
c--- calculate deflection due to plastic zone
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      deflp = (2*dp/3)*(simp1 + simp2)
90    continue
      write(6,*) 'Deflection (plastic) =',deflp
c--- calculate total deflection
      defl = defle + deflp
      write(6,*) 'Deflection (total) =',defl
100   continue
      return
      end
```

```
      subroutine FULFIX(l,ne,npt)
c-----------------------------------------------------------------------
c     Purpose    : To find the deflection at the centre of a fully
c                  fixed prestressed beam with specified load, dead
c                  load, tendon profile and beam dimensions.
c     Definitions: rotlp,rot2p,rote - the end rotation due to the end
c                  (negative) plastic zone, the central (positive)
c                  plastic zone and the elastic zones respectively
c                  rot - the total end rotation
c     Details of routine: For fully fixed beams, redistribution takes
c     place as soon as the first section cracks. to find the central
c     deflection the amount of redistribution must be determined. This
c     is done by first guessing an amount of redistribution and adding
c     it linearly to the elastic moment diagram. The curvatures at
c     certain sections are determined as for cantilevers and simply
c     supported beams, but first the end rotation must be determined.
c     The Newton-Rhapson iteration method is used to determine the
c     correct amount of redistribution, giving zero end rotation, and
c     the deflection can then be found.
c-----------------------------------------------------------------------
c
      integer nmt,l,ne,npt,i,mat,mesp,i1,i2,ix,ix2,ii
      integer icode,its,in,inp,j,ij,iph
      character*1 inchr
      real etasi,area,yp,leng,lenip,ll,dl,dm,x3,x4,msecmp
      real atp,btp,ctp,atpi,btpi,ctpi,d,dp,msl,msd,dpl
      real mcr,mcrn,mcri,ei,phicr,x1,mmax,curv,mmaxn,m,mc
      real x2,ml,xdx,delm,deldm,mdx1,m2,simp1,simp2,msecmx
      real xn,xm,ms,cu,cup,jj,jp,defle,deflp,defl,ml,md
      real rotlp,rot2p,rote,rot,cup2,cup3,mmaxpn,redist
      real dml,rotl,dmdx,x,deflp1,deflp2,seclen,yp1,yp2,yp3
      dimension etasi(10),area(10),yp(10),mat(10)
      dimension atp(3,10),btp(3,10),ctp(3,10),lenip(10,3)
      dimension cu(10),jj(10),cup(10),jp(10),cup2(10)
      dimension cup3(10),in(10),mc(20)
      common /bp/ mat,area,yp
      common /bi/ etasi
      common /cu/ curv
      common /mesp/ mesp
      common /mcr/ mcr,mcrn
      common /sec/ seclen
      common /its/ its
      common /fulf/ leng,in,lenip,atp,btp,ctp,dl
      common /m/ mc,m
      common /ph/ iph
c
5     continue
c--- check for tension stiffening
      write(5,1030)
1030  format('Should tension stiffening be concidered? (y/n)')
      call KBDGET (inchr)
      icode=ichar(inchr)
      if(icode.eq.121)then
        its = 1
      else if(icode.eq.89)then
        its = 1
```

```
      else if(icode.eq.110)then
        its = 2
      else if(icode.eq.78)then
        its = 2
      else
        goto 5
      end if
      i1 = 0
      i2 = 0
      seclen = 0
c--- get material types
      call MATINP(nmt,1)
c--- get cross-sectional values
      call ELINP(ne,npt,1)
c--- for second run check to change properties
      if(1.ne.0)then
7       continue
        write(5,1061)
        write(5,1063)
1061  format('Do you wish to (a) change other beam properties,')
1063  format('              or (b) leave all beam properties as is?')
        call KBDGET (inchr)
        icode = ichar(inchr)
        if(icode.eq.97)then
          goto 8
        else if(icode.eq.65)then
          goto 8
        else if(icode.eq.98)then
          goto 25
        else if(icode.eq.66)then
          goto 25
        else
          goto 7
        end if
      end if
8     continue
c--- get initial prestrain in tendons
      do 10 i = 1,npt
        write(5,1001) i
        write(5,1002)
1001  format('For prestress tendon ',i2)
1002  format('...enter initial prestrain')
        read(*,*) etasi(i)
10    continue
c--- get input for beam dimension
      write(5,1003)
      read(*,*) leng
1003  format('Enter length of beam')
c--- get tendon profile equations
      do 20 i = 1,npt
        write(5,1001) i
        write(5,1050)
        read(*,*) in(i)
```

```
      do 15 j = 1,in(i)
        write(5,1051) j
        write(5,1052)
        read(*,*) lenip(i,j)
        write(5,1053)
        write(5,1054)
        read(*,*) atp(j,i),btp(j,i),ctp(j,i)
1050  format('...how many equations define the tendon profile?')
1051  format('for equation ',i1)
1052  format('...enter length to end of equation (from beam end)')
1053  format('For quadratic equation ax2 + bx + c...')
1054  format('...enter values for a, b and c respectively.')
15      continue
20    continue
c--- get loading values
25    continue
      write(5,1009)
      read(*,*) dl
      write(6,*) 'dead load =',dl
1009  format('Enter dead load (Mn/m)')
      write(5,1010)
      read(*,*) ll
      write(6,*) 'live load =',ll
1010  format('Enter live load at centre (Mn)')
c--- find if plastic hinge exists
      mmaxn = -((ll * leng /8) + (dl * (leng **2) /12))
      mmax = ((ll * leng /8) + (dl * (leng **2) / 24))
c--- first check for hinge with +ve moment
      do 30 i = 1,npt
        yp1 = (atp(in(i),i)*((leng/2)**2))
        yp2 = (btp(in(i),i)*leng/2)
        yp3 = ctp(in(i),i)
        yp(i) = yp1 + yp2 + yp3
30    continue
      itb = 1
      iph = 1
      call ANALYS(nmt,npt,ne,1,4,itb,mcri)
      msecmp = m
      call CRACKM(npt,phicr,ei,mcri)
c--- if max moment is less than cracking moment
      write(6,*) 'Mcr =',mcr
      write(6,*) 'Mmax =',mmax
      if(mcr.ge.mmax)then
        write(6,*) 'No cracking at the centre'
        i1 = 1
      end if
c--- now for hinge with -ve moment
      do 40 i = 1,npt
        yp(i) = ctp(1,i)
40    continue
      itb = 2
      iph = 1
      call ANALYS(nmt,npt,ne,1,4,itb,mcri)
      msecmx = m
      call CRACKM(npt,phicr,ei,mcri)
      write(6,*) 'Mcrn =',mcrn
      write(6,*) 'Mmaxn =',mmaxn
```

```
      if(mcrn.le.mmaxn)then
        write(6,*) 'No cracking occurs at the ends'
        i2 = 1
c--- for no cracking anywhere along the beam
        if(i1.eq.1)then
          write(6,*) 'Cracking moment doesnt occur'
c--- find centre deflection for no cracking
          d = leng / 16
          xn = 0
          do 60 i = 1,9
            do 50 j = 1,npt
              do 45 ij = 1,in(j)
                if(xn.le.lenip(j,ij))then
                  inp = ij
                  goto 46
                else if(xn.gt.lenip(j,in(j)))then
                  inp = in(j)
                  goto 46
                end if
45            continue
46            continue
              yp(j) = (atp(inp,j)*(xn**2))+(btp(inp,j)*xn)+ctp(inp,j)
50          continue
            msl = l1*(xn-(leng/4))/2
            msd = d1*((leng*xn)-((leng**2)/6)-(xn**2))/2
            ms = msl + msd
            if(i.eq.1)then
              ms = ms + .0001
            else if(i.eq.9)then
              ms = ms + .0001
            end if
            call DESIGN(1,nmt,npt,ne,ms)
            cu(i) = curv
            jj(i) = (xn / 2) * cu(i)
            xn = xn + d
60        continue
          simp1 = jj(1) + (4*jj(2)) + (2*jj(3)) + (4*jj(4)) + jj(9)
          simp2 = (2*jj(5)) + (4*jj(6)) + (2*jj(7)) + (4*jj(8))
          defle = (2*d/3)*(simp1+simp2)
          write(6,*) 'Deflection (elastic) =',defle
          write(6,*) 'Deflection (plastic) = 0.00000'
          write(6,*) 'Deflection (total) =',defle
          goto 500
        end if
      end if
c--- for load causing cracking moments to be reached
c--- use a Newton-Rhapson iteration to find moments
c--- first guess the maximum -ve moment of end section
      if(msecmx.le.mmaxn)then
        dm = 0
      else
        dm = (msecmx - mmaxn) + .01
      end if
      if((mmax + dm).gt.msecmp)then
        write(6,*) 'Section properties exceeded'
        goto 500
      end if
```

```
61    continue
      ii = 1
62    continue
c--- find where hinges occurs (dist x1 from support)n
c--- first for negative moment - guess at 0
      x1 = 0
      ix = 0
65    continue
      if(x1.lt.0)then
        x1 = -x1
      end if
      if(x1.gt.leng/2)then
        x1 = x1 - (leng/2)
      end if
      do 70 i = 1,npt
        do 68 j = 1,in(i)
          if(x1.le.lenip(i,j))then
            inp = j
            goto 69
          else if(x1.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 69
          end if
68        continue
69        continue
          yp(i) = (atp(inp,i)*(x1**2))+(btp(inp,i)*x1)+ctp(inp,i)
70    continue
      call CRACKM(npt,phicr,ei,mcri)
      x2 = x1
c--- find actual moment due to loads at section
      m1 = l1 * ( x2 - (leng/4) ) /2
      md = d1 * ((leng * x2) - (x2**2) - ((leng**2)/6))/2
      m1 = m1 + md + dm
      if(ix.eq.0)then
        if(mcrn.le.m1)then
          write(6,*) 'Cracking doesnt occur at ends'
          x1 = 0
          goto 85
        end if
      end if
      if(m1.gt.0)then
        x1 = .5 * x1
        goto 65
      end if
      delm = mcrn - m1
c--- check if moment within required range
      if(delm.le..0001)then
        if(delm.ge.-.0001)then
          goto 80
        end if
      end if
c--- set x1 for second part of iteration
      x1 = x2 + .1
```

```
      do 75 i = 1,npt
        do 72 j = 1,in(i)
          if(x1.le.lenip(i,j))then
            inp = j
            goto 73
          else if(x1.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 73
          end if
72      continue
73      continue
        yp(i) = (atp(inp,i)*(x1**2))+(btp(inp,i)*x1)+ctp(inp,i)
75    continue
      call CRACKM(npt,phicr,ei,mcri)
      xdx = x1
c--- get actual moment due to loads at section
      ml = ll * ( xdx - (leng/4) ) /2
      md = dl * ((leng * xdx) - (xdx**2) - ((leng**2)/6))/2
      m2 = ml + md + dm
      deldm = mcrn - m2
      mdx1 = (deldm - delm)/.1
c--- get next value for x1 for iteration
      x1 = x2 - (delm/mdx1)
      ix = ix + 1
c--- continue with iteration
      goto 65
80    continue
c--- distance to hinge from support
      write(6,*) 'xf =',x1
      write(6,*) 'mcrn =',mcrn
85    continue
c--- now for positive moment - guess at mid-span
      x3 = leng / 2
      ix2 = 0
90    continue
      if(x3.lt.0)then
        x3 = -x3
      end if
      if(x3.gt.leng/2)then
        x3 = leng - x3
      end if
      do 95 i = 1,npt
        do 92 j = 1,in(i)
          if(x3.le.lenip(i,j))then
            inp = j
            goto 93
          else if(x3.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 93
          end if
92      continue
93      continue
        yp(i) = (atp(inp,i)*(x3**2))+(btp(inp,i)*x3)+ctp(inp,i)
95    continue
      call CRACKM(npt,phicr,ei,mcri)
      x4 = x3
```

```
c--- get actual moment due to loads at section
      ml = ll * ( x4 - (leng/4) ) /2
      md = dl * ((leng * x4) - (x4**2) - ((leng**2)/6))/2
      ml = ml + md + dm
      if(ix2.eq.0)then
        if(mcr.ge.ml)then
          write(5,*) 'Cracking doesnt occur at centre'
          x3 = leng / 2
          goto 110
        else
          x3 = x3 - .1
          ix2 = 1
          goto 90
        end if
      end if
      if(ml.lt.0)then
        x3 = 1.2 * x3
        goto 90
      end if
      delm = mcr - ml
c--- check if moment within required range
      if(delm.le..0001)then
        if(delm.ge.-.0001)then
          goto 105
        end if
      end if
c--- reset x for second part of iteration
      x3 = x4 + .1
      if(x3.gt.leng/2)then
        x3 = x4 - .15
        goto 90
      end if
      do 100 i = 1,npt
        do 97 j = 1,in(i)
          if(x3.le.lenip(i,j))then
            inp = j
            goto 98
          else if(x3.gt.lenip(i,in(i)))then
            inp = in(i)
            goto 98
          end if
97      continue
98      continue
        yp(i) = (atp(inp,i)*(x3**2))+(btp(inp,i)*x3)+ctp(inp,i)
100   continue
      call CRACKM(npt,phicr,ei,mcri)
      xdx = x3
c--- get actual moment at section due to loading
      ml = ll * ( xdx - (leng/4) ) /2
      md = dl * ((leng * xdx) - (xdx**2) - ((leng**2)/6))/2
      m2 = ml + md + dm
      deldm = mcr - m2
      mdx1 = (deldm - delm)/.1
c--- get next value for x for iteration
      x3 = x4 - (delm/mdx1)
      ix2 = ix2 + 1
c--- continue iteration
      goto 90
105   continue
```

```
c--- distance from support to hinge
      write(6,*) 'xf =',x3
      write(6,*) 'mcr =',mcr
110   continue
c--- must now find rotation
c--- first due to end plastic zone ( 0 to x1 )
      dp1 = x1 / 8
      xm = 0
      do 130 i = 1,9
        do 120 j = 1,npt
          do 115 ij = 1,in(j)
            if(xm.le.lenip(j,ij))then
              inp = ij
              goto 116
            else if(xm.gt.lenip(j,in(j)))then
              inp = in(j)
              goto 116
            end if
115       continue
116       continue
          yp(j) = (atp(inp,j)*(xm**2))+(btp(inp,j)*xm)+ctp(inp,j)
120     continue
        msl = ll * ( xm - (leng/4) ) /2
        msd = dl * ((leng * xm) - (xm**2) - ((leng**2)/6))/2
        ms = msl + msd + dm
        if(i.eq.9)then
          ms = ms - .001
          seclen = dp1 / 2
        else if(i.eq.1)then
          seclen = dp1 / 2
        else
          seclen = dp1
        end if
        call DESIGN(l,nmt,npt,ne,ms)
        if(mesp.eq.1)then
          write(6,*) 'Moment exceeds section properties'
          goto 500
        end if
        if(ii.eq.1)then
          cup(i) = curv
          jp(i) = cup(i)
        else
          cup3(i) = curv
          jp(i) = curv
        end if
        xm = xm + dp1
130   continue
c--- calculate end rotation due to end plastic zone
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      rot1p = (2*dp1/3)*(simp1 + simp2)
140   continue
      write(6,*) 'End rotation (plastic) =',rot1p
```

```
c--- next due to centre plastic zone ( x3 to l/2 )
      dp2 = ((leng/2) - x3) / 8
      xm = x3
      do 160 i = 1,9
        do 150 j = 1,npt
          do 145 ij = 1,in(j)
            if(xm.le.lenip(j,ij))then
              inp = ij
              goto 146
            else if(xm.gt.lenip(j,in(j)))then
              inp = in(j)
              goto 146
            end if
145       continue
146       continue
          yp(j) = (atp(inp,j)*(xm**2))+(btp(inp,j)*xm)+ctp(inp,j)
150     continue
        msl = ll * ( xm - (leng/4) ) /2
        msd = dl * ((leng * xm) - (xm**2) - ((leng**2)/6))/2
        ms = msl + msd + dm
        if(i.eq.1)then
          ms = ms + .001
          seclen = dp2 / 2
        else if(i.eq.9)then
          seclen = dp2 / 2
        else
          seclen = dp2
        end if
        call DESIGN(l,nmt,npt,ne,ms)
        if(mesp.eq.1)then
          write(6,*) 'Moment exceeds section properties'
          goto 500
        end if
        if(ii.eq.1)then
          cup2(i) = curv
          jp(i) = cup2(i)
        else
          cup3(i) = curv
          jp(i) = curv
        end if
        xm = xm + dp2
160   continue
c--- calculate end rotation due to central plastic zone
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      rot2p = (2*dp2/3)*(simp1 + simp2)
170   continue
      write(6,*) 'Centre rotation (plastic) =',rot2p
c--- next due to elastic zone ( x1 to x3 )
      d = (x3 - x1) / 8
      xm = x1
      do 190 i = 1,9
        do 180 j = 1,npt
          do 175 ij = 1,in(j)
```

```
            if(xm.le.lenip(j,ij))then
              inp = ij
              goto 176
            else if(xm.gt.lenip(j,in(j)))then
              inp = in(j)
              goto 176
            end if
175       continue
176       continue
          yp(j) = (atp(inp,j)*(xm**2))+(btp(inp,j)*xm)+ctp(inp,j)
180     continue
        msl = ll * ( xm - (leng/4) ) /2
        msd = dl * ((leng * xm) - (xm**2) - ((leng**2)/6))/2
        ms = msl + msd + dm
        if(i.eq.1)then
          ms = ms + .001
        else if(i.eq.9)then
          ms = ms - .001
        end if
        call DESIGN(l,nmt,npt,ne,ms)
        if(ii.eq.1)then
          cu(i) = curv
          jj(i) = cu(i)
        else
          cup3(i) = curv
          jj(i) = curv
        end if
        xm = xm + d
190   continue
c--- calculate end rotation due to elastic zone
      simp1 = jj(1) + (4*jj(2)) + (2*jj(3)) + (4*jj(4)) + jj(9)
      simp2 = (2*jj(5)) + (4*jj(6)) + (2*jj(7)) + (4*jj(8))
      rote = (2*d/3)*(simp1 + simp2)
195   continue
      write(6,*) 'Centre rotation (elastic) =',rote
      rot = rot1p + rot2p + rote
      if(ii.eq.1)then
        rot1 = rot
      end if
      write(6,*) 'Total rotation =',rot
c--- check if rotation close enough to zero
      if(ii.eq.1)then
        if(rot.ge.-.000001)then
          if(rot.le..000001)then
            mmaxpn = mmaxn + dm
            redist = mmaxpn/mmaxn
            write(6,*) 'Redist. value =',dm
            write(6,*) 'Redistribution =',redist
            goto 210
          end if
        end if
      end if
      if(ii.eq.2)then
        dmdx = (rot - rot1) / .01
c--- get next value for redistribution for iteration
        dm = dm1 - (rot1/dmdx)
```

```
        if(dm.le.0)then
          dm = - dm
        end if
        if(dm.gt.(msecmp-mmax))then
          dm = msecmp - mmax - .01
        end if
c--- continue with iteration procedure
        goto 200
      end if
      ii = 2
      dm1 = dm
c--- set redistribution value for second part of iteration
      dm = dm1 + .01
      goto 62
200   continue
      goto 61
210   continue
c--- find deflection for redistributed moments
c--- first for end plastic zone
      dp1 = x1/8
      write(6,*) 'x1 =',x1,':dp1 =',dp1
      do 220 i = 1,9
        x = dp1 * (i - 1)
        write(6,*) 'x =',x
        write(6,*) 'cu(',i,') =',cup(i)
        jp(i) = cup(i) * (x/2)
220   continue
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      deflp1 = (2*dp1/3)*(simp1 + simp2)
c--- next for centre plastic zone
      dp2 = ((leng/2) - x3) / 8
      write(6,*) 'x3 =',x3,':dp2 =',dp2
      do 230 i = 1,9
        x = x3 + (dp2 * (i - 1))
        write(6,*) 'x =',x
        write(6,*) 'cu(',i,') =',cup2(i)
        jp(i) = cup2(i) * (x/2)
230   continue
      simp1 = jp(1) + (4*jp(2)) + (2*jp(3)) + (4*jp(4)) + jp(9)
      simp2 = (2*jp(5)) + (4*jp(6)) + (2*jp(7)) + (4*jp(8))
      deflp2 = (2*dp2/3)*(simp1 + simp2)
c--- finally for elastic zone
      d = (x3 - x1) / 8
      write(6,*) ':d =',d
      do 240 i = 1,9
        x = x1 + (d * (i - 1))
        write(6,*) 'x =',x
        write(6,*) 'cu(',i,') =',cu(i)
        jj(i) = cu(i) * (x/2)
240   continue
      simp1 = jj(1) + (4*jj(2)) + (2*jj(3)) + (4*jj(4)) + jj(9)
      simp2 = (2*jj(5)) + (4*jj(6)) + (2*jj(7)) + (4*jj(8))
      defle = (2*d/3)*(simp1 + simp2)
      defl = deflp1 + deflp2 + defle
      write(6,*) 'Deflection (plastic 1) =',deflp1
      write(6,*) 'Deflection (plastic 2) =',deflp2
      write(6,*) 'Deflection (elastic) =',defle
      write(6,*) 'Deflection (total) =',defl
500   continue
      return
      end
```

# APPENDIX C RESULTS OF TESTING

As part of a B Sc. thesis, five simply supported beams were constructed and loaded to failure to obtain the moment-curvature relationships for each beam. The beams each had rectangular cross-sections of breadth 158 mm by height 250 mm. They each contained various amounts of prestressing steel, which was layered straight across the beam to give a uniform cross-section.

The first two beams, refereed to as B1 and B2, were each prestressed with three 5 mm wire strands placed 38 mm above the beam soffit. The next two, C1 and C2, had an additional two 5 mm wire strands placed 50 mm above the other three. The final beam, D1, had four 7 mm wire strands, two 38 mm above the soffit and the other to 88 mm above the soffit. This gave a steel area of 58,89 mm² for B1 and B2, 98,15 mm² for C1 and C2 and 153,95 mm² for D1. These areas represent 0,15%, 0,25% and 0,4% prestress steel area respectively.

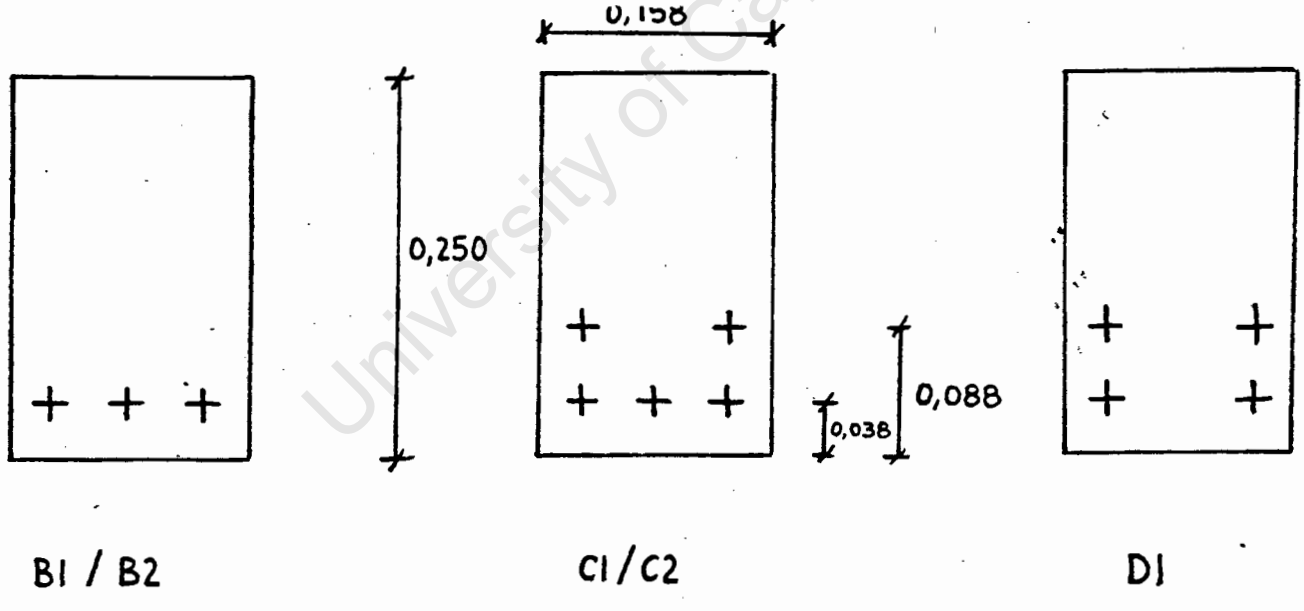



Fig. C.1 **Cross-section showing position of steel**

The wires were pulled to 75% of the ultimate strength, and 10% prestress losses were assumed. For 5 mm wore $f_{pu}$ = 1632 MPa, and thus for B1, B2, C1 and C2 the prestress was approximately 1101,6 MPA, while for D1 the prestress was approximately 1147,5 MPa.

From the stress-strain curves the strain due to prestressing was approximately 0,00648 for the 5 wire cables and 0,0057375 for the 7 wire cables.

The ultimate concrete strength at testing was found to be 53 MPA for beams B1 and B2, while this was found to be 45 MPa for beams C1, C2 and D1. The steel was tested and the plot of stress-strain relationships were found and used in the program. These are shown in Figs. C.2 and C.3.

The moment-curvature relationships are shown in Figs C.4, C.5 and C.6 for the experimental and theoretical results.

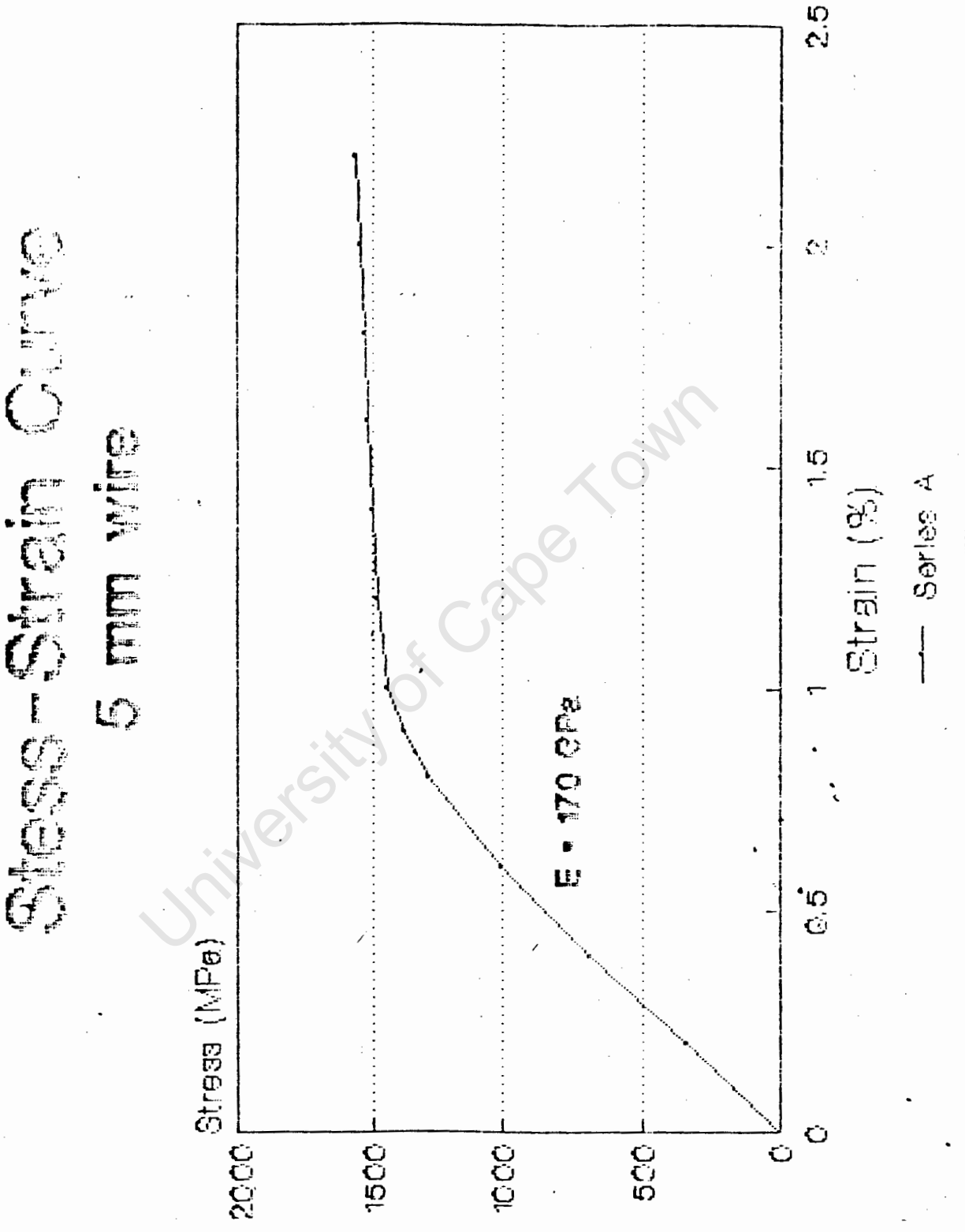





**Fig C.2** Stress–strain relationship for 5 mm$\phi$ wire tendons

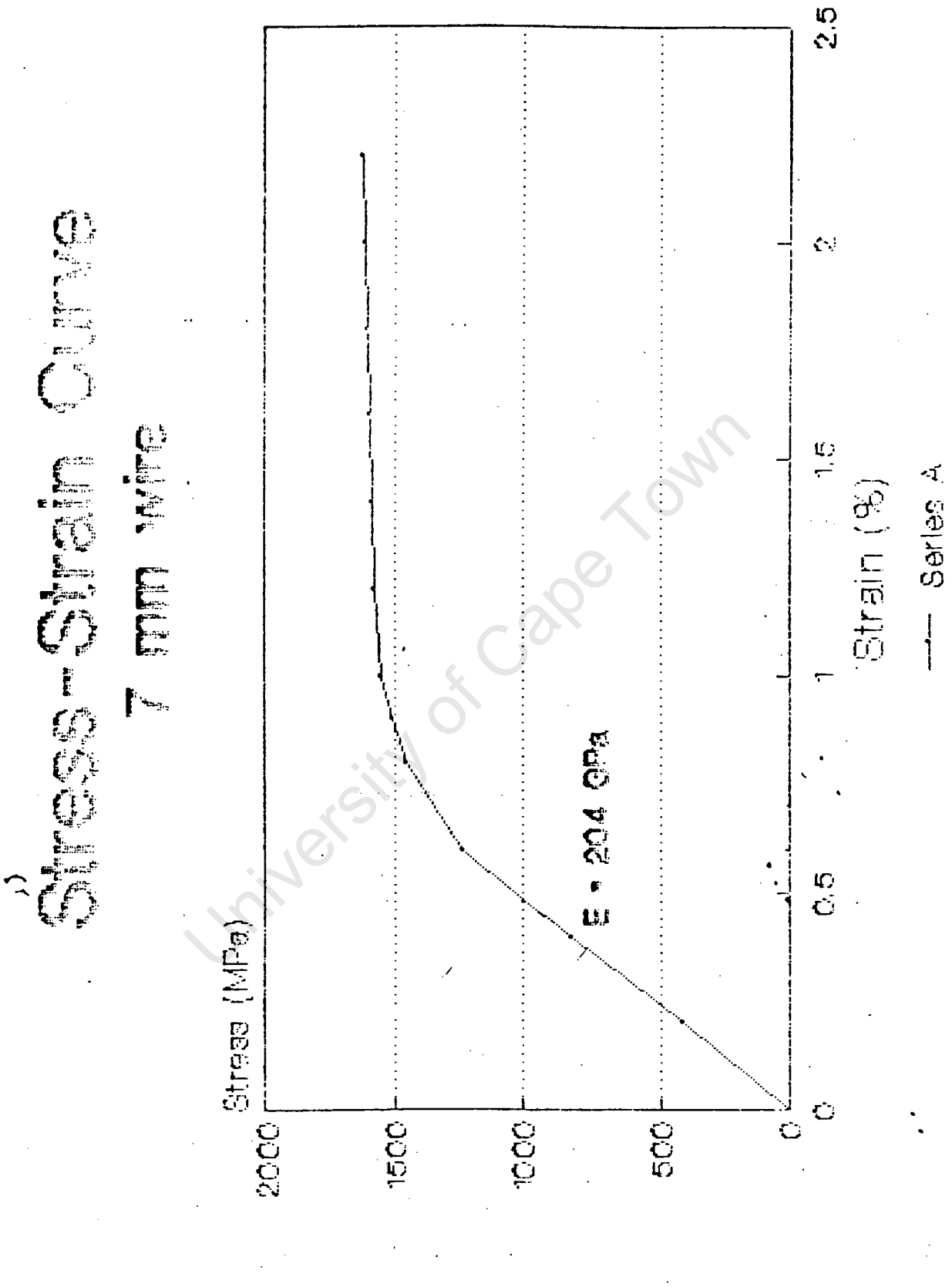





**Fig C.3** Stress-strain relationship for 7 mm$\phi$ wire tendons

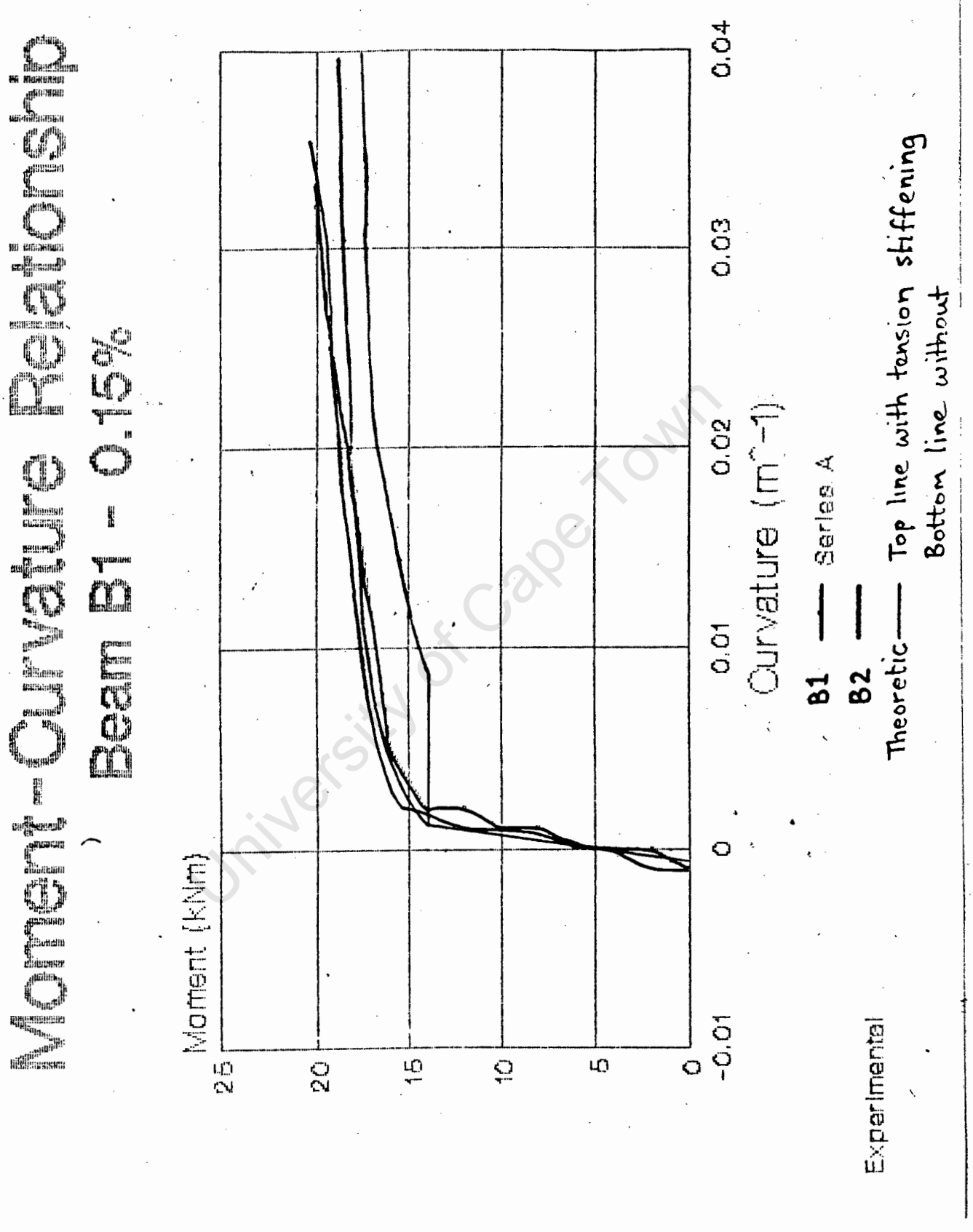





**Fig C.4 Moment - curvature relationship for 0,15% steel examples**

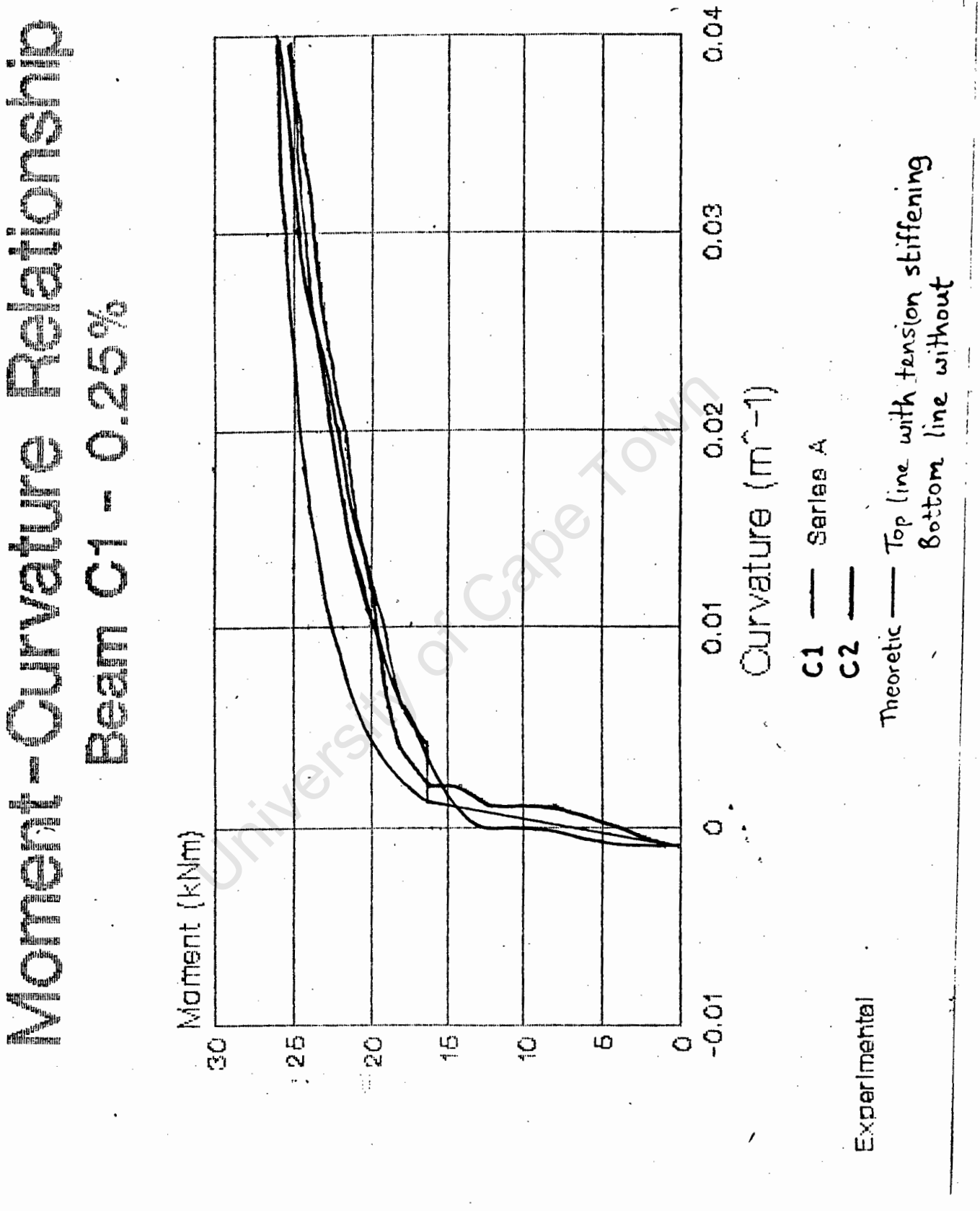





Fig C.5 Moment - curvature relationship for 0,25% steel examples

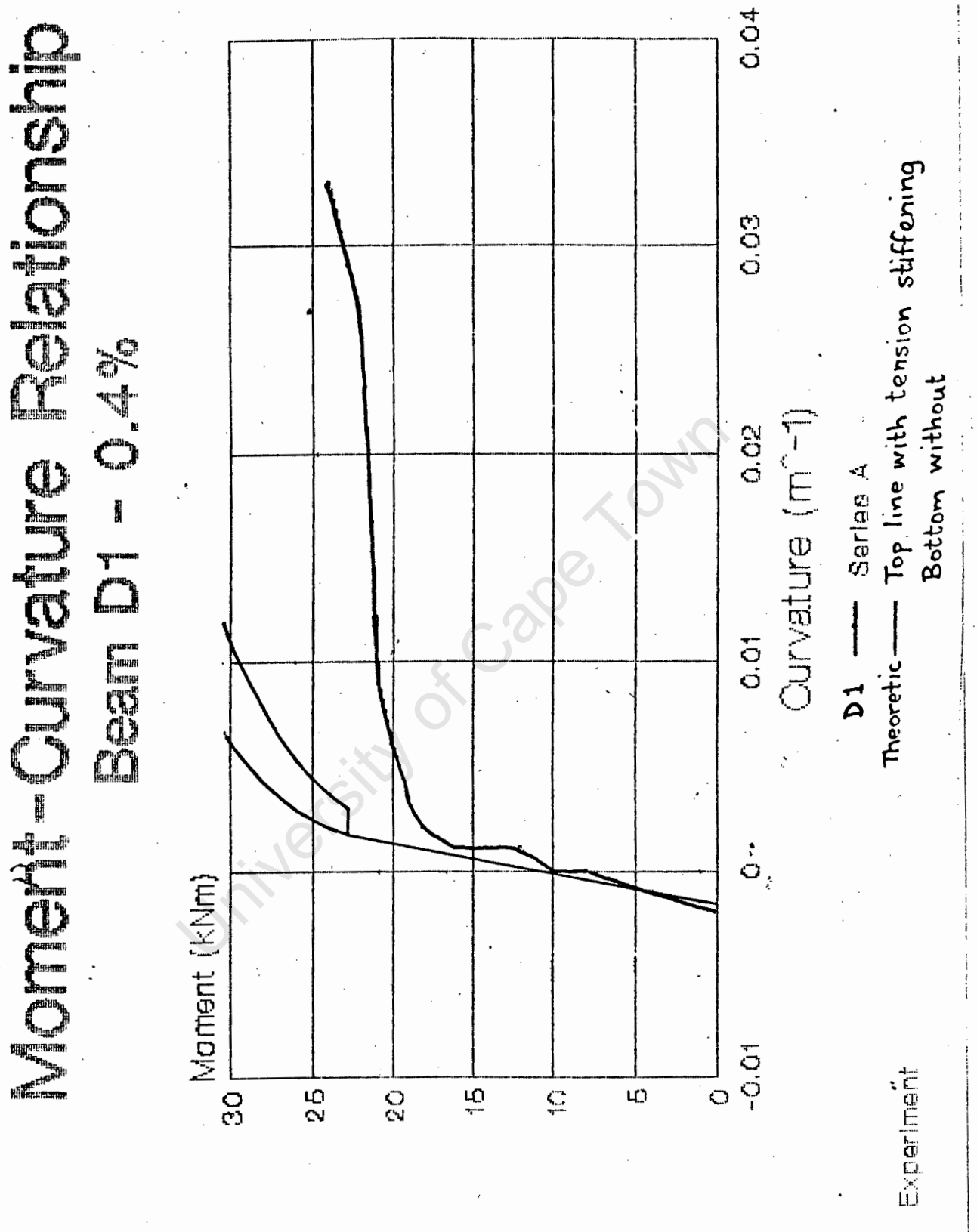





**Fig C.6** Moment - curvature relationship for 0,4% steel examples

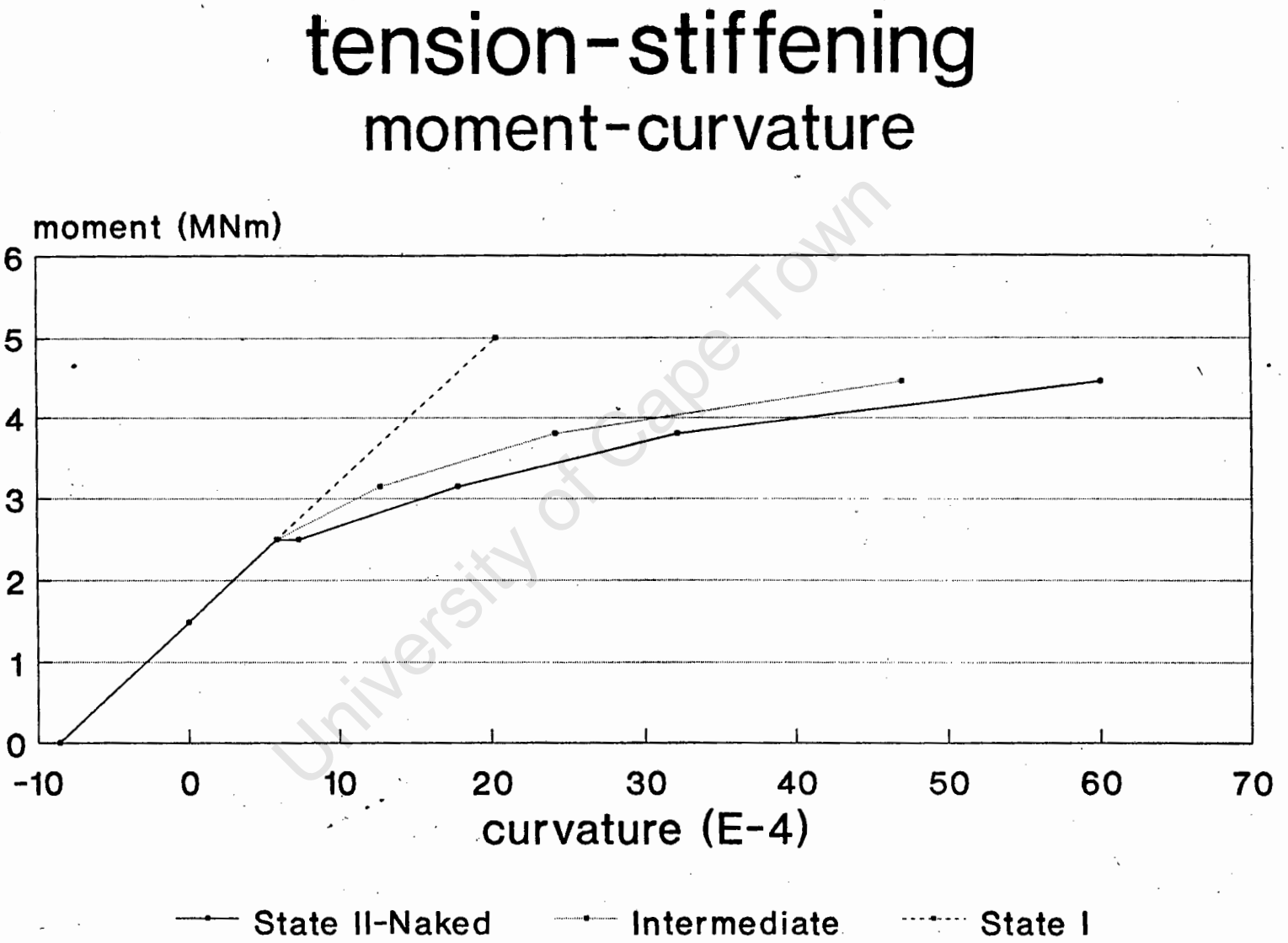



Fig D.1 Moment-curvature relationship for a cross-section

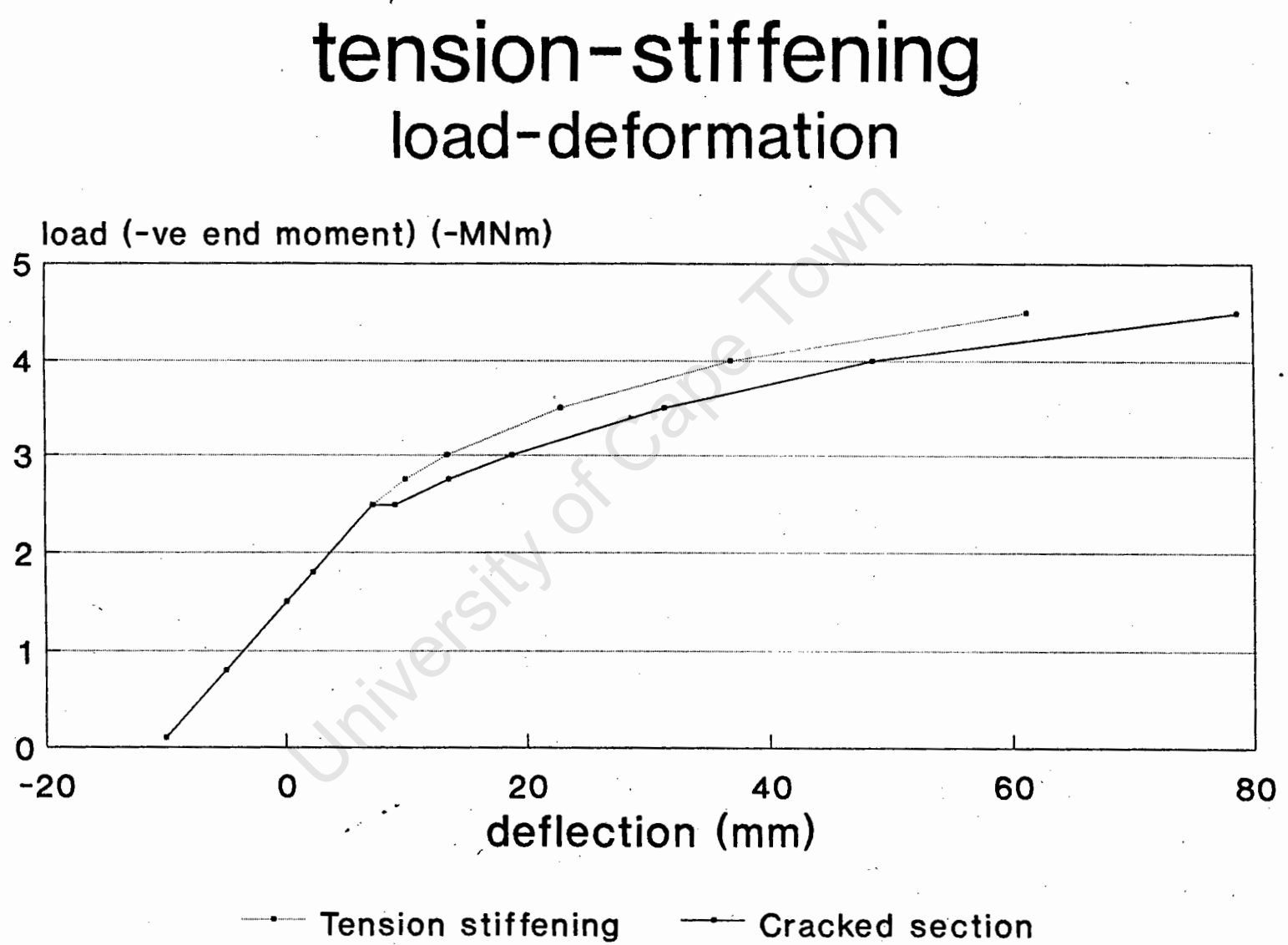





Fig D.2 Load- deflection curve for the beam.

# APPENDIX E FULLY FIXED T-BEAM EXAMPLE

a) The following example is used to investigate the redistribution of bending moments. A fully fixed T-beam of 50m is considered. The beam has cross-section dimensions of 4,5 x 0,3 m for the flange, and 0,4 x 3,35 m for the web, as shown in Fig. E.1. The concrete is of grade 50 MPa with Elastic Modulus $E_c$ = 35 GPa.

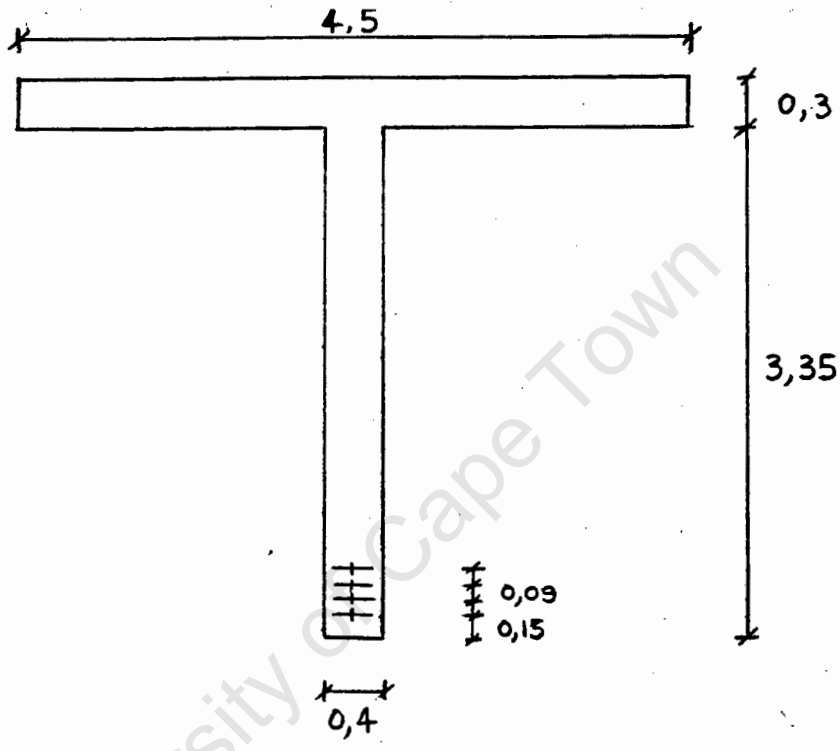



**Fig E.1 Beam cross-section**

The steel consists of 4 prestress cables, each containing 10 tendons, which are 15,2 mm 7 wire strands. The total cross-sectional area of the prestress steel is 5532 mm². The steel has strength $f_{pu}$ = 1700 MPa. Assuming, with losses, a prestress of $F_{pe}$ = 0,65 $F_{pu}$, the initial prestrain is 0,0056.

The cables are draped with a minimum edge distance of 150 mm and a minimum edge separation of 90 mm. For this example the profile is taken as the average of the four profiles and is defined by two quadratic equations. The axes are shown in Fig. E2, with the x axis along the soffit of the beam, and the y axis up the left hand support. The first equation is;

$$y = -0.01286x^2 + 3,5$$

from the support to a distance 10 m along the beam, while the second equation is;

$$y = -0,00857333x^2 - 0,428667x + 5,64334$$

from 10 m to the centre of the beam. The beam is symmetrical and therefore only half the profile need be defined.

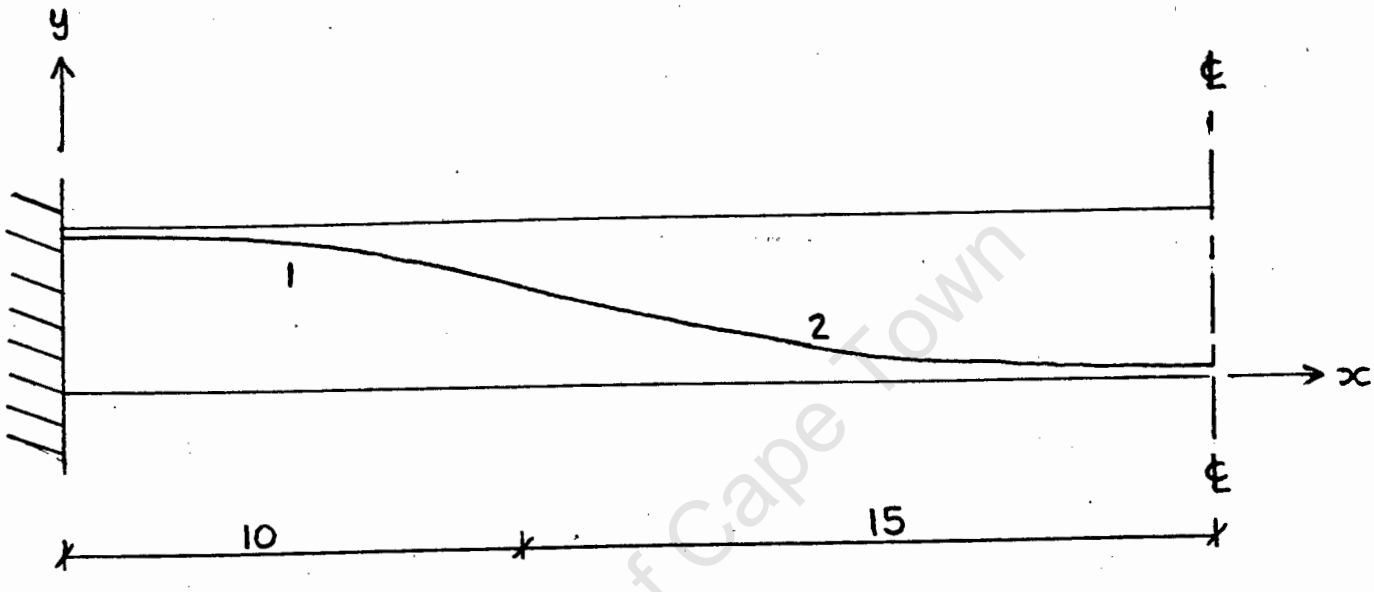



**Fig E.2 Cable profile**



The loading on the beam consists of a downward point load of 144 kN at the centre of the beam, and a variable distributed load. The dead load consists of the beam self-weight of 67,25 kN/m and a 4,5 kN/m premix roadsurfacing, giving 71,75 kN/m. With safety factors of 1,4 for dead load and 1,6 for live load, the loading is 230,4 kN central point load, 100,45 kN/m dead load and a live distributed load.

Using the computer program described in this thesis, the live load was increased until the strain capacity was 52,25 kN/m. For this example at this load the strain in the top fibre of the concrete reached its maximum of 0,35%, while at the same time the steel strain due to loading reached its maximum of 1%. The redistribution of bending moments was found to be 17,132%. The central capacity was calculated as being 32,611 MNm, but the

moment at the centre, which is dependant on the amount of prestressing, was 21,99 MNm. The capacity at the supports was 27,556 MNm.

b) The above example was then modified to determine the effect of untensioned reinforcement on the ductility and thus redistribution. Over the first 10 m the area of the prestress steel was halved, and substituted with untensioned reinforcing steel of strength $f_s$ = 450 MPa. An area of 10 443,2 mm²was used to give the same cross-sectional tensile strength as before. The steel was straight and placed at a height of 3,5 m above the soffit.

It was found that the strain capacity at the supports was exhausted with a factored load of 58,55 kN/m. The bending moment redistribution was found to be 17,227% and the moment reached at the centre was 23,957 MNm.

c) The same examples were used to investigate the effect of using design factors of $\gamma_c$ = 1,5 and $\gamma_s$ = 1,15 on the material properties. The design stress-strain relationships for the concrete and steel were used instead of the real relationships used before.

For the fully prestresses example, the strain capacity at the support was exhausted with a load of 40,55 kN/m. The redistribution was found to be 18,906%. The central capacity was 27,142 MNm, but only 21,953 MNm was reached. The cracking moment at the centre was 20,65 MNm. The capacity at the supports was 25,008 MNm.

d) For the partially prestressed example, the strain capacity was exhausted with a load of 41,65 kN/m. The redistribution was found to be 19,472%. The central moment reached was 22,142 MNm.

It should be noted that for all the above examples the effect of tension stiffening was ignored. By taking this into account the ductility at a section would be reduced, as shown in Appendix D, and therefore the redistribution would be even less than in the above examples.

# APPENDIX F EXAMPLE FOR A FULLY FIXED I-BEAM

A bottom flange similar to the top flange was added to the beam cross-section of Appendix E to give an I-beam. The cross-section is shown in Fig. F.1. The partially prestressed example was investigated with prestress steel area 5532 mm² and reinforcing steel area 20 899 mm² over the first 10 m from the supports, and prestress steel area 11 064 mm² over the remainder of the beam. The prestrain was kept at 0,0056 and all other values were kept the same as for Appendix E. The new factored dead load was 143,5 kN/m.

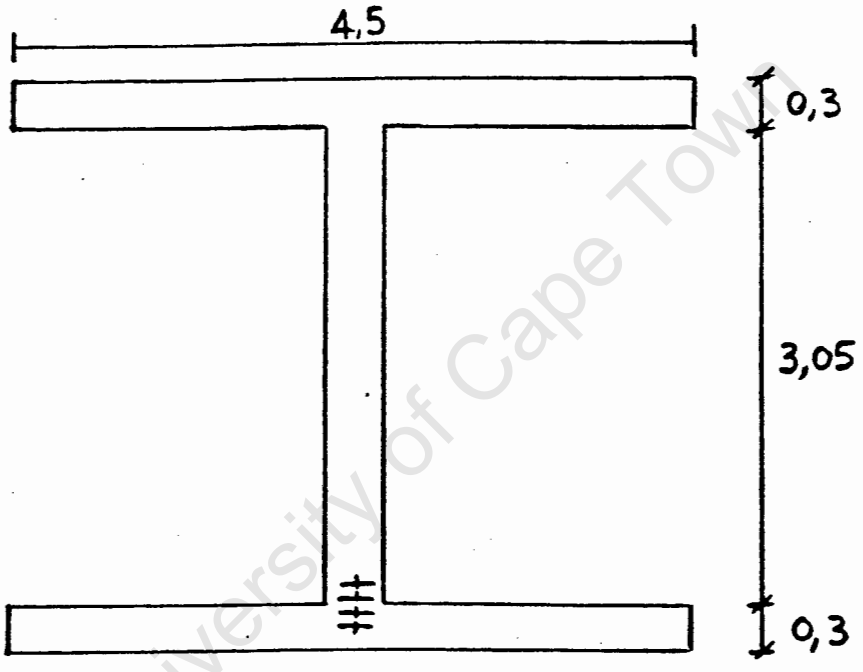



Fig F.1 Beam cross-section.



a) Considering real material relationships, it was found that a maximum load of 203,5 kN/m caused the strain capacity at the supports to be reached. In this case the strains in the steel proved to be critical. The strain in the prestressing steel due to the loading reached 1%, as did the strains in the reinforcing steel at the supports. At this stage the strain in the bottom fibre of the concrete at the support was only 0,096%, with 0,35% being the maximum allowable value. The redistribution of bending moments was found to be 17,18%. The capacity at the centre was 56,1 MNm, while only 50,25 MNm was reached. The cracking moment

at the centre was 49,4 MNm. The capacity at the supports was 61,06 MNm.

b) The above example was also investigated using design material relationships. A load of 170,3 kN/m caused the strain capacity at the supports to be reached and the bending moment redistribution was found to be 17,1%. The capacity at the centre was 53,15 MNm, while only 45,545 MNm was reached. The cracking moment at the centre was 45,11 MNm. The capacity at the supports was 55,39 MNm.

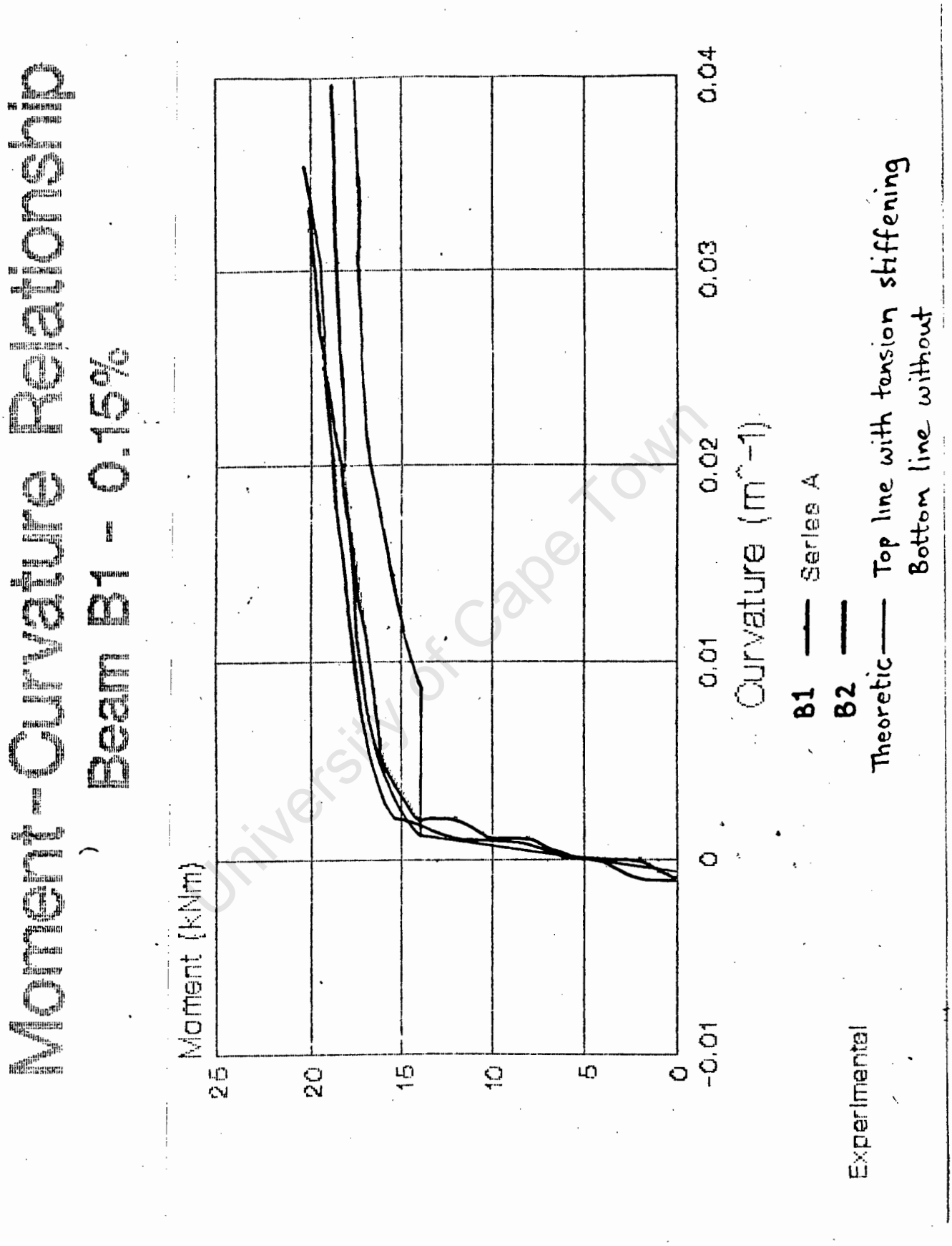





**Fig C.4 Moment - curvature relationship for 0,15% steel examples**