A Short Introduction to Structural Analysis

Algis Kabaila

2009-04-18

Contents

0.1 Preface

It is usual and seems logical to present the subject in the sequence of *axioms and/or assumptions*, followed by *theorems* and then the *applications*. However, it is my intention to present the subject of Structural Analysis in the sequence of discovery - from particular to the general. Consequently, in the first instance we look at the analysis of the simplest form of structures - *trusses*. Next part deals with two dimensional frames, then some generalisations, followed by three dimensional structures.

In all of this investigation I lean heavily on a wondrous programming language, *Python*. There are many useful references on Python. So much so, that sometimes it seems that all young computer science graduates are writing a book - their first real book - on Python. Rather than reinvent the wheel, we will refer the reader to numerous suitable sources. However, we will present full working programs on most aspects of our subject.

The programs are listed in these pages and their latest versions are also saved in archival form for downloading. You will notice that wherever possible I use free, open source tools. Fortunately, that is still possible in most problems of Structural Analysis. I hope that the reader will benefit from the ready availability of various tools and will come to like the idea of free exchange of information, free exchange that has been a major force in most learning institutions for many hundreds of years.

0.2 Thank You!

Many people had contributed to the development of the author of this book and thereby to the book itself. Foremost my parents who raised me in Lithuania and instilled in me the love of learning. My mother studied mathematics in 1920's, at a time when that was an unusual thing to do for a young married woman. My father during the First World war became an officer and remained in the military after the independence wars of Lithuania. Formerly a teacher of a primary school, he never had a chance to formally graduate from a high school. Yet he helped all his children, including me, to do their homework, particularly in maths. My mother preferred to excuse herself from that demanding task, claiming that she had forgotten it all. My elder sister and younger brother excelled academically - he becoming a professor of mathematics at the university of Vilnius, she a senior language teacher and a writer. Yes, I owe a big debt of gratitude to my parents, siblings and the country where we were brought up, Lithuania.

My wife and I have come to love Australia, which has been our home

longer than half a century. It provided us with shelter when we really needed it. It also provided us with many opportunities to learn, to work and to teach.

My wife of more than 60 years stood by me during the long years of study, happily accepting the lower income in order to allow me to work at a university. I would not have been able to do anything, let alone write this book, if I had not had her support over the years.

Stan Hall, the coauthor of the "Basic Concepts of Structural Analysis", was my teacher, my boss and later a senior colleague. He took some unorthodox steps to get me admitted to a graduate course of Master of Engineering Science at the University of New South Wales. This enabled me later to take up research towards a PhD at the same university.

Stan was my supervisor. As a coauthor of the old book, he always remained true to his principle of never trusting any theory, unless it could be proven to him *to his satisfaction.* He gave me hard time for seven years "drilling holes in me", in the words of one of our colleagues. I think that we both, Stan and I, through our extended discussions gained a deeper understanding of the fascinating subject of structural analysis.

Our eldest son Paul, a professor at the La Trobe university in Melbourne has been responsible for kindling my interest in LaTeX, the markup language used to typeset this text.

Professor Shigeharu Takeno of the Niigata Institute of Technology has provided most valuable and prompt advice on the configuration of latex2html program. A Fellow Australian, Ross Moore of the Macquarie University, who looks after the latex2html program, has given invaluable advice and was most generous with his time. I hope to use latex2html for translation of LaTeX, in order to display this text on the web in a user friendly format.

There are many others who helped me along the way and I regret not being able to mention them all in these lines. I hope that the numbers of people mentioned in the "Hall of Fame" will grow as this text becomes known among the teachers and particularly among the students. I will wait for your feedback and hope to include your name amongst the others already there.

Lex Mulcahy, who attended my lectures many years ago, has submitted a raft of corrections which have improved the text substantially.

I would like to mention a fellow academic Jorge Lampurlanes Castel of the Departament d'Enginyeria Agroforestal Escola Tcnica Superior d'Enginyeria Agrria Universitat de Lleida (Spain), who has come into the project much later, but whose interest has given me another burst of energy to do some programming that would help him and his students in their research.

Last but not least, a former colleague, Damian McGuckin of Pacific Engineering Systems International has helped to keep the lights of this project

burning by his encouragement of it.

I am sincerely grateful to all who have contributed to this manuscript. Grateful more than I can express in words.

0.3 SMAP

SMAP is an acronym for Segmented Matrix Analysis Package. It was developed in 1983 by three enthusiastic young persons, all on the lecturing staff of the Department of Structural Engineering at the University of New South Wales: Dr Ray Lawther, Dr Francis Tin Loi (now a Professor) and I, the author of this text. As the oldest of the "three musketeers" I assumed the role of the leader and the copyright holder. Actually, I doubt if this was really fair, as each of us worked independently, though in a close knit team. We often consulted with each other, and probably discretely competed with each other, but competed frankly and fairly. It was all a work of love - there was no reward other than the knowledge that SMAP was an achievement that not only deepened our knowledge of the subject, but also gave others a useful tool, particularly for learning about analysis of structures. Our hardware was limited, to say the least. We had a computing laboratory, equipped with *Apple II* mini computers. If I remember it correctly, the mini computers had a "big" RAM - 32 kB each (not MB!). They were networked to a common hard disk of about 10 MB (or was it 1 MB?) capacity. That was wonderful, as the only other storage of programs was on cassettes of magnetic tape, which were costly, slow and unreliable. Of course that was at the time when a Pascal compiler and its basic interpreter did fit on a 360 kB floppy (and at the time we even did not yet have floppies with the Apple II). Nevertheless, we could take work home in a cassette and do a bit of weekend hacking on private Apples.

What we developed was not unlike today's tools like, say, *scilab*. As all of us were structural engineers, we adapted it to structural analysis. The program was very versatile, even by more modern standards. Because of the RAM limitations, a great deal of ingenuity was needed to minimise the memory requirements. Hence, the program was segmented, the fact reflected in the name of the package. The largest example that we solved was a rigid three bay frame for a 20 storey building - substantial problem for that time!. Even now, I found it advantageous to use some SMAP examples in this book, as they cover many facets of structural analysis.. To me SMAP was a wonderful experience: both my companions in this undertaking were exceptionally gifted people. Ray practically single handedly wrote programs for *skyline* solution of equations, using a vector storage in a very compact

format. Being a real individualist, he wrote the way he liked: all programming in capital letters, regardless of the fact that Pascal language is not case sensitive. He liked global variables, which was not my preferred method of work. However, his methods and programming were faultless. Francis and I worked particularly closely. Not only our programming style was similar, but also in lunch break we often played chess. He invariably beat me at it! I am grateful to both Ray and Francis, for their cooperation which left an indelible stamp for the rest of my life.

It was one project that in my long and varied career I enjoyed more than any previous or subsequent project. It was a good example of cooperative effort. As we did not hide the code from each other, it was, in a limited sense, an Open Source project.

The experience of working with like-minded people remains deeply entrenched in my memory and reinforces my belief in the advantages Open Source Software. One of the reasons for undertaking this task of writing this text is to present to wider audience some of the ideas that germinated during the production of SMAP.

Chapter 1

Introduction

1.1 What?

Structural Analysis traditionally was the core of Civil and Aeronautical Engineering, including Structural Engineering. Though Civil Engineering has undergone a significant transformation with many new areas of specialisation, the simple fact remains, that no bridges, and therefore transportation, no tall buildings, and therefore cities could be built without Structural Analysis. An introduction to the modern version of Structural Analysis, making full use of personal computers, is the subject of this work.

It would be helpful if the reader had the basic knowledge of Statics and Mechanics of Solids, but we will try to make the subject matter simple enough for anyone with a high school education to be able to follow the main ideas of Structural Analysis.

We will pay special attention to the formation of algorithms, for that is the core requirement of modern, computer aided analysis. We will use Matrix Algebra as this greatly simplifies a systematic approach to the subject.

1.2 Why?

Why have I embarked on this journey, writing a short book at the age of well over 80? Well, I am still alive and have some of my marbles intact. Recently I tried some of the old structural methods that were second nature to me. Lo and behold, I could not remember them and had to derive it from scratch. So I better write some notes about it!

Why do I think that anybody would want to read it? Together with my senior colleague and my boss, Emeritus Professor A S Hall, we wrote a book on structural analysis. I believe we gained a good grasp of the subject after

arguing about it for some 7 years. We also produced a good textbook that was well received by students and teachers alike.

That book - "Basic Concepts of Structural Analysis" is now out of print. Also, things have changed dramatically in the computer world. An average desktop PC has more power than our university central computer of yesteryear. Computer that served the whole campus and cost well over a million pounds. New programming languages have matured. With the increased speed, interpreters became more attractive, and competive with compilers. This requires a fresh look at the subject. However, the layout of the subject was and remains - analyse the simplest of structures - trusses - and then show the similarity of analysis with the more complex structures.

The original book did not have a single computer program listing, though the analysis that it covered was computer oriented. In computer use we always were aware of the "principle" of *Garbage in, garbage out.* So we need computer programs that allows us to practice structural analysis. Hand calculation is too hard and too time consuming.

There is no substitute to understanding of analysis. One should never just feed the numbers in and hope that the computer analysis is correct and will give us the right answers. How do we know it does? So we need to learn how to verify the results. The results of analysis can be verified by a separate procedure to check equilibrium and geometric compatibility. And we intend to emphasise those aspects.

Hopefully, most of our readers will become sufficiently interested in the subject and will wish to write their own structural analyses programs. There is no substitute for this to achieve a better understanding of the subject. It is to these keen readers that I want to address this text. To our aid comes the wonderful programming language, *Python.* It is useful, it is elegant and, most importantly, it is fun!

1.3 How?

In the first part of this text we will deal exclusively with trusses. The deformation of members of the truss, pin-ended bars, are the simplest component of any structure. Member forces are defined by a single quantity - axial force. Likewise, member deformation is defined by a single quantity - bar extension.

In the second part the same methods are extended to the analysis of plane frames, frames in two dimensions.

We then dive into generalisation of the principles and methods already covered for particular types of structures.

Finally, we apply those methods to the 3 dimensional structures. The

principles remain the same, albeit the algebraic and numerical work is much more extensive. After all, a member stiffness for a two dimensional frame member is a 6×6 matrix with 36 individual terms, whilst in three diemensions the corresponding terms are in a 12×12 matrix with 144 individual terms - four times as many!

Chapter 2

Trusses

2.1 Why Pin Jointed Trusses?

The reason that we choose to analyse pin jointed trusses is simple - it is the simplest structure possible, so the analysis is also simple. So we can explore the methods of analysis without having to spend much time and effort. Once the methods for this simple structure are mastered, it is easy to extend it to more complex structures

Why pin jointed trusses when most trusses are not pin jointed at all? Just look at their welded joints! Traditionally pinjoints are assumed as an approximation. The experience shows that the forces in a pin jointed truss approximate those of the corresponding real truss reasonably well.

If all truss members are joined to the structure by pin and the lines of the centroids of the cross-section of all members intersect at the respective pins, then the truss members are subjected only to axial forces and no bending at all. And that accounts for the relative simplicity of their analysis.

We stipulate that the member centre lines pass through the centroids of their members. It should be noted that the approximation of a real truss by a pin jointed truss depends on all truss member centrelines intersecting at one point at each joint. That point is the imaginary pin of a pin jointed truss that represents the real truss.

2.2 Statically Determinate Trusses

2.2.1 What is Statically Determinate?

If a structure can be solved by equations of statics alone, it is said to be statically determinate.

It is sometimes stated that a truss is statically determinate if

$$2 \cdot j = m + r, \tag{2.1}$$

where j is the number of joints and m is the number of members and r - the number of reaction components. This is not necessarily so, as it depends on how the members are arranged as well as on the support conditions. It is possible for parts of the structure to be statically indeterminate and for other parts to be a mechanism. Not the usual arrangement. Arrangement which is useless for structural purposes, and yet it may satisfy the above equation.

In summary, we can say that the above condition is necessary, but not sufficient to assure that a truss is statically determinate and firm.

In many structural problems it is possible to have a fair idea about it by what may be called *intuition.* Intuition is not always reliable, but it should not be scoffed at, particularly when the rigorous answer is not easy to find. This is the case in determining if a truss is statically determinate or not. Every structure is constrained against a motion in several directions. If the removal of one such constraint changes the firm, load resisting structure into a mechanism which would deform without any resistance against deformation, it is likely that the structure in its firm state was statically determinate. It is probably because of this that additional constraints that render a structure statically indeterminate are often called *reduntants*, or a little more accurately, *redundant constraints.*

Please do not assume that in this context *redundant* means useless! Far from it.

2.2.2 Simple Trusses

Simple trusses can be solved by considering equilibrium of each joint. By judiciously choosing the joints, the equilibrium equations are *uncoupled* with two equations at each joint and two unknowns. This leads to simple computation that lead to solutions requiring little more than hand calculation, or graphical resolution of forces.

Here is a little practical demonstration of a statically determinate truss. Notice that the member numbering seems 'higglety-pigglety' - simply because we will use the same diagram later for statically indeterminate truss and it is convenient to preserve the notation on the diagram.

Considering equilibrium of the whole structure and taking moments about support A, we find the reaction at G as $133\frac{1}{3}$ kN. Similarly, moments about G, give the reaction at A as $116\frac{2}{3}$ kN. Now at joint G there are two unknowns and two equations of equilibrium. Considering sum of horizontal forces it is

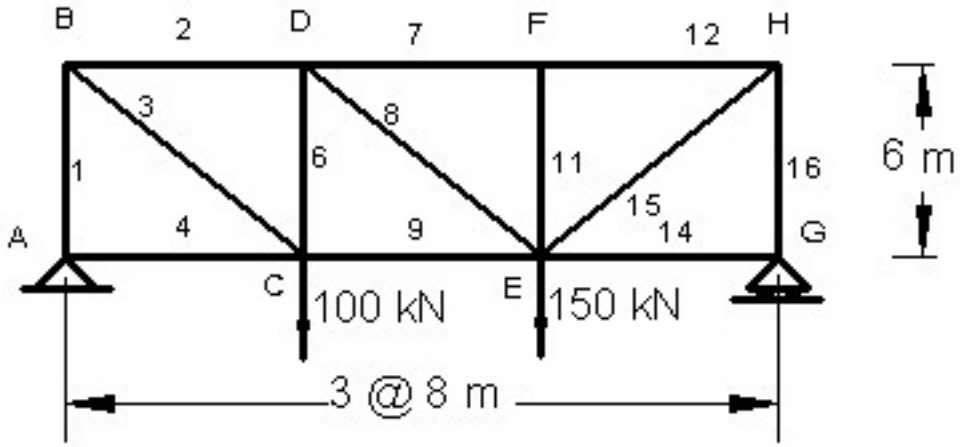

Figure 2.1: Simple Statically Determinate Truss

seen that member 14 carries no load. Vertical equilibrium of pin at G gives the force in member 16 as $-133\frac{1}{3}$ kN. The negative sign indicates that the member force causes compression. It is convenient to denote the force in member 16 by N_{16} and similarly forces in other members by N_i, where i is the member number.

Now joint at H again has two unknowns and two equations: sum of vertical forces yields $N_{15} = 222.22$ kN and the sum of horizontal forces - $N12 = -177.78$ kN. It is now possible to similarly deal with joint *F, E, D, C*. Equilibrium of joints *A, B* then gives three check conditions, since we have used three conditions of equilibrium of the structure as a whole.

But wait, we only took moments about A and G – that's only two equations. Where was the third equation of the equilibrium of the whole structure? Well, the horizontal equilibrium of the structure as a whole sneaked in through the back door, did it not?

The results for all member forces are recorded in a table 2.1. The reader is urged to verify that the table shows correct results - a check is instructive. Any mistakes in it are deliberate ;-). (An unmitigated lie...)

That's one of the ways that statically determinate trusses can be solved by hand calculation. For computer analysis it is not a convenient way to tackle the problem, as the algorithms are not well developed. Generally, more universally applicable methods, such as stiffness analysis, are employed.

Table 2.1: Member forces for truss of Figure 2.1

Member	Force kN	Member	Force kN	Member	Force kN
1	-116.67	2	-155.56	3	194.44
4	0	6	-16.67	7	-177.78
8	27.78	9	155.56	11	0
12	-177.78	14	0	15	222.22
16	-133.33				

2.2.3 Displacements

In Physics, one learns that the 'work done is the product of force times the corresponding displacement'. Let us consider a linearly elastic, vertical spring on which we can drop a load that produces a gravitational force P. If the load is put onto the spring gently, so that the spring deflects without vibration, it causes a displacement, say v. The strain energy, stored in the spring is then $\frac{1}{2}Pv$. But we have just heard that the work done is Pv. Confusing? Probably not. But wait, if we drop the load from zero height onto the spring, it will deflect roughly twice the amount v. The spring begins to vibrate with an approximate amplitude of $2v$ and would continue to do so for ever, were it not for the damping effects that dissipate the energy by friction,air resistance etc. Just think, if there were no damping, the spring would vibrate for ever! Actually, if there were no damping, everything would be vibrating with forces induced at the creation of the world. Cute, no? And yet. in the analysis one invariably starts by ignoring, at least temporarily, the effects of damping.

Also, if the spring is not linear, the strain energy is no longer $\frac{1}{2}Pv$. If the spring is partly plastic and undergoes plastic deformation, the strain energy will again differ greatly from that of an elastic spring.

The frustrating aspect of this possibly confusing mix of various factors of Pv is that in many, many instances the product of Pv is of a fundamental importance. So much so, that it is worth while to give it a separate name, we call it *the Work Product*. It is best to regard it as an abstract quantity. The definition of this abstract quantity for a set of forces is very simple indeed.

Definition of Work Product: The Work Product is the sum of Forces multiplied by their corresponding Displacements.

Let us now look at a simplest possible truss - a two bar structure. The intention is to examine the relationship between bar extensions and joint displacements. Do not be deceived by the simplicity of the problem, it will reveal some fundamental principles of structural analysis. The truss is shown

in the figure (2.2). A bar extension is taken as positive. Similarly, bar force is positive if the bar is in tension.

Our aim is to determine the displacement of point B due to to bar extensions (or contractions), denoted by e_1 and e_2. The displacement of point B will be calculated in several ways. Easiest to understand, but not necessarily the easiest to calculate is to do it from the first principles of geometry.

The geometry of the problem is defined in figure (2.2) (a). The effect of e_1. is shown in figure (2.2) (c). As a result of elongation of bar 1 by e_1, point B moves to B'. We limit our considerations to *small displacements.* if bars 1 and 2 were disjointed at B, the bar 1 would extend horizontally by the (small) distance e_1. To bring both bars together again, without any other extension or contraction, the bar 1 can rotate about the point A and the bar 2 - about the point C. As the displacements are small, the arc of rotation can be represented with adequate accuracy by a tangent to the circle at the point B'. This gives rise to a displacement triangle, which is similar to the triangle ABC. The horizontal displacement is denoted by u and the vertical displacement by v, as shown in the figure.

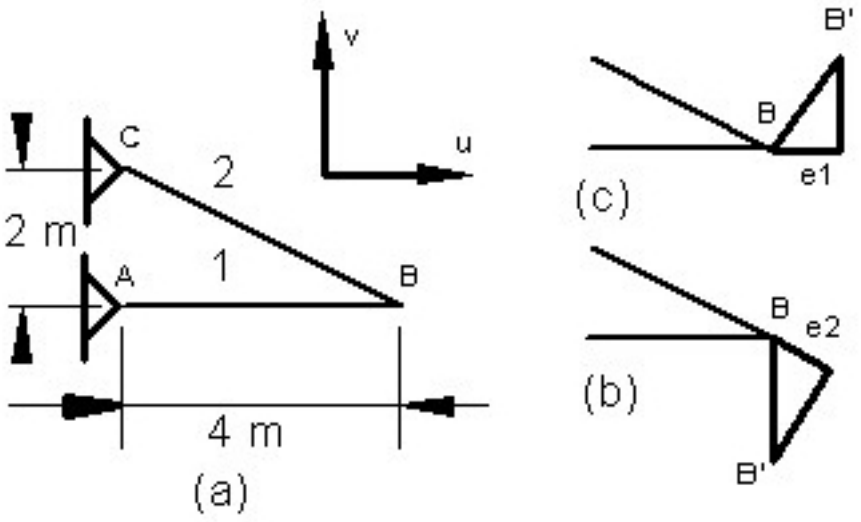

Figure 2.2: Two Bar 'Truss'

By similar triangles, $u = e_1$ and $v = \frac{4}{2}e_1$.

Similarly, by considering the displacements, shown in figure (2.2) (b) we can calculate the horizontal displacement, u, and the vertical displacement, v, of joint B due to the elongation of bar 2. Since we are considering the extension of bar 2 only, there can be no movement of joint B in the horizontal

direction. In the truss, the distance $\bar{BC} = \sqrt{4^2 + 2^2}$, which is approximately 4.472 m, so that the $v = -\frac{4.472}{2}e_2$. The negative sign indicates that due to e_2, B moves downwards. Combining the effects of both extensions, we finally obtain by consideration of geometry alone, the expressions for the horizontal and vertical displacement of joint B.

$$u = e_1 \tag{2.2}$$

$$v = \frac{4}{2} \cdot e_1 - \frac{4.472}{2} \cdot e_2 \tag{2.3}$$

Example: If the bars suffer extensions $e_1 = -2\,mm$ and $e_2 = 2.236\,mm$ what is the horizontal displacement, u, and the vertical displacement, v, of joint B?

Solution: $u = -2.0\,mm$
$v = \frac{4}{2} \times (-2) - \frac{4.472}{2} \times 2.236 = -9.0\,mm$

2.3 Virtual Work

Whilst the calculation by geometry of the displacement for the very simplest truss of Figure (2.2) is simple enough, it can get quite complex for real life trusses. A much more systematic approach is by *Virtual Force*. Some further definitions are required before we can state the principle of *Virtual Work* and then *Virtual Forces* as well as *Virtual Displacements*. Fortunately these definitions are really simple to state and simple to apply once the initial shock is overcome. It is necessary to refer collectively to all forces acting on a given truss: the applied loads, the reactions and the internal forces that the pins transmit onto the members. We call this collection of forces a *Force Field.*

When all forces are in equilibrium with reactions and the forces by the pins onto the bars are in equilibrium with the forces that the bars exert onto the members, this collection is an *Equilibrium Force Field.*

The displacements of a truss and all bar extensions or contractions are defined by the displacement of the pins. We call this collection a *Displacement Field.* When the displacements satisfy all conditions of geometry, they constitute a *Geometrically Compatible Displacement Field.* Unless noted otherwise, we will refer to Equilibrium Force Field simply as the *Force Field.*

Similarly, we shall refer to Geometrically Compatible Displacement Field simply as *Displacement Field.*

We have defined the *Work Product* as the sum of forces multiplied by their displacements. We shall separate the Work Product into two parts, *The Internal Work Product,* denoted by U and the *External Work Product*, denoted by W. The forces exerted onto the bars by the pins cause the internal work product. Since each bar has an axial force N, which acts at both ends of the bar, but in opposite directions, and the extension of the bar is given by the difference of the displacements measured in the direction of the bar at its two ends, each bar contributes $N \cdot e$ to the internal work product. This is expressed by the following:

$$U = \Sigma N \cdot e \tag{2.4}$$

Similarly, if all the forces, including the reactions, are denoted by P, and their corresponding displacements as u, the equation for the external work product can be written as follows:

$$W = \Sigma P \cdot u \tag{2.5}$$

It is fairly obvious that the work product of an equilibrium force field and a compatible displacement field must be zero. Some clarification of the internal force field, however, is necessary.

Let us denote the internal force field that the bars exert onto the pins by $\bar{U}'$. Then the above paragraph of describing equilibrium in terms of work product can be written as

$$W + U' = 0 \text{ or } W = -U' \tag{2.6}$$

Since the forces exerted by the pins *onto the bars* are in opposite direction, but of the same magnitude as the forces of the bars onto the pins, $U' = -U$ and, therefore,

$$W = U \tag{2.7}$$

This is the *Equation of Work Product* employed throughout this text.

Basically, the rather long winded talk about U' and U is an attempt to underline the different forms of possible representation of internal work product, each the same, but with different sign. Which one is used is largely a matter of taste - in this text the Equation (2.7), which equates the internal work product with the external work product, is used.

2.3.1 Virtual Forces

Suppose we have a real displacement field and a virtual, i.e. imaginary, force field, an equilibrium force field. To stress that the force field is imaginary, let the virtual forces be indicated by a bar over them, viz. $\bar{P}$ and $\bar{N}$. Equation (2.7) applies to this real displacement field and an equilibrium force field. The resulting expression is called the *Principle of Virtual Forces*, which can be written as

$$\Sigma \bar{P} \cdot u = \bar{N} \cdot e \tag{2.8}$$

The virtual force equation can be applied to solve the geometry problem of our two bar truss example, shown in Figure (2.2). The question of the problem is restated below:

Example: If the bars suffer extensions $e_1 = -2\,mm$ and $e_2 = 2.236\,mm$ what is the vertical displacement, v, of joint B?

Solution: It is convenient to apply a vertical virtual unit force at point B. The units of that force are immaterial. To be less abstract, let it be 1 kN. As the force field must be an equilibrium force field, we resolve the upward unit force into directions of bar one and bar two. This yields the virtual bar forces as follows:

$$\bar{N}_1 = \frac{4}{2} \text{ and } \bar{N}_2 = -\frac{4.472}{2} \tag{2.9}$$

Substitution into (2.8) yields

$$1 \cdot v = \frac{4}{2} \cdot (-2) - \frac{4.472}{2} \cdot 2.236 \tag{2.10}$$

Dividing both sides of the equation by 1 kN, yields the vertical displacement, $v = -9.0\,mm$, which, not very remarkably, agrees with the result obtained directly by geometry. What, however, is more remarkable, is that we have solved a problem of geometry of small displacements by statics *alone.*

2.3.2 Virtual Displacements

It is evident that if we can determine small displacements by equations of statics, the reverse is also possible - we can solve equations of statics by geometrical considerations of small displacements. This procedure is called the method of *Virtual Displacements.* In some problems, for instance to calculate influence lines, virtual displacement method is quite useful.

In the two bar structure, we can determine the forces in all bars by considering virtual vertical and virtual horizontal displacements of point B, calculating by geometry the bar extensions and then equating the resultant internal work with the external work.

This gives rise to two equations which are fully equivalent to the equations that arise from summing up the horizontal forces and then the vertical forces at joint B.

2.3.3 Complex Trusses

Most practical statically determinate trusses are simple trusses. Occasionally, however, one encounters a peculiarly arranged truss which, whilst statically determinate, has a set of simultaneous equilibrium equations which can not be readily uncoupled. These trusses are called *Complex Trusses.*

Complex trusses lend themselves to a simple computer solution with algorithms resembling those of the more general stiffness analysis of structures. Since all trusses, statically determinate and indeterminate can be solved with equal ease with the stiffness method, there is no real need to cover special methods for statically determinate trusses. The author of these lines has written a Python program for the Complex Trusses which, of course, can also solve simple trusses. If there is an expressed interest in it, I will be happy to add it to this text.

2.3.4 Relative Displacements

The need to calculate relative displacements arise in truss analysis quite often. In the flexibility solution of trusses, it is often required to calculate bar extensions due to the displacements at the end pins. This is precisely the relative displacement of those pins.

It would be rather clumsy to calculate the two displacements in the direction of the bar separately and only then to calculate the extension by calculating the difference of these two displacements. It is much more convenient to apply the two virtual (usually unit) forces simultaneously. This is best illustrated by an example.

2.4 Flexibility Solution of Trusses

The flexibility solution of statically indeterminate trusses is not favoured in computer programs. On the other hand, if hand calculation is considered, the flexibility solution can well be more convenient. Furthermore, the ideas

of flexibility solution are sufficiently important for a better understanding of the problem of dealing with statically indeterminate structures. Also, use of flexibility formulation is most convenient for a truly independent check of stiffness solution, which is the subject of the next section.

To save effort and space, I will limit the discussion of the flexibility solution to trusses with one redundant bar only.

Some explanation of what is meant by *redundant bar* is desirable.

The quickest and probably clearest way to explain the flexibility solution is by an example. A computer program for the flexibility solution is not worth developing and the favourite tool for this task is the spread sheet.

Chapter 3

Truss Stiffness

3.1 Stiffness Solution of Trusses

3.1.1 Member Stiffness

In this section we begin our journey to the stiffness analysis of trusses, with other types of structures in the following chapter. The stiffness solution of a structure starts with the determination of stiffnesses of the individual members. These member stiffnesses are then assembled into a structure stiffness. The loads of each load case are assembled into a load vector. The relationship between the load vector, the structure stiffness and the structure displacements yields the *Structure Stiffness Equations*. The solution of these equations yields the values of all displacements.

Once the displacements are determined from the Structure Stiffness Equations, the displacements of each member are extracted and the forces in individual members determined.

3.1.2 Member axes

It is convenient to analyse member properties in *member axes*, which pass through the centroids of each member. The Figure 3.1 shows a member with *member axes* $\bar{x}$ and $\bar{y}$. Member axis $\bar{x}$ runs from end1 to the end2 of the member. We are free to decide which end is the end1 - it is simply a matter of convenience. The $\bar{y}$ axis runs at right angle to the $\bar{x}$ axis and is so chosen that in the right angle coordinate system, the invisible axis $\bar{z}$ always points towards the viewer. Of course, it is at the right angles to the paper (or the computer screen, as the case may be). The axes follow the right hand rule: when x is rotated towards y a right hand screw would move in the direction of the z axis; if y axis is rotated to z axis, a right hand screw thus rotated,

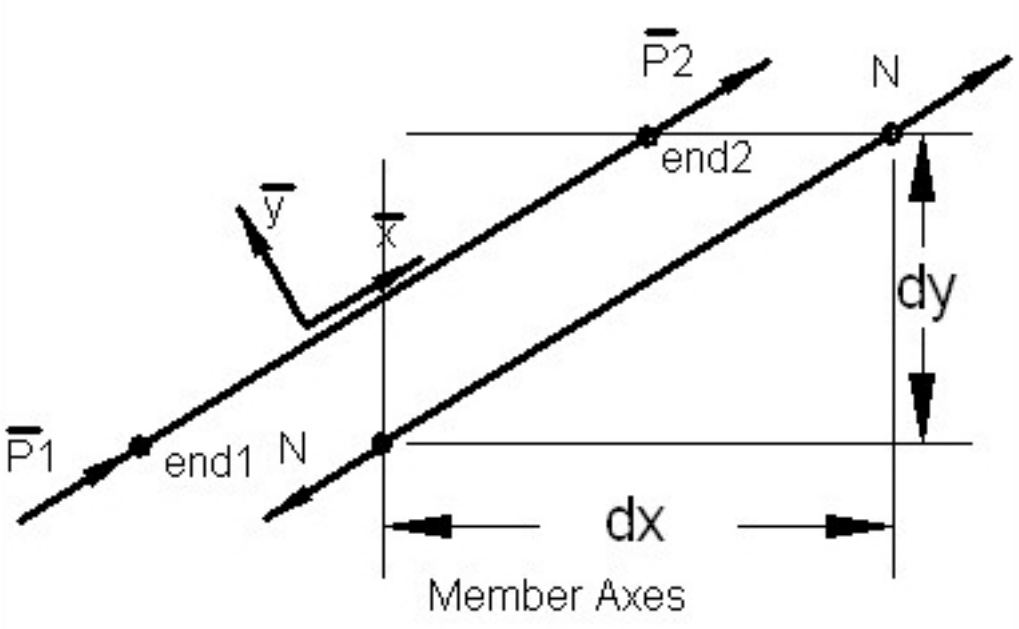

Figure 3.1: Member Axes

would move in the direction of the x axis; if z axis is moved to the x axis a right hand screw rotated in the corresponding direction would move in the direction of y. The forces, applied on the member, $\bar{P}_1$ and $\bar{P}_2$ are taken to be positive when they act in the same direction as the $\bar{x}$ axis. The tension in the bar, N actually is made up of two forces, which are in equilibrium and therefore act in opposite directions.

The orientation of the bar is defined by its projections on the structure x and y (horizontal and vertical) axes and are denoted by dx and dy. The length of the bar is then given by $\sqrt{(dx)^2 + (dy)^2}$.

As the ends of the member are pinned to the truss as a whole, the force at each end must pass through the pin. We postulate a sign convention that defines tension in the member as positive. Therefore a positive member force N at the end 2 of the member acts in the direction of $\bar{x}$ whilst at the end 1 - in the direction opposite of $\bar{x}$.

First of all, we will consider the member held in position by the pin at end 1. The stresses in the member are uniform of magnitude $\sigma = \frac{N}{A}$, where A is the area of the member. The member extension, e is simply

$$e = \frac{L}{EA} \cdot N \tag{3.1}$$

where E is the *Modulus of Elasticity*, also called *Young's Modulus*. L is the member length. The term $\frac{L}{EA}$ is called the *flexibility* of the member. The inverse of member flexibility is the *end stiffness* of the member. In terms of member end stiffness, the *force-displacement* relation becomes

$$N = \frac{EA}{L} \cdot e \tag{3.2}$$

where $\frac{EA}{L}$ is the *member end stiffness in member axes.* We denote this stiffness by $\bar{k}$. Generally, $\bar{k}$ is a matrix, but for a pin jointed member it is a scalar, or a (1×1) matrix.

3.1.3 "Global" Member Axes

In order to ensure the equilibrium of each joint, it will be necessary to augment forces of all members meeting at any one joint. For this reason we will need to work in axes for each member that are parallel to some agreed axes, which we refer to as *global axes* or, alternatively, as *structure axes.*

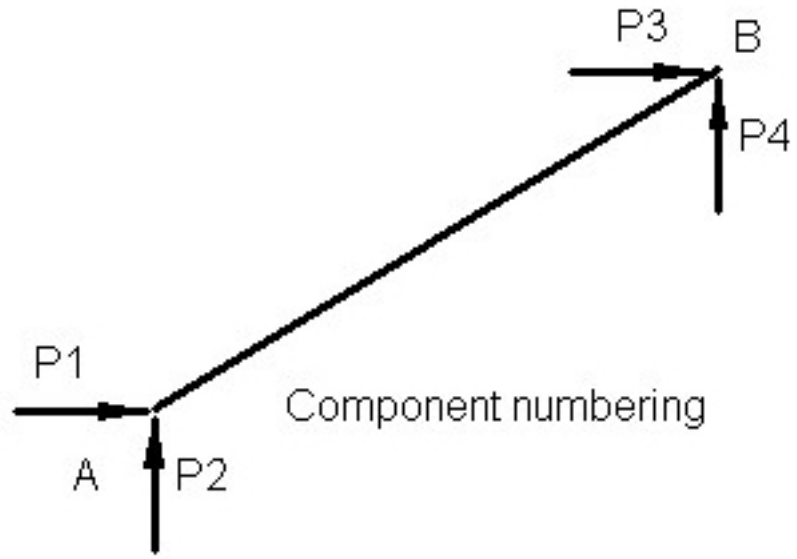

Figure 3.2: Component Numbering

The diagram shows the numbering of forces in global axes for a member. In global axes the displacements of the member ends correspond to the forces.

First, consider the tensile axial force at the top end of the member. If we denote the angle between the member and the x axis by α and the angle between the member and the y axis by β, then

$$P_3 = cos\alpha \cdot N \quad and \quad P_4 = cos\beta \cdot N \tag{3.3}$$

The cosines of the member with the coordinate axes are referred to as *direction cosines.* and are useful in extending current work on two dimensional trusses to the three dimensional structures. In practice, it is convenient to express the direction cosines in terms of member projections onto the coordinate axes and member length L.

Noting that $cos\alpha = \frac{dx}{L}$ and $cos\beta = \frac{dy}{L}$ and that at end 1 the axial tensile force N is in the opposite direction to N, we can write the following expression for end forces in global axes x and y in terms of N as follows:
$P_1 = -\frac{dx}{L} \cdot N$
$P_2 = -\frac{dy}{L} \cdot N$
$P_3 = \quad \frac{dx}{L} \cdot N$
$P_4 = \quad \frac{dy}{L} \cdot N$
It is convenient to express these relationships in matrix notation. We denote the (4x1) (4 rows, one column) Load Vector as follows:

$$\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \tag{3.4}$$

And the (4x1) Transformation Matrix B as follows:

$$\mathbf{B} = \begin{bmatrix} -dx/L \\ -dy/L \\ dx/L \\ dy/L \end{bmatrix} \tag{3.5}$$

So in terms of this Transformation Matrix, member end forces in global axes are given, in terms of member forces in local axes N, as follows:

$$\mathbf{P} = \mathbf{B} \cdot (N) \tag{3.6}$$

For each force component there is a corresponding displacement component. The displacement components are so chosen that the work product is equal to the sum of every force P multiplied by the corresponding displacement u. In matrix notation that can be written as

$$\mathbf{P}^T \cdot \mathbf{u}$$

We now look at the end displacements. In the same manner as for end forces, consider first the displacements of end 2 in global x and y directions. If we look at the geometry of the bar end displacements, bearing in mind that they are small and we can approximate the direction of displaced bar by the direction of the bar before the displacement, by similar triangles we find that

$$e = (dx/L).u_3 + (dy/L).u_4$$

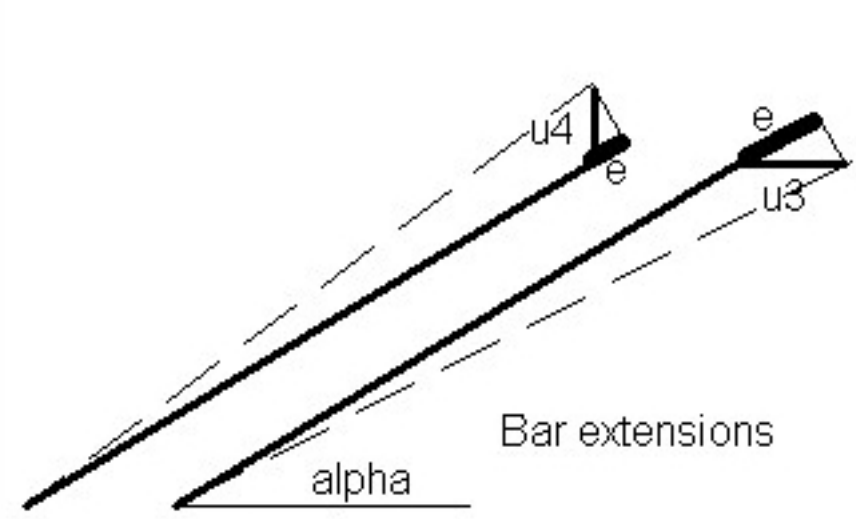

Figure 3.3: Bar Extensions

To get the bar extension due to displacements of both end nodes, we need to add the effect of the displacements at end 1 to obtain:

$$e = -(dx/L).u_1 - (dy/L).u_2 + (dx/L).u_3 + (dy/L).u_4$$

If we look at the above expression, we see that the terms of matrix **B** appear in it. They do appear in the same order, but in a row, rather than a column. The displacement componets u_1, u_2, u_3, u_4 written in a column are a (4x1) matrix **u**. In matrix terms the above expression becomes:

$$e = \mathbf{B^T}.\mathbf{u} \tag{3.7}$$

Comparing the equation for the force N (equation 3.6) with the equation for the elongation e (equation 3.7), we see a remarkable similarity. Later we will generalise this result for all types of structures in a section entitled "Contragredience". It is useful to record these two equations side by side:

$$\mathbf{P} = \mathbf{B} \cdot (N)$$

$$e = \mathbf{B^T}.\mathbf{u}$$

Notice that $\mathbf{P}$ corresponds to $\mathbf{u}$ as N corresponds to e. The second equation looks a kind of transposition of the first. That makes these equations easy to remember. We are now ready to develop an expression for member stiffness in global axes. Recalling the stiffness expression in member axes (3.2), we obtain from the first contragredience equation:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot e$$

.

Substituting the second contragredience equation into the above, we have:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot \mathbf{B^T}.\mathbf{u}. \tag{3.8}$$

Member stiffness in global axes, $\mathbf{k}$ defines the relationship between member forces in global axes and member displacements in global axes, as follows:

$$\mathbf{P} = \mathbf{k} \cdot \mathbf{u}. \tag{3.9}$$

So from equation (3.8) we have the expression for the member stiffness in global axes $\mathbf{k}$ in terms of the transformation matrix $\mathbf{B}$ and the member stiffness in member axes, $\bar{\mathbf{k}}$:

$$\mathbf{k} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot \mathbf{B^T}, \tag{3.10}$$

where the member stiffness in member axes is actually a scalar and is equal to $\bar{\mathbf{k}} = \frac{EA}{L}$ It is evident that the member stiffness $\mathbf{k}$ is symmetrical, as its transpose is equal to itself.

To see that we need to recall that in matrix algebra the symmetry of a matrix, say $\mathbf{X}$ - the symmetry about the leading diagonal - means that $\mathbf{X} = \mathbf{X}^T$. Also, a transpose of a product of matrices, say $\mathbf{X} \cdot \mathbf{Y}$, is equal to $\mathbf{Y}^T \cdot \mathbf{X}^T$.

Bearing this in mind, it is evident that $\mathbf{k}$ is symmetrical, provided that $\bar{\mathbf{k}}$ is symmetrical. For truss members, $\bar{\mathbf{k}}$ is a scalar, or a (1 x 1) matrix, and is thus symmetrical.

The form of the equation (3.8) is the same for most, perhaps for all structures, though the transformation matrix $\mathbf{B}$ and the member stiffness in member axes $\bar{\mathbf{k}}$ are, of course, different.

An alternative form is possible because of the simplicity of a pin jointed truss member. The force transformation matrix **B** can be written in terms of the angle of the member with the horizontal (global x) axis:

$$\mathbf{B} = \begin{bmatrix} -sin(\alpha) \\ -cos(\alpha) \\ sin(\alpha) \\ cos(\alpha) \end{bmatrix} \tag{3.11}$$

Which of the two forms for **B** one uses is largely a matter of taste, but the previous expression in terms of direction cosines is more general.

3.1.4 Assembly of Truss Stiffness Matrix

It has already been hinted that the member stiffnesses must be assembled into a structure stiffness. In addition to adding values of member stiffnesses in global axes, it is necessary to adjust the numbering into a numbering system of the truss as a whole. This particular task is accomplished by the *location vectors.* Each member in member axes will start at node no 1 and end with node no 2. Node no 1 will have forces P_1 and P_2 with their corresponding displacements u_1 and u_2, or simply *freedoms* 1 and 2. At the end node, the freedoms in member axes are 3 and 4. However, in the structure the same nodes will have different numbers and different freedoms. Suppose that these corresponding freedoms are 7, 9, 25 and 3. The two numbering systems are conveniently related by *location vectors.* All counting is done starting with 1 (This statement may sound strange, but it is significant, because numbering in the program starts from zero. So each node and each freedom has two kinds of numbers: a) *world* that starts with 1 and b) *program* that starts with 0. All data is prepared in world numbering and all output is in world numbering. World numbering is used in this text). The member numbers correspond to the *position* in the location vector, whilst the structure numbers of freedoms are the *values* at that location. So for our example, the location vector would be

$$loc = [7,\ 9,\ 25,\ 3]$$

The contribution of a member to the structure stiffness is obtained by adding to the structure stiffness matrix term in row loc[i] and column loc[j], the term $\mathbf{K}_{loc[i],loc[j]}$, the member stiffness term $k_{i,j}$ for i = 1 to 4 and for j = 1 to 4.

In programming terms, we can write this as:

$$\mathbf{K}[loc[i], loc[j]] = \mathbf{K}[loc[i], loc[j]] + \mathbf{k}[i, j] \tag{3.12}$$

for all members and for each member i and j looping for all i-s and all j-s. There is another point that needs to be stated. Without considerable modification and complication, the stiffness solution algorithm yields all member displacements. From the displacements, all member forces are then calculated. The *reactions* are determined by equilibrium of member forces and the reactions at the supports, since the corresponding displacements are set to zero. We do this by setting the location vector values to zero to correspond to zero displacements. The assembly algorithm then takes care of it by only assembling the contributions of the member stiffness to the structure stiffness if the location vector's value is not zero. This is best examined in the program listing. Below is the program snippet where this is accomplished. Notice that structure stiffness is denoted by Ks, location vector by locVec and the $i,\ j-th$ element of the member stiffness by k[i, j].

```
def assemble(self,k,locVec):
    Ks = self.Ks   # alias
    for i in range(4):
        if locVec[i] != 0:
            for j in range(4):
                if locVec[j] != 0:
                    Ks[locVec[i]-1,locVec[j]-1] += k[i,j]
    return Ks
```

If you look at the program listing, you will discover that there are other refinements that probably are less important, so they are not discussed here.

The assembly of member stiffnesses yields the structure stiffness $\mathbf{K}_s$. As member stiffness is symmetrical, structure stiffness is also symmetrical. It should also be noticed that its leading diagonal terms are all non-zero and positive. If either of the two conditions is not satisfied, then the structure stiffness is wrong, or the structure is a mechanism and can deform under infinitely small loads.

Structure stiffness relates structure nodal loads with the nodal displacements as follows:

$$\mathbf{P} = \mathbf{K}_s \cdot \mathbf{u_s} \tag{3.13}$$

Whilst we consider the nodal loads a vector and the nodal displacements as a corresponding vector, in practice we often have to solve the structure

for several load cases, viz, dead loads, live loads, wind loads and so on. This extension is achieved simply by adding the load vector for each load case to a load matrix, with the solution of equations yielding the matrix of displacements.

It is simpler to derive all the expressions for one load case only and take care of several load cases directly in the computer program.

3.1.5 Displacements and Forces

Truss Displacements

The structure stiffness matrix has many zero terms. For large structures it can be a very sparse matrix. This sparseness can be advantageously taken into account by the equation solving program.

The structure stiffness matrix is *positive definite*. In plain language, all the leading diagonal terms of the matrix are positive numbers. The matrix is also symmetrical, so that stiffness matrix elements are symmetrically placed along the leading diagonal, thus $K_{ij} = K_{ji}$

All these features, particularly sparsness of the stiffness matrix and symmetry, can be taken advantage of in the solution. The solution of sparsely populated matrices is a large topic, which deserves a separate treatise or treatises and is not discussed here.

The solution of truss stiffness equations (3.13) yields displacements of all pins in global axes $\mathbf{u_s}$.

For our purposes we simply use a precompiled module'numpy', 'Numeric' or 'mumarray', written mainly in C.

Member Displacements

Member displacements in global axes are extracted by the same *location vectors*. For any member, the i-th term of member end displacements is equal to:

$$u[i] = u_s[loc[i]] \tag{3.14}$$

These terms constitute the member displacement vector in global axes $\mathbf{u}$. In order to calculate member forces, we need to change those displacements into the local member axes. We do this using the contragredience equations, 3.7:

$$e = \mathbf{B^T} \cdot \mathbf{u} \tag{3.15}$$

Member Forces

Once member end displacements in member axes are known, we know member extension (or shortening) and hence the member force:

$$N = \frac{EA}{L} \cdot e \tag{3.16}$$

Stress Matrix

Stress Matrix is simply an alternative expression for internal forces in terms of element displacements. It is defined as matrix which post-multiplied by the element displacements yields stress resultants. In case of truss member, there is only one stress resultant, namely N. Substitution of equation (3.15) into the equation (3.16) yields the required expression:

$$N = \frac{EA}{L} \cdot \mathbf{B^T} \cdot \mathbf{u} \tag{3.17}$$

So we can formally write the expression for the *Stress Matrix* as:

$$\mathbf{StressMatrix} = \frac{EA}{L} \cdot \mathbf{B^T} \tag{3.18}$$

We usually write this as

$$\mathbf{StressMatrix} = \bar{\mathbf{k}} \cdot \mathbf{B^T} \tag{3.19}$$

where the member stiffness in member axes is equal to

$$\bar{\mathbf{k}} = \frac{EA}{L} \tag{3.20}$$

Summary

It may be useful to summarise the whole process in few lines:

- $\mathbf{k} = \mathbf{B} \cdot (\frac{EA}{L}) \cdot \mathbf{B}^T$
- Assemble all $\mathbf{k}$ to structure stiffness $\mathbf{K}_s$
- Assemble load vector, $\mathbf{P_s}$

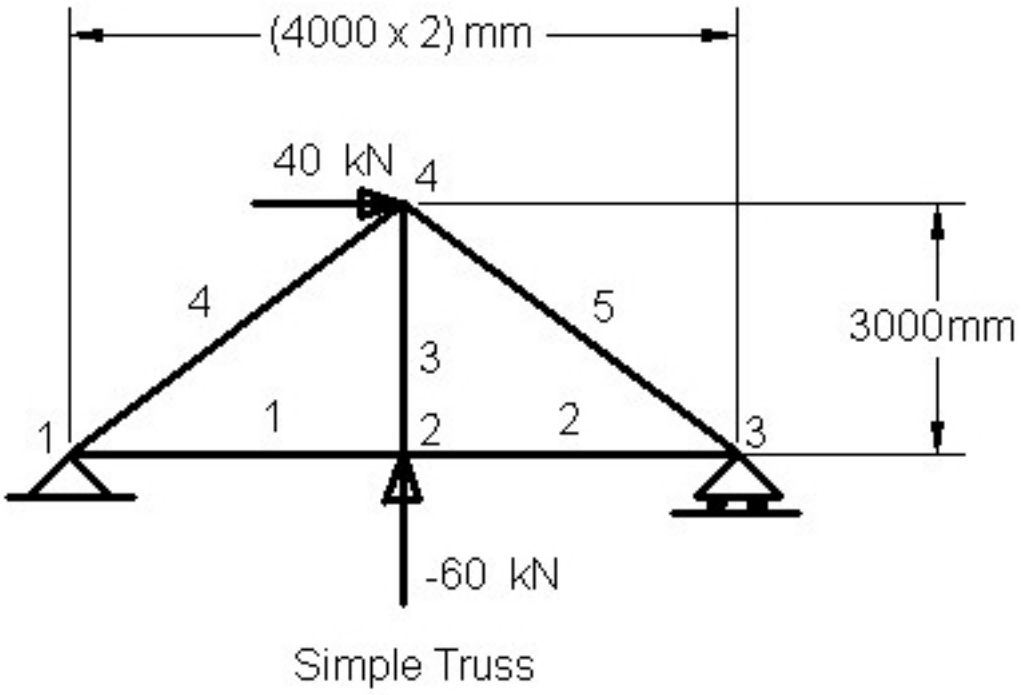

Figure 3.4: Simple Truss

- Solve stiffness equations, $\mathbf{P_s} = \mathbf{K}_s \cdot \mathbf{u_s}$
- Extract member displacements $\mathbf{u}$ from $\mathbf{u_s}$
- Find member forces, $N = \frac{EA}{L} \cdot \mathbf{B^T} \cdot \mathbf{u}$

3.1.6 Examples of Analysis using a Python Program

An example is presented in this section. It should be noticed that computer programs are an integral part of this text and that they should be read and understood. However, to help the reader, only the examples are first presented. Complete program listings are omitted, but are part of the tarball - not because the listing is less important than the text, but because for the first reading the fairly extensive listing may be an unwarranted distraction.

The program is simple, without any GUI. Data can be prepared by any plain text editor. I like "Kate" in Linux and would recommend "Notepad" in Windows. A very simple truss is chosen as the first example to show the solution of a structure using the program. This 5 bar truss is shown in figure 3.4.

We use for this our first example N (Newton) and mm (milimeter) units, though generally it is better to use kN (kilo Newton) and m (meter) units. Any consistent units can be used, however please notice that all data must be in the same units.

Initially, a different data scheme was used in the original development of programs to solve trusses or frames.

Once the concept of combination of trusses with frames was accepted, and fem.py was conceived, a new, more flexible format was adopted. Instead of a fixed number of lines for titles, the progam now ignores all lines with a # symbol. Consequently, any number of lines can be used as titles for different parts of the program.

It is necessary to to specify the data in a consistent way. To make it simpler to remember, we adopt a convention that each type of data begins with a header of any number of lines. Header lines and comments start with #. We specify rather artificially three load cases: first, with both loads on, next only with the vertical load on and finally, the third load case, only with the horizontal load. The program that starts the process of analysis is *fem.py*. In Linux and other Unix like operating systems, the program can be started either from IDLE or from the *Command Line Interface* - (CLI) as follows:

- 1 Change to the directory where fem.py is.
- 2 Type ,/fem.py

For Unix like operating systems, the second line can alternatively be

- 2 fem.py truss1

The data is in a subdirectory dat in a file truss1.dat. Note that only the root (truss1) needs to be given to the program. Of course, truss1 is chosen only as an example. One can equally well chose any other truss.

It is expedient and least error prone to list the full text of data for truss1 and then two describe each major part of it.

```
# truss1.dat - fem format.
# A small truss example-\label{fig:simpleTruss}} Fig. 3.4
   general
# noMaterial,noSection,noNodes,noMembers,noLoadCases,
# noPrescribes (= 0):
# I notice that the prescribed displacements are not
# specified here - it should be impementd to bring trusses
# in line with frames.
```

```
 g   1  1                4  5  3  0
# The second line specifies how many lines of input there will
# be for each load case.
  g  2  1  1
# repeat for beams and trusses:
# type  noNodes  noMembers
   b     0        0
   t     4        5
# Materials:
# no of material group,  Young's modulus, E only
 g 1                       200e3
# Section List (one section only in this example):
#    no area       ix
g  1 1600.0      1.0e-25
    nodes
# Nodal List:       Freedom no
#    no, x,     y (mm),    X,   Y
 t m 1   0.     0.         r    r
 t m 2   4000.  0.         f    f
 t m 3   8000.  0.         f    r
 t m 4   4000.  3000.      f    f
    members
# Member List:
#    no, end1 node no, end2 node no, material no, area no.
 t   1   1             2                       1 1
 t   2   2             3                       1 1
 t   3   2             4                       1 1
 t   4   1             4                       1 1
 t   5   4             3                       1 1
# Please take note that this program does not tollerate
# blank lines in the input data.
   loads
# LoadCase 1:
# node no,      X load, Y load
 t     2        0       -60e3
 t     4        40e3       0
# LoadCase 2:
# tp  node no, X load, Y load
  t   2        0       -60e3
# LoadCase 3:
# tp  node no, X load, Y load
  t   4        40e3       0
   end
```

The above listed data is in a subdirectory dat and file truss1.dat. As lines with # are comments, one hopes that they do not need explanation, so we will discuss only the commands - lines without #.

The first command is *general* and signals that the section of data is, well

- general - applicable to either truss or frame. You see, the initial set of programs could solve either plane trusses or plane frames, but the fem.py tackles both, including structures that are part trusses, part frames.

The next command starts with flag *g* that signals that the line belongs to the *general* section. The numbers (1, 1, 4, 5, 3, 0) tell the program how many types of materials, sections, nodes, members, load cases and prescribed displacements there are in the structure.

In the examples very often there are only one type of material and probably one type of section, but the program allows for any number of types. The reason for using types is to ease data preparation for practical, real life structures. Other items are self explanatory, but we have not discussed so far the notion of *Prescribed Displacements*. The term prescribed displacements mainly applies to the reference points of the structure where it is necessary to specify non-zero displacement of that point, rather than the load at that point.

Next line *g 2 1 1* tells the program how many lines of data there will be in load case 1, 2 and 3 respectively.

The following line *b 0 0* tells the program that there are no frame (or beam) nodes and members. Then the line *t 4 5* states that the are 4 truss nodes and 5 truss members in the structure.

The line *g 1 200e3* says that the no 1 type of material has a Young's modulus of 200 000 kN/mm^2. How do we know it is in kN/mm^2 units? Because we are using kN to specify the loads and mm to specify the dimensions and all items must have the same units. This convention enables use of any units (viz ft and lb) and, more importantly, reduces the possibility of a mix up of units.

Cross-section properties list *g 1 1600.0 1.e-25* is the last line of data in the general section and tells the program that the type 1 section has an area of 1600 mm^2 and a very small moment of Inertia, I_x

The last data line completes the *general* section. Next line, *nodes* signals listing of nodes. In the first line, *t m 1 0 0 r r*, each item signifies: *t* tells the program that this is a *truss node m* that this is a *master* node, followed by the node number. Then follow x and y coordinates. The remaining two items indicate if this freedom is *restrained - r* or *free* to move as dictated by the loads. *Master node* signals that the freedoms need to be calculated and not copied from an already defined node. It is clear that node 1 is a support in which the displacement in x and in y direction is inhibited. Similarly, the third node is clearly a roller support, free to move in x direction, but restrained against the y direction.

Next comes the signal *members*, indicating that member list follows: *t 1 1 2 1 1* the first item *t* signals that this member is a *truss* member; *1* is the

member number and the memberts starts at node 1, ending at node 2. The material type is *1* and so is the section type.

We now have reached the section about *loads*. The program accepts loads at the nodes only, so the data in the first line *t 2 0 -60e3* means that a truss *t* node *2* has *0* horizontal load and *-60000 N* load, acting in the opposite direction to the y axis, i.e. acting downwards. Other lines carry similar information.

And finally we arrive at the *end* signal. What a relief!

Main Program Output.

The main part of program output follows. The program first outputs a copy of data, so that the possibility of finding *right answers* to the *wrong data* is greatly diminished - a born optimist would say eliminated. This part is omitted in the given output extract.

Next the program determines all nodal displacements, Then the program calculates from the displacements the member extension and thus the member forces. These forces are shown here. Since the truss is very simple indeed, these forces can be easily calculated by hand. As the hand calculation agrees with the program output, one can gain some confidence of the hand calculation, and/or perhaps the validity of the program output.

```
# truss1.dat - fem format.

Data echo is omitted for brevity in this output extract.
Please read truss1.out file for a sample of complete
output.  All xxx.out files are in "out" subdirectory.

Structure displacements (transposed matrix).
A matrix of dimensions (m x n), where m, n, LineLen =  3 5 5

  7.50000E-01  -2.86458E+00   1.50000E+00   1.23828E+00  -2.30208E+00

  5.00000E-01  -2.53125E+00   1.00000E+00   5.00000E-01  -1.96875E+00

  2.50000E-01  -3.33333E-01   5.00000E-01   7.38281E-01  -3.33333E-01


  Member stress resultants.
  Member  LC    Axial        Shear        B.M. 1       B.M. 2
    1     1  6.00000E+04
    1     2  4.00000E+04
    1     3  2.00000E+04

    2     1  6.00000E+04
```

```
2     2  4.00000E+04
2     3  2.00000E+04

3     1  6.00000E+04
3     2  6.00000E+04
3     3  0.00000E+00

4     1 -2.50000E+04
4     2 -5.00000E+04
4     3  2.50000E+04

5     1 -7.50000E+04
5     2 -5.00000E+04
5     3 -2.50000E+04
```

Please notice that the output is formatted in a simple way. As you can see, the output is for each member and all load cases. For each load case, shown are corresponding member force. This makes practical sense - when a member is designed, the designer wants to see stresses for every load case to which that member is subjected. It is desirable to keep all that information in one place.1

3.2 Checking Results of Analysis

It is tempting to "check" the results by comparison with another program. This is probably the simplest check. Whilst such check is undoubtedly useful, it is not necessarily easy nor is it necessarily adequate.

First, it implies that one has at hand another and independent computer program. But wait, how do we know that it is "independent". I am sure that this text has typographical errors. What if both programs followed the methods elucidated in this text, but without their authors bothering to understand, to check the formulae of this text and to correct the errors? If both programmers were competent in programming, they would produce programs that would give the same result, but the wrong result!

Secondly, in many situations one would afford only one program - they are costly. Of course, the Python program listed in this text is useful for the checking of other programs and is not costly, but the first limitation certainly still applies.

For these two reasons, particularly the first one, we consider truly independent checks.

3.2.1 Example

To see how to check the results, we will first analyse a more realistic example of a statically indeterminate truss, shown in the figure 3.5.

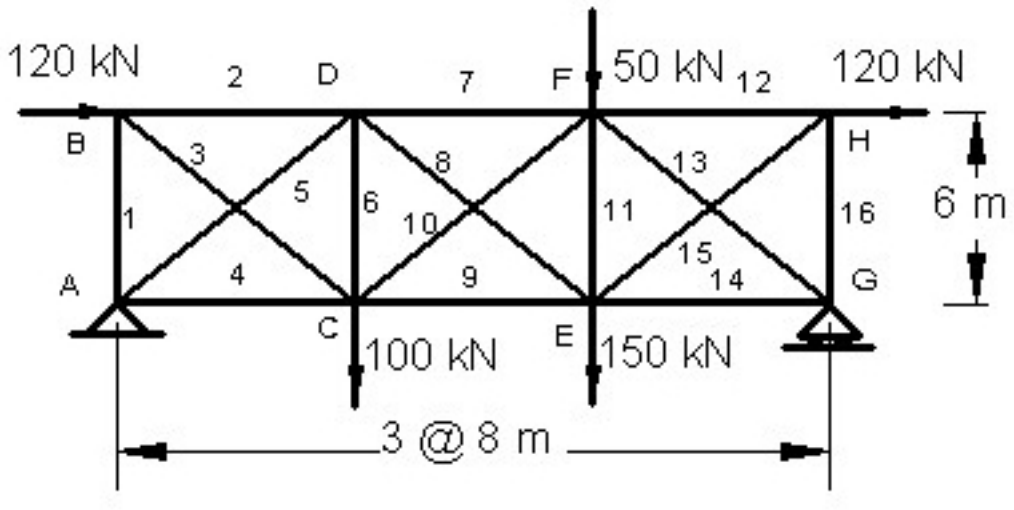

Figure 3.5: Truss with Redundant Members

We shall use kN and m units throughout. Any consistent units, such as pounds and feet, could be used. Complete data for this example is in file truss3.dat in the *dat* subdirectory. It is given that $E = 200 \cdot 10^6\ kN/m^2$ for all members. All uprights and bottom chord (members 1, 4, 6, 9, 11, 14, 16) have a sectional area of $2 \cdot 10^{-3}\ m^2$; the top chord (members 2, 7, 12) - area of $4 \cdot 10^{-3}\ m^2$; diagonals (bars 3, 5, 8, 10, 13, 15) - area of $1 \cdot 10^{-3}\ m^2$.

There are two load cases:

- LC 1: 100 kN load at C and 150 kN load at E
- LC 2: A horizontal 120 kN load at B and at H and a vertical load of 50 kN at F.

We take the X axis from right to left as positive, and the Y axis upwards as positive. This means that the downward loads are *negative* - a little counter intuitive, as we are more accustomed to downward loads than to upward loads.

Member axes $\bar{x}$ for uprights are taken from bottom to top, the chords - from left to right, the diagonals - so that dx is positive for all of them. With these conditions, the data for the truss is shown in the first part of the output table 3.1, as the program echos all data.

Freedoms, i.e. the numbers of force and corresponding displacement components are generated by the program, with zero number assigned to restraints (supports) where the displacements are inhibited.

Here are the results, excluding the echo of the data:

Table 3.1: Output of computer program

```
# truss3.out
Echo of data omitted - please refer to truss3.dat file
  Member stress resultants.
  Member  LC    Axial        Shear        B.M. 1       B.M. 2
    1     1 -6.03967E+01
    1     2  8.88919E+00

    2     1 -8.05289E+01
    2     2 -1.08148E+02

    3     1  1.00661E+02
    3     2 -1.48153E+01

    4     1  7.50266E+01
    4     2  1.94074E+02
```

```
 5     1 -9.37833E+01
 5     2  5.74069E+01

 6     1  3.68077E+01
 6     2 -2.11592E+01

 7     1 -1.81505E+02
 7     2 -4.45090E+01

 8     1  3.24371E+01
 8     2 -2.21416E+01

 9     1  1.51828E+02
 9     2  1.42158E+02

10     1  4.65934E+00
10     2  5.00807E+01

11     1  6.07051E+01
11     2 -3.67848E+01

12     1 -9.31101E+01
12     2  5.32403E+01

13     1 -1.05835E+02
13     2 -7.21059E+01

14     1  8.46677E+01
14     2  5.76847E+01

15     1  1.16388E+02
15     2  8.34496E+01

16     1 -6.98326E+01
16     2 -5.00698E+01
```

We are now ready to check these results for equilibrium of internal forces with the external forces at joints and for compatibility of deformation.

3.2.2 Equilibrium Checks

A simple and obvious check is to check the equilibrium of joints and the structure as a whole. If any condition of equilibrium is not satisfied then the solution is incorrect. However, only for statically determinate structures equilibrium checks are a sufficient condition to verify the correctness of the solution. Simply put, for trusses equilibrium is checked by working out the

sum of horizontal and vertical components of all member forces joints:

$$\Sigma X = \Sigma N \cdot \frac{dx}{L} + P_x \tag{3.21}$$

$$\Sigma Y = \Sigma N \cdot \frac{dy}{L} + P_y \tag{3.22}$$

ΣX = horizontal force at the joint;
P_x = x component of the applied force at the joint
ΣY = vertical force at the joint.
P_y = y component of the applied force at the joint

At the supports, the sum yields a reactions in the direction of support restraints. These checks are easily done by hand, though a simple computer program is useful. One simple way to implement this checking is to set up a spread sheet. It is pretty straight forward.

It would be a useful exercise for the reader to write this program as an exercise in Python.

For each joint we will input joint identification, then for each member meeting at that joint, the member number, its force N and the member projections onto the global axes, dx and dy. With this data the sums of X and Y components are calculated.

First, consider joint A and Load Case 2. The following listing shows the input data and output as well. To simplify tabulation, we take into account member forces in first place and then the applied forces, if any.

The sum of X components is the horizontal force on, acting from left to right pin A. Thus the horizontal reaction is 240 kN and acts from right to left. We see from the equilibrium of the structure as a whole that this is the correct result.

The sum of Y components indicates that the vertical reaction at A is downwards and is $43\frac{1}{3}$ kN. Taking moments about point G we can calculate the reaction as

$$\frac{2 \times 120 \times 6}{24} - \frac{50}{3} = 43\frac{1}{3} \quad kN$$

Not surprisingly, hand calculation of reactions agrees with the summation of member forces at that joint.

When the reactions can be calculated by statics alone, the structure is said to be *externally determinate*. The truss of figure 3.5 is externally determnate, but the number of bars and their arangement make it *internally indeterminate.*

Table 3.2: Truss Joint Equilibrium

```
Joint data, LC 2
Joint  No of members
  A  3
 1  8.88918977448   0  6
 4  194.074475255   8  0
 5  57.4069059314   8  6

 ------------------------
Program output:
Select fileName (<CR> selects 'dat.txt':  jointA-LC2.txt
Joint data, LC 2
Joint  No of members
['A', '3']
[1, '8.88918977448', '0', '6', 8.8891897744800001, 0.0, 6.0]
[4, '194.074475255', '8', '0', 194.07447525500001, 8.0, 0.0]
[5, '57.4069059314', '8', '6', 57.406905931399997, 8.0, 6.0]

SumN(X) =  240.0    SumN(Y) =  43.3333333333
```

We will now check the equilibrium of joint B under the Load Condition 1. Again, data for the problem is shown in the same listing as the program output.

Typically for most computer calculation the sums that are supposed to be zero are not quite zero, but a very small number. This occurs because the computer uses binary system of numbers and the conversion from decimal to binary very often introduces round off errors. Nothing to be allarmed about! In the same manner the equilibrium conditions of all pins can be verified.

3.2.3 Compatibility Checks

For statically indeterminate trusses, the equilibrium is a necessary but insufficient condition. One must also check the *compatibility.* Simply put, compatibility requires that after the deformation all the members joined at a common pin remain joined at that pin.

Also, the external support conditions *prescribe* the displacements at the supports. In a structure which is *indeterminate externally* the compatibility of displacements at those supports need be checked. In our present example, displacement compatibility at the supports is not really meaningful, as the support reactions for a given loading depend on the conditions of equilibrium alone.

Table 3.3: Joint B Equilibrium

```
Joint data, LC 1
Joint  No of members
  B  3
 1  -60.3966954268  0 -6
 2  -80.5289272358  8  0
 3  100.661159045   8 -6

 ------------------------
Program output:
Select fileName (<CR> selects 'dat.txt':  jointB-LC1.txt
Joint data, LC 1
Joint  No of members
['B', '3']
[1, '-60.3966954268', '0','-6', -60.396695426800001, 0.0,-6.0]
[2, '-80.5289272358', '8', '0', -80.528927235799998, 8.0, 0.0]
[3, '100.661159045', '8', '-6', 100.66115904500001, 8.0, -6.0]

SumN(X) =  2.00003569262e-10    SumN(Y) =  -2.00010674689e-10
```

We have introduced the work product as the sum of all forces with their corresponding displacements. We also equated the work product of the external forces P to the work product of the internal forces N, as the two system of forces are in equilibrium. (They better be in equilibrium for a structure that is firm and not going any place!) It is now convenient to formally state *The Principle of Virtual Work:*

For a given structure, if an equilibrium force field is considered with a compatible displacement field, then the Work Product, $\bar{W}$, of the external force and displacements is equal to the work product, $\bar{U}$, of the internal stress resultants and deformations.

Actually, the force field and the displacement field need not be related. As long as the force field satisfies the conditions of equilibrium and the displacement field satisfies the conditions of geometric compatibility, the principle of virtual work applies to the structure.

The principle of virtual work is fundamental in the modern structural analysis and we will return to it later. One form of the principle of virtual work is *The Principle of Virtual Forces:*

When the work products $\bar{W}$ and $\bar{U}$ are considered between a real displacement field and a virtual (imaginary) force field, the Principle of Virtual Work constitutes the Principle of Virtual Forces.

The principle of virtual forces can be used to calculate displacements

which result from known deformations, viz. bar extensions. The cause of the deformations is irrelevant for the applicability of the principle. We will use the principle of virtual forces to check the compatibility of deformations, calculated by the stiffness analysis of our truss, Figure (3.5).

It is convenient to remove enough bars to render the truss statically determinate. For given bar extensions that does not affect the joint displacements caused by the deformation of bars. We shall remove three diagonals, members 3, 8 and 13, but will not change the member numbering. That enables us to calculate quite simply the extensions along BC, DE and FG and compare the result with that calculated by the stiffness analysis.

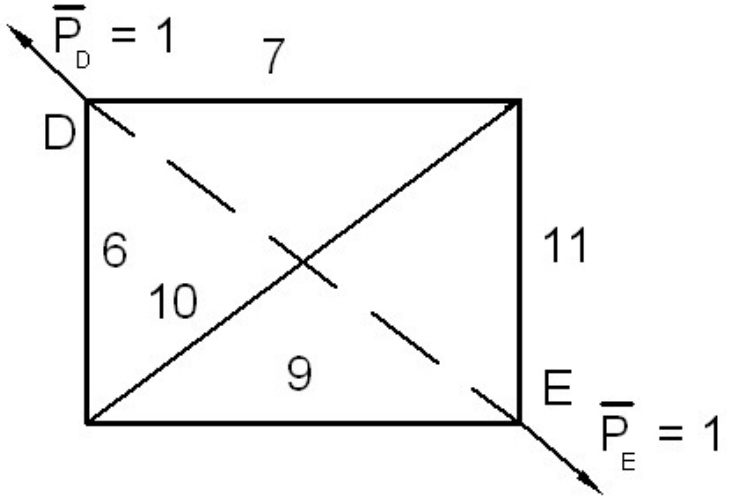

Figure 3.6: Truss Cell CEFD

As an example of this, let us consider the extension (or shortening) of bar DE under the displacements of the load condition 1. We apply *virtual forces* at D and E in the direction of bar DE, as shown in the figure 3.6. For convenience of calculation of $\bar{W}$ we chose virtual forces equal to 1 force unit. In this context we use the bar over a quantity to indicate that it is virtual, i.e. imaginary.

As the two virtual unit forces act in opposite directions and are of equal magnitude, in the force field there are no reactions and all the virtual forces in the bars are only in the cell CDFD. Thus, we can separate the cell from the rest of the structure and determine the virtual force field.

We will do the calculations by hand. Resolving the unit force at E into the horizontal and the vertical direction, we obtain $\bar{N}_9 = 0.8$ and $\bar{N}_{11} = 0.6$.

Similarly resolving the unit force at D, we obtain $\bar{N}_6 = 0.6$ and $\bar{N}_7 = 0.8$. Equilibrium conditions of point F yield $\bar{N}_{10} = -1.0$. Considerations of equilibrium at point C confirm that $\bar{N}_{10} = -1.0$. So we can now write the equation for the extension of bar 8:

$$1 \cdot e_8 = 0.6 \cdot e_6 + 0.8 \cdot e_7 + 0.8 \cdot e_9 - 1.0 \cdot e_{10} + 0.6 \cdot e_{11} \tag{3.23}$$

Substitution of the values for e, yields the value for the extension of bar 8 as $1.6219E-03$ This is in good agreement of $1.621856E-03$ given by the stiffness analysis.

As already mentioned, it is useful to do this calculation in a spread sheet, so that the results are easily verified. Frankly, I first did it by hand, but made a mistake. When one expects the results to be correct, when *checking* it is easy to make mistakes. Somewhere in our brain buried is the idea that checking is a waste of time, anyway... So it is a good idea to set out the calculations in a spread sheet.

In conlusion, if *all* the equilibrium conditions and *all* the compatibility conditions are satisfied, then the structural solution for the problem is assuredly correct.

Chapter 4

Plane Frames

4.1 Frame Members

Let us look at a typical member of a plane frame, which is lying the xy plane. It is convenient to draw the member horizontally - see Figure 4.1

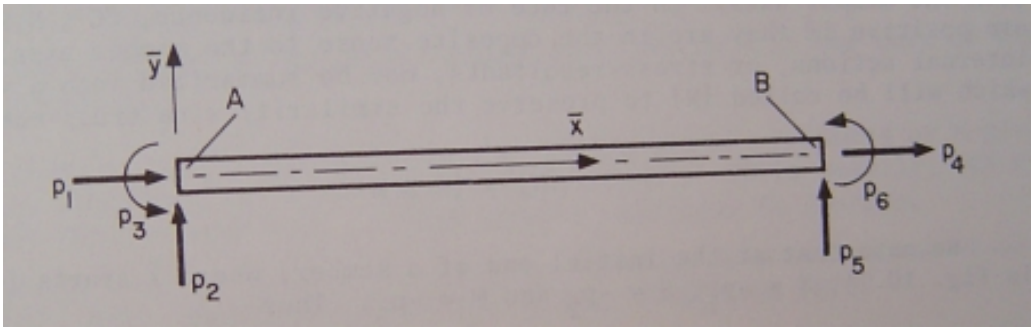

Figure 4.1: Plane Frame Member

For any member, the $\bar{x}$ axis is taken along the beam axis. The $\bar{y}$ axis is at right angle to the $\bar{x}$ axis. The $\bar{z}$ axis is at right angles to the $\bar{x}$ and $\bar{y}$ axes and in the direction of movement of a right-handed screw when turned in the direction of rotating the $\bar{x}$ towards the $\bar{y}$ axis.

With all these conditions there are still two choices for the $\bar{y}$ axis. For instance, in Figure 4.1, one could choose the $\bar{y}$ to point downwards instead of upwards. For consistency another condition is necessary - we choose the $\bar{y}$ so that the $\bar{z}$ always points out of the paper plane towards the viewer.

This then defines a consistent orthogonal system of member axes. The convention is the same for bars of a truss.

The joints at the end of a member may transmit an axial force, a transverse shear force and a couple onto the member. These forces are denoted by $\bar{P}_1 \ldots \bar{P}_6$. The forces $\bar{P}_1 \ldots \bar{P}_3$ act on the member at the end where $\bar{x}$ commences and the forces $\bar{P}_4 \ldots \bar{P}_6$ - on the end where $\bar{x}$ terminates. These are forces *external* to the member and are positive if they are in the same direction as the member axes.

The member ends also undergo displacements. The end displacements which correspond to $\bar{P}_1 \ldots \bar{P}_6$ are denoted by $\bar{u_1} \ldots \bar{u_6}$, each $\bar{u}$ taken in the same direction as the corresponding $\bar{P}$

The end forces give rise to axial force, shear force and bending moment. In absence of member forces, the axial and the shear forces are constant throughout the member, but the bending moment usually varies linearly along the member.

4.2 Decomposition into Elements

In order to determine the internal work product, i.e. the work product of the stress resultants, $\bar{U}$ the structure is decomposed into elements and $\bar{U}$ is calculated as the sum of the work products for the individual elements.

There is some choice as to the degree of decomposition. In chapter on trusses the elements were tacitly taken to be the bars of the truss: for any bar the work product $\bar{U}$ then was $N \cdot e$ where e was the overall bar extension and N was the bar force. In a frame with flexural members it is necessary to subdivide the members further into elements of infinitesimal length, dx, because the rotational deformations, in general, are not the same in all parts of the member.

Notice that the internal actions are always made up of *two* forces, acting in opposite direction of the element (ref. Figure 4.2)

In the element of Figure 4.2, the $\bar{x}$ axis is the outward normal to the face DD', which we refer to as *positive incidence.* Figure 4.2(b) shows the displacement components u and v, which are functions of $\bar{x}$. Figure 4.2(c) shows the internal actions in their positive directions. The axial force, N, the shear force, S, and the bending moment, M, produce actions on the face DD' which agree with the direction of member axes. (The couples in the xy plane are positive when they are in the direction when x axis is rotated towards the y axis). On the face of *negative incidence*, CC', N, S, and M are positive if they act in the opposite sense to the member axes.

These internal actions are the stress resultants. They may be summarised in a 3 by 1 matrix, or a vector, which will be called **N**.

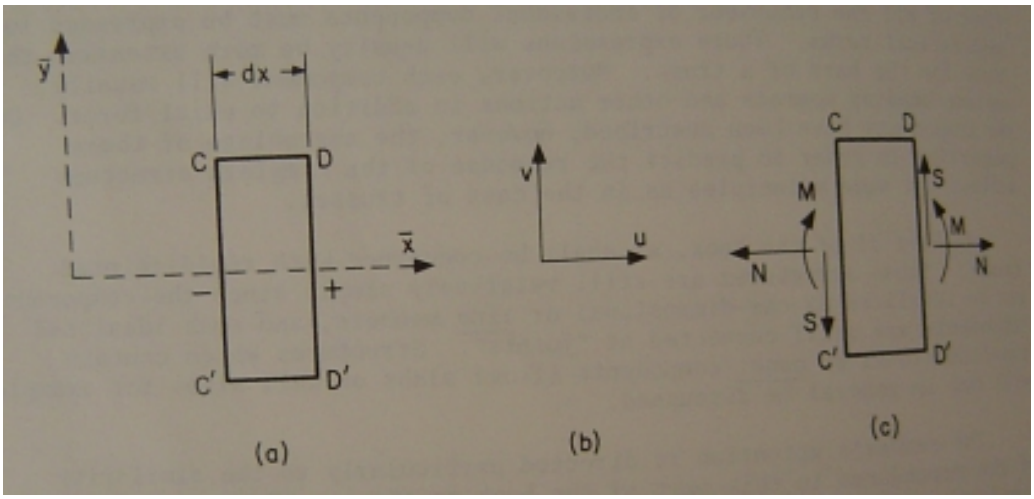

Figure 4.2: Elements of a Member

$$\mathbf{N} = \begin{bmatrix} N \\ S \\ M \end{bmatrix} = \{N, S, M\} \tag{4.1}$$

where the column matrix is shown twice - once with the usual square brackets, [, and,], and once with the curly brackets, {, and, }. The latter notation is less intrusive in the text. We will employ either notation for a column matrix. Calling the vector of internal actions $\mathbf{N}$ will stress the similarity of these equations to those of trusses.

Notice that at the starting end of the member, (where $\bar{x}$ starts, end A of Figure 4.1) $\quad N = -\bar{P}_1, \quad S = -\bar{P}_2, \quad M = -\bar{P}_3$, which we will write as

$$\bar{\mathbf{N}}_A = -\{\bar{P}_1 \quad \bar{P}_2 \quad \bar{P}_3\} = -\bar{\mathbf{P}}_A \tag{4.2}$$

At the other end, where the $\bar{x}$ has a positive incidence,
$N = +\bar{P}_4, \quad S = +\bar{P}_5, \quad M = +\bar{P}_6$, which we will write as

$$\bar{\mathbf{N}}_B = \{\bar{P}_4 \quad \bar{P}_5 \quad \bar{P}_6\} = \bar{\mathbf{P}}_B \tag{4.3}$$

Since the end forces $\bar{\mathbf{P}}_A$ and $\bar{\mathbf{P}}_B$ are the only forces on the member, they must be in equilibrium and we could write the equations for $\bar{\mathbf{P}}_A$ in terms $\bar{\mathbf{P}}_B$ and vice versa. However, our first aim is to describe member *deformations* in terms of the end forces. It is convenient to use *Virtual Forces* for the calculation of deformations.

4.3 Virtual Work

We will present the theorem of virtual work. It is applicable for elastic and non-elstic structures and can be used for the solution of very different problems, viz. determination of displacements caused by deformations due to temperature, inaccuracies in manufacture and so forth. In order to get to the "meat" of the problem, we will postpone the proof of the theorem to a later time and deal with it in the chapter on Generalisation of Structural Analysis. It has been already stated for trusses, but it is worth repeating it here.

> *For a given structure, if an equilibrium force field is considered with a compatible displacement field, then the Work Product, W, of the external forces and displacements is equal to the work product, U, of the internal stress resultants and deformations.*

It must be stressed that the stress resultants are part of the force field and thus are in equilibrium with the forces. To use the virtual force theorem, the force field must satisfy all equilibrium conditions, but not necessarily all compatibility conditions. So the theorem can be applied to internal actions that are calculated on the primary, statically determinate structure with real displacements. In that form, the theorem is known as *Method of Virtual Forces*.

For the application of the theorem, the displacement field and the force field may be independent of each other. On the other hand, the displacement field can be caused by the force field. All that we require of the displacement field is that it satisfies *all* the conditions of geometry.

4.3.1 Virtual Forces

When virtual work theorem is used with a virtual force field and real deformations, the resulting method is called *Virtual Force* method. 'Virtual' in this context is simply 'not real', 'imaginary'. It is a device of convenience. The virtual force method can be conveniently applied to the problem at hand: calculation of displacements. The method of virtual forces is used often enough to be worth formally stating it here:

> *For a given structure, if a virtual (imaginary) equilibrium force field is considered with a real, geometrically compatible displacement field, then the Work Product, $\bar{W}$, of the external virtual forces and real displacements is equal to the work product, $\bar{U}$, of the internal, virtual stress resultants and real deformations.*

To stress that the work products are calculated with virtual (imaginary) force field, we use a bar over U and W.

It is important to stress that the deformations are caused by whatever reason, *not* by virtual forces. Very often it is convenient to apply a virtual force of unit magnitude in units, consistent with the force and displacement units used in the calculation.

It is worth repeating that the deformations can be elastic, or plastic; they may be caused by temperature changes or any other cause. Even the displacements from the intended shape due to manufacturing inaccuracies can be calculated by the method of virtual forces. The only requirement in addition to geometrical consistency is that we can only deal with small displacements.

4.4 Member Flexibility in Member Axes

Consider the member of Figure (4.1), which is restrained at A. Let the loads be applied at the member ends only. At end B the force exerted by the joint B onto the member are denoted collectively as $\bar{\mathbf{P}}_B$. At end A the reaction to the loads at end B are denoted collectively as $\bar{\mathbf{P}}_A$ The two sets of forces, the applied forces at B and the reactions at A, are in equilibrium.

Member flexibility is a matrix which enables a quick determination of end displacements in terms of end force $\bar{\mathbf{P}}_B$:

$$\bar{\mathbf{u}}_B = \mathbf{f} \cdot \bar{\mathbf{P}}_B \tag{4.4}$$

where

$$\bar{\mathbf{u}}_B = \{\bar{u}_4, \bar{u}_5, \bar{u}_6\} \tag{4.5}$$

and

$$\bar{\mathbf{P}}_B = \{\bar{P}_4, \bar{P}_5.\bar{P}_6\} \tag{4.6}$$

After expanding equation 4.4 fully, the terms are as follows:

$$\begin{bmatrix} \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix} \cdot \begin{bmatrix} \bar{P}_4 \\ \bar{P}_5 \\ \bar{P}_6 \end{bmatrix} \tag{4.7}$$

In the above, the right hand side terms $\bar{P}_4$ $\bar{P}_5$ $\bar{P}_6$ are respectively, the axial force in $\bar{x}$ direction, the transverse force in $\bar{y}$ direction and a couple about the $\bar{z}$ axis. The left han side terms $\bar{u}_4$ $\bar{u}_5$ $\bar{u}_6$ are respectively the elongation

in the $\bar{x}$ direction, the transverse displacement in the $\bar{y}$ direction and the rotation about the $\bar{z}$ axis.

In order to calculate the terms of matrix $\mathbf{f}$, it is necessary to consider internal actions and deformations due to the end loads $\mathbf{P}_B$. The displacements, and therefore the terms of $\mathbf{f}$, are then determined by the method of virtual forces.

Since the member is loaded at its ends only, the axial force is constant throughout the member and is equal to $\bar{P}_4$. Similarly, the shear force is constant throughout the member and is equal to $\bar{P}_5$. The bending moment varies linearly with $\bar{x}$ and is equal to

$$M = (L - \bar{x}).\bar{P}_5 + \bar{P}_6 \tag{4.8}$$

Deformations due to shear are ignored in this derivation. For axial force, the flexibility term is the same as in a truss bar, so that

$$f_{11} = L/EA$$

As far as member flexibility and member stiffness is concerned, in member axes there is no interaction between the axial force and the bending moment. In other words, the axial force produces no bending and bending produces no elongation of the member. Of course, a bent member is shorter than a straight member, but with small deflections the shortening is negligibly small. Therefore

$$f_{12} = f_{13} = f_{21} = f_{31} = 0$$

.

The real bending moment divided by EI, M/EI, is equal to the actual curvature of the member. Therefore other terms of the flexibility matrix can be readily calculated by integrating over the member length L the product of the applied bending moment, divided by EI, with the virtual bending moment. (Equation 4.8 gives an expression for the bending moment in terms of end forces).

For instance, for $\bar{P}_6$ virtual unit couple, the integral becomes

$$\bar{u}_6 = \int_0^L 1 \cdot \frac{\bar{P}_5(L-x)}{EI} dx = \frac{L^2}{2EI}\bar{P}_5 = f_{32}\bar{P}_5 \tag{4.9}$$

The term f_{23} is similarly calculated when $\bar{P}_5$ is a virtual unit force and P_6/EI is the relative end rotation of an element of length dx:

$$\bar{u}_5 = \int_0^L 1 \cdot (L-x)\frac{\bar{P}_6}{EI} dx = \frac{L^2}{2EI}\bar{P}_6 = f_{23}\bar{P}_6 \tag{4.10}$$

Since $f_{32} = f_{23}$, the matrix f is symmetrical and therefore the matrix $\bar{k}$ will be symmetrical. As a consequence, the structure stiffness, $\mathbf{K}$ will also by symmetrical.

To determine f_{22}, we need to integrate M/EI caused by real $\bar{P}_5$, multiplied by the virtual bending moment caused by virtual unit force that corresponds to $\bar{P}_5$:

$$f_{22}\bar{P}_5 = \frac{\bar{P}_5}{EI}\int_0^L (L-x)^2 dx = \frac{L^3}{3EI}\cdot\bar{P}_5 \tag{4.11}$$

Similarly,

$$f_{33}\bar{P}_6 = \frac{\bar{P}_6}{EI}\int_0^L dx = \frac{L}{EI}\cdot\bar{P}_6 \tag{4.12}$$

We can summarise the *Member Flexibility* in member axes in a matrix equation as follows:

$$\mathbf{f} = \begin{bmatrix} \frac{L}{EA} & 0 & 0 \\ 0 & \frac{L^3}{3EI} & \frac{L^2}{2EI} \\ 0 & \frac{L^2}{2EI} & \frac{L}{EI} \end{bmatrix} \tag{4.13}$$

or, more compactly,

$$\mathbf{f} = \frac{L}{EI}\begin{bmatrix} I/A & 0 & 0 \\ 0 & L^2/3 & L/2 \\ 0 & L/2 & 1 \end{bmatrix} \tag{4.14}$$

Chapter 5

Stiffness of Plane Frames

5.1 End Stiffness in Member Axes

Inversion of the member end flexibility matrix (Equation 4.14) yields *Member End Stiffness* in member axes. The inversion is carried out algebraically and is very closely related to solution of linear simultaneous equations.

$$\bar{\mathbf{k}} = \frac{EI}{L}\begin{bmatrix} A/I & 0 & 0 \\ 0 & 12/L^2 & -6/L \\ 0 & -6/L & 4 \end{bmatrix} \tag{5.1}$$

Matrix $\mathbf{k}$ is the member end stiffness. If member starts at point A and ends at point B, and if the end A is constrained against all movement, then the displacements of B pre-multiplied by $\mathbf{k}$ are equal to the end forces at B.

Notice that both $\bar{\mathbf{k}}$ and $\mathbf{f}$ are symmetrical, hence $\bar{\mathbf{k}} \cdot \mathbf{f} = \mathbf{f} \cdot \bar{\mathbf{k}}$. If the inversion of $\mathbf{f}$ was correct, then the product is a unit matrix:

$$\mathbf{f} \cdot \bar{\mathbf{k}} = \mathbf{I} \tag{5.2}$$

where $\mathbf{I}$ is a (3x3) unit matrix. Actually, the above equation can be regarded as a basis for algebraic inversion of the $\mathbf{f}$ matrix: if we treat $\mathbf{I}$ as the right hand side of equations, the solution yields the values of $\bar{\mathbf{k}}$!

5.2 Member Stiffness in 'Global Axes'

Just as in trusses, we will need to transform the end stiffness to member stiffness in "global axes", but first we need to establish the equilibrium relationship between end forces at B to all end forces, acting on the member. It is a simple matter to calculate the end forces at A due to end forces at B when the member is in equilibrium:

$$\bar{P}_1 = -\bar{P}_4 \tag{5.3}$$
$$\bar{P}_2 = -\bar{P}_5 \tag{5.4}$$
$$\bar{P}_3 = -\bar{P}_5 \cdot L - \bar{P}_6 \tag{5.5}$$

It is convenient to write this in matrix form:

$$\begin{bmatrix} \bar{P}_1 \\ \bar{P}_2 \\ \bar{P}_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -L & -1 \end{bmatrix} \begin{bmatrix} \bar{P}_4 \\ \bar{P}_5 \\ \bar{P}_6 \end{bmatrix} \tag{5.6}$$

In matrix form the reaction at A due tho the forces at B are written as follows:

$$\bar{\mathbf{P}}_\mathbf{A} = \mathbf{A_1}.\bar{\mathbf{P}}_\mathbf{B} \tag{5.7}$$

where

$$\mathbf{A_1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -L & -1 \end{bmatrix} \tag{5.8}$$

When the member is inclined to the global axes by angle α, and therefore the $\bar{x}$ coordinate axes is also inclined to the global axes by α, one can express the horizontal and vertical components of the end forces. I have tacitly assumed that the global axis x is horizontal and y - vertical.

$$P_4 = \bar{P}_4 \cos\alpha - \bar{P}_5 \sin\alpha \tag{5.9}$$
$$P_5 = \bar{P}_4 \sin\alpha + \bar{P}_5 \cos\alpha \tag{5.10}$$

It is a little more systematic to express the above equation in terms of *direction cosines*. To that end let the angle between $\bar{x}$ and x be α. Furthermore, let the angle between $\bar{x}$ and y be β. To simplify writing of equations let $\cos\alpha = a.$ and $\cos\beta = b.$ Please notice that $\cos\beta = \sin\alpha$. If the projection of the member of lenght L on the x axis is d_x and its projection on the y axis is d_y, then

$$a = d_x/L \tag{5.11}$$
$$b = d_y/L \tag{5.12}$$

Then equilibrium of the joint at the free end can be expressed as

$$P_4 = a.\bar{P}_4 - b.\bar{P}_5 \tag{5.13}$$
$$P_5 = b.\bar{P}_4 + a.\bar{P}_5 \tag{5.14}$$

Similarly at the end 1, we have

$$P_1 = a.\bar{P}_1 - b.\bar{P}_2 \tag{5.15}$$
$$P_2 = b.\bar{P}_1 + a.\bar{P}_2 \tag{5.16}$$

The rotation about the z axis has no effect on the end moments. They remain the same in structure axes as they were in member axes. With that in mind, in matrix form the effect of the rotation of axes on the forces at A can be expressed as

$$\mathbf{P_A} = \mathbf{A_2}.\bar{\mathbf{P}}_\mathbf{A} \tag{5.17}$$

where the matrix $\mathbf{A}_2$ is

$$\mathbf{A}_2 = \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

We can now write a matrix equation for the the reaction at A in global axes due to forces applied at B in member axes. This involves first the translation of forces and then the rotation, using the above (3x3) transformation matrix $\mathbf{A}_2$:

$$\mathbf{P_A} = \mathbf{A_2}.\mathbf{A_1}.\bar{\mathbf{P}}_\mathbf{B} \tag{5.19}$$

The product of matrices $\mathbf{A_2}.\mathbf{A_1}$ is readily evaluated algebraically and is equal to the following matrix:

$$\mathbf{A_2}.\mathbf{A_1} = \begin{bmatrix} -a & b & 0 \\ -b & -a & 0 \\ 0 & -L & -1 \end{bmatrix} \tag{5.20}$$

The transformation of *all end forces* from member to global axes can now be written in matrix form as

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{P}}_\mathbf{B} \tag{5.21}$$

where the transformation matrix B, in detail, is equal to

$$\mathbf{B} = \begin{bmatrix} -a & b & 0 \\ -b & -a & 0 \\ 0 & -L & -1 \\ -- & -- & -- \\ a & -b & 0 \\ b & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.22}$$

Now it is possible to write the expressions for the member stiffness in terms of the end stiffness:

$$\bar{\mathbf{P}}_\mathbf{B} = \bar{\mathbf{k}}.\bar{\mathbf{u}}_\mathbf{B} \tag{5.23}$$

Member stiffness, in general is defined by the following equation

$$\mathbf{P} = \mathbf{k}.\mathbf{u} \tag{5.24}$$

By contragredience, the member stiffness matrix is defined as follows:

$$\mathbf{k} = \mathbf{B}.\bar{\mathbf{k}}.\mathbf{B}^\mathbf{T} \tag{5.25}$$

After assembly of member stiffnesses and the solution of equations, one can extract by location vectors member end displacement $\mathbf{u}$ and hence determine the *end B displacements, relative to the member axes with A fixed in position and direction*:

$$\bar{\mathbf{u}}_\mathbf{B} = \mathbf{B}^\mathbf{T}.\mathbf{u} \tag{5.26}$$

Stress resultants at B are then given by

$$\bar{\mathbf{P}}_\mathbf{B} = \bar{\mathbf{k}}.\bar{\mathbf{u}}_\mathbf{B} = \bar{\mathbf{k}}.\mathbf{B}^\mathbf{T}.\mathbf{u} \tag{5.27}$$

where

$$\bar{\mathbf{P}}_\mathbf{B} = \{\bar{P}_4 \;\; \bar{P}_5 \;\; \bar{P}_6\} \tag{5.28}$$

The axial force, N is constant throughout the member and so is the shear force, S. They are equal to:

$$N = \bar{P}_4 \tag{5.29}$$
$$S = \bar{P}_5 \tag{5.30}$$

The bending moment varies linearly between the member ends A and B and is equal at the two ends to the following:

$$M_B = \quad \bar{P}_6 \tag{5.31}$$
$$M_A = -\bar{P}_3 = L \cdot \bar{P}_5 + \bar{P}_6 \tag{5.32}$$

The equations, derived in this section, give adequate information to implement a computer program. The complete program and some sample data is in the tar ball of Python programs.

The following section presents some examples solved by this program.

5.3 Examples

5.3.1 Simple Portal Frame

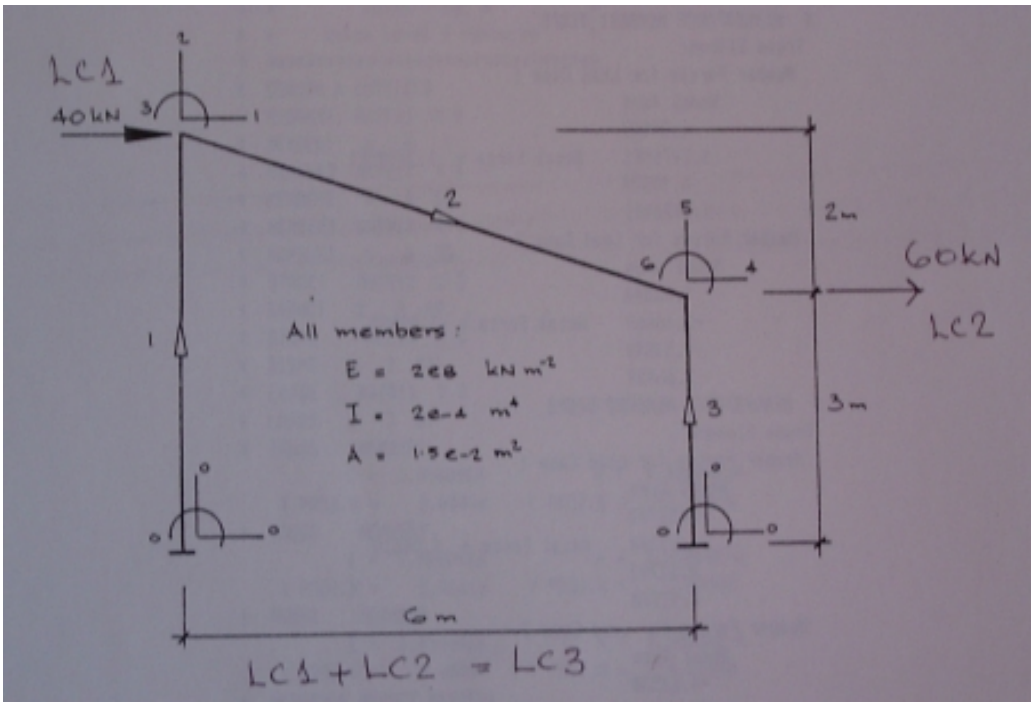

Figure 5.1: Frame 1

This is the input file for the frame in Figure 5.1. Three simple loading cases. Very straight forward. Note, however, that the main, starting input line has an additional zero parameter to signal that there are no prescribed non-zero displacements. In most situations this additional flag will have to

appear to signal 'ordinary' analysis of a frame, i.e. the response of the frame to loading on its joints.

The input file, frame1.dat, is listed in the adjacent listing. Three load cases are specified: LC1 - a horizontal load of 40 kN at the top of the first column; LC2 - a horizontal load of 60 kN at the top of second column; LC3 - both loads acting simultaneously.

Data

```
A small frame example
noMaterials,noSections,noNodes,noMembers,noLoadCases & Load nos
 1  1  4  3  3   0
 1  1  2
Materials: (kN m units)
no of material group,  Young's modulus, E only
 1  2e8
Section List (in this example one section only):
no, area, Ix  (m units)
1  1.5e-2  2e-4
Nodal List:
no, x, y (mm),  flagX, flagY, flagXYmount /dev/hda5 hda5
1   0.     0.         1  1  1
2   0      5.         0  0  0
3   6      3.         0  0  0
4   6      0          1  1  1
Member List:
no, end1 node no, end2 node no, material no, area no.
1  1 2                                    1 1
2  2 3                                    1 1
3  4 3                                    1 1
LoadCase 1:
 node no, X load, Y load   XY moment  (kN m units)
   2        40       0      0
LoadCase 2:
 node no, X load, Y load   XY moments
   3        60       0      0
LoadCase 3:
 node no, X load, Y load   XY
   2        40       0      0
   3        60       0      0
```

Output

The following listing shows the output of member forces. Of course, there is a rough and ready and totally inadequate check: member forces in load case three is the sum of member forces for load case 1 and load case 2.

```
   Member stress resultants:
 Member    LC       Axial        Shear       B.M. 1       B.M. 2
    1      1     18.8615      -9.7905     -27.2067      21.7458
    1      2      8.3645     -14.4810     -40.0189      32.3859
    1      3     27.2260     -24.2715     -67.2255      54.1318
    2      1    -34.6238       8.3405      21.7458     -31.0043
    2      2     11.0928      12.5145      32.3859     -46.7630
    2      3    -23.5310      20.8551      54.1318     -77.7673
    3      1    -18.8615     -30.2095     -59.6242      31.0043
    3      2     -8.3645     -45.5190     -89.7941      46.7630
    3      3    -27.2260     -75.7285    -149.4183      77.7673
```

5.3.2 Symmetrical Frame

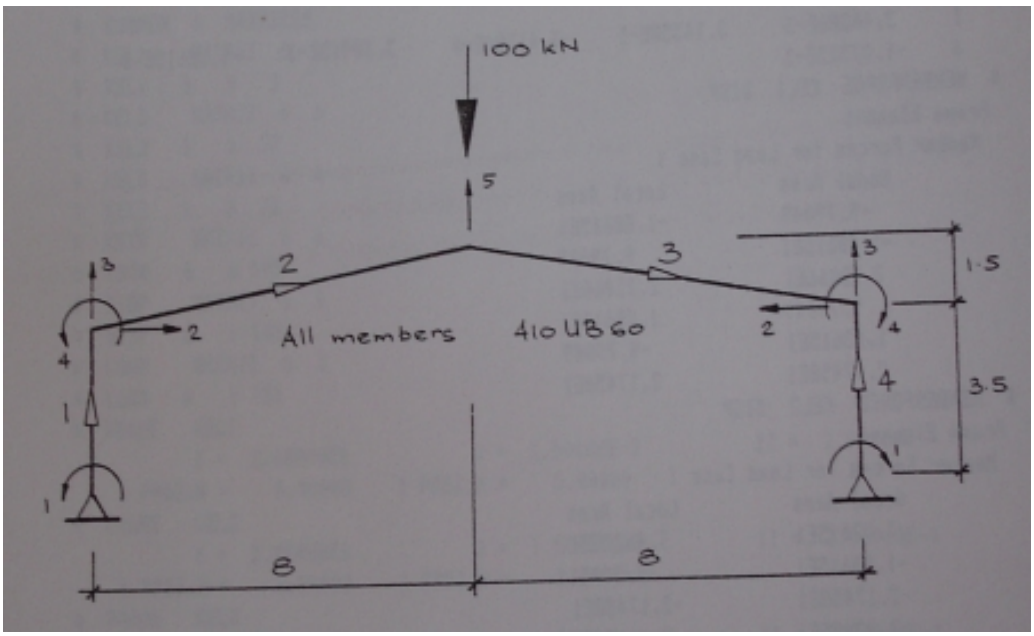

Figure 5.2: Frame 2

Figure 5.2 shows a symmetrical frame with a symmetrical load. It is a simple matter to take advantage of this symmetry. The joint on the line of symmetry will not undergo any rotation, nor any horizontal displacement. Furthermore, half the load will be carried on the left side of the frame, the other half - on the right side of the frame. Consequently, only half the members need be specified. To ensure that the symmetry conditions are satisfied, one can restrain the centre joint against rotation and against horizontal displacement, i.e. the flags flagXY and flagY are set to 1. The following listing shows the input to the program for the frame (frame2dat file). SMAP is a program used for comparison. A historical note on SMAP is the preamble of this book.

Data

```
SMAP example 5: Section 410USB60 - symmetry; watch node 3
noMaterials,noSections,noNodes,noMembers,noLoadCases &Load nos
 1  1  3  2  1   0
 1
Materials: (kN m units)
no of material group,  Young's modulus, E only
 1  2e8
Section List (in this example one section only):
no, area, Ix  (m units)
1  7600e-6  215e-6
Nodal List:
no, x, y (mm),  flagX, flagY, flagXY
1   0.     0.         1  1  0
2   0    3.5          0  0  0
3   8      5.         1  0  1
Member List:
no, end1 node no, end2 node no, material no, area no.
1  1 2                                    1 1
2  2 3                                    1 1
LoadCase 1:
 node no, X load, Y load   XY moment  (kN m units)
   3        0       -50      0
```

Output

Next listing shows a part of the program output. Member forces, or stress resultants, are listed in a table. The units are the same as those of input -

forces are in kN and moments are in kN m units.

```
Calculate member forces ------------------------->
   Member stress resultants:
 Member    LC       Axial        Shear       B.M. 1       B.M. 2
    1      1     -50.0000      44.9127       0.0000    -157.1943
    2      1     -53.3578     -40.8667    -157.1943     175.4367
```

5.3.3 Beam with Prescribed Support Settlement

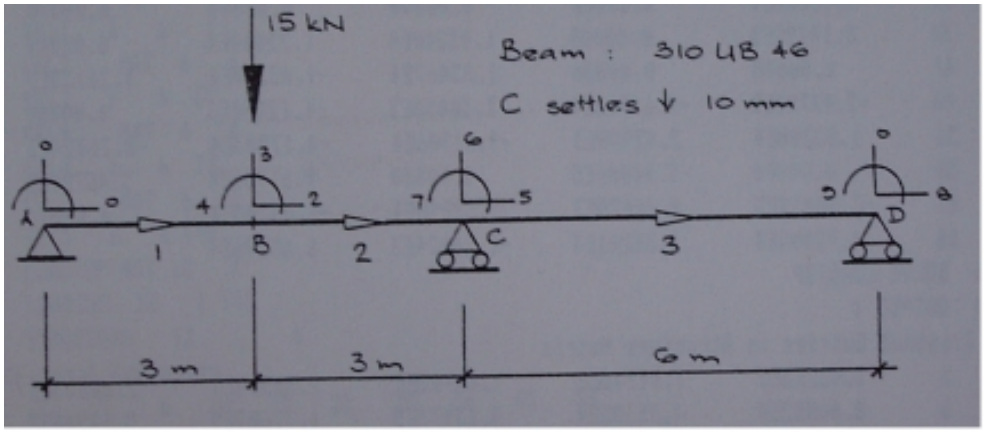

Figure 5.3: Beam

The example of the beam shows a new feature - prescribed non-zero displacements. Next listing is the data for beam3 (shown in Figure 5.3). Two loading cases are considered, both carry the same applied load, as shown in the figure (5.3). In the first loading case (LC1) taken into account is a settlement of support C of 10 mm. This is the same as the results of beam2.dat. The second loading case (LC2) there is no settlement of support C. This is the same as beam1.dat with a technical difference. In the current example a very small (negligibly small) settlement is specified (1e-23 mm), whilst in beam1.dat it is simply specified as 0 by assigning it a flagY of 1, which signals that the vertical displacement is zero.

Just in case it is confusing - all three data sets (beam1.dat, beam2.dat and beam3.dat) refer to the same structure, shown in figure 5.3. In beam1 zero displacement of support C is specified. In beam2, there is a settlement

of C of 10 mm. In beam 3, both situations (beam1 and beam2) are investigated. Please notice that in beam3, once a non-zero displacement is explicitly specified in LC1, it must be explicitly specified in all loading conditions.

The reason for this requirement is that in order to solve a structure for non-zero specified displacements, we modify the stiffness equation and the load vector. Instead of the load that corresponds to the specified displacement, the load matrix contains the displacement, or a displacement multiplied by a "large number". The corresponding row of the stiffness matrix, is either filled with zeros with 1.000 on the diagonal. Alternatively, the diagonal term in that row is replaced by the "large number" and other terms remain untouched, as assembled normally.

Both methods lead the same result, provided that the "large number" is suitably chosen. The advantage of the "large number" technique is that the structure stiffness matrix remains symmetrical. Whilst we have not taken account of symmetry, in large problems most storage and solution schemes do take advantage of it.

There is another small advantage of the "large number" technique - it is relatively easy to implement it in programs. The disadvantage is that we have to rely on numerical "swamping" of smaller numbers with a large number. If one tried hard enoutogh, it may well be possible to find examples where the large number technique is incapable of yielding an accurate solution. Of course, that is true in all numerical solutions. I say it again - a functioning program is not an adequate substitute for understanding of the problem and knowledge of the method of solution.

It the following, I will list the data and the most results for problem 3. Then for comparison we will look at some of the input and output for beam1 and beam2.

Data

```
This is the input of beam3.dat
noMaterials,noSections,noNodes,noMembers,noLoadCases,noPrescribes:
 1  1  4  3  2  1
 1  1
Materials: (kN m units)
no of material group,  Young's modulus, E only
 1  2e8
Section List (in this example one section only):
no, area, Ix  (m units)
1  5890e-6  99.5e-6
Nodal List:
no, x, y (mm),  flagX, flagY, flagXY
```

```
1   0.     0.          1  1  0
2   3      0           0  0  0
3   6      0           0  0  0
4  12      0           0  1  0
Member List:
no, end1 node no, end2 node no, material no, area no.
1  1 2                                      1 1
2  2 3                                      1 1
3  3 4                                      1 1
LoadCase 1:
 node no, X load, Y load   XY moment  (kN m units)
   2        0       -15      0
LoadCase 2:
The same as LC1, but different displacements
   2        0       -15      0
Prescribed displacements LC1
node no with the prescribed displacement for each load case
   3        0     -10e-3    0
LC2
Prescribed displacements - small number, not zero!
   3        0     1e-25     0
```

Output of stiffness matrix

This is the output in several parts. First, the stiffness matrix.

```
Diagnostic:
[[  2.65333333e+04   0.00000000e+00  -1.32666667e+04   1.32666667e+04
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00]
 [  0.00000000e+00   7.85333333e+05   0.00000000e+00   0.00000000e+00
   -3.92666667e+05   0.00000000e+00   0.00000000e+00   0.00000000e+00
  0.00000000e+00]
 [ -1.32666667e+04   0.00000000e+00   1.76888889e+04  -1.81898940e-12
    0.00000000e+00  -8.84444444e+03   1.32666667e+04   0.00000000e+00
    0.00000000e+00]
 [  1.32666667e+04   0.00000000e+00  -1.81898940e-12   5.30666667e+04
    0.00000000e+00  -1.32666667e+04   1.32666667e+04   0.00000000e+00
    0.00000000e+00]
 [  0.00000000e+00  -3.92666667e+05   0.00000000e+00   0.00000000e+00
    5.89000000e+05   0.00000000e+00   0.00000000e+00  -1.96333333e+05
    0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00  -8.84444444e+03  -1.32666667e+04
    0.00000000e+00   7.85333333e+25  -9.95000000e+03   0.00000000e+00
    3.31666667e+03]
 [  0.00000000e+00   0.00000000e+00   1.32666667e+04   1.32666667e+04
    0.00000000e+00  -9.95000000e+03   3.98000000e+04   0.00000000e+00
    6.63333333e+03]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   -1.96333333e+05   0.00000000e+00   0.00000000e+00   1.96333333e+05
    0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   3.31666667e+03   6.63333333e+03   0.00000000e+00
    1.32666667e+04]
```

Notice in the listing of stiffness that the term (6,6) is 7.85333333e+25, which is large, compared with other diagonal terms of the structure stiffness matrix. Evidently, the program uses 1e20 as a large multiplier and

the large diagonal terms of the structure stiffness matrix before modification was 7.85333333e+05. The product of the large multiplier and the greatest diagonal term is the large number, 7.85333333e+25, used in the solution.

The next table lists the structure load matrix, Ps, and the structure displacement matrix, us. Notice that the 6-th row of the load matrix is the prescribed (specified) displacement, multiplied by the large number. That insures that after the solution the corresponding term of us is the specified displacement.

Output of load matrix

```
Ps =
[[  0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00]
 [ -1.50000000e+01  -1.50000000e+01]
 [  0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00]
 [ -7.85333333e+23   7.85333333e+00]
 [  0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00]]
Nodal displacements for all load cases, us =
[[ -3.77198492e-03  -1.27198492e-03]
 [  0.00000000e+00   0.00000000e+00]
 [ -9.31297111e-03  -2.43797111e-03]
 [ -1.76900126e-03   1.05998744e-04]
 [  0.00000000e+00   0.00000000e+00]
 [ -1.00000000e-02  -3.13136672e-26]
 [  8.47989950e-04   8.47989950e-04]
 [  0.00000000e+00   0.00000000e+00]
 [  2.07600503e-03  -4.23994975e-04]]
```

Output of stress resultants

Member stress resultants:

Member	LC	Axial	Shear	B.M. 1	B.M. 2
1	1	0.0000	-8.8576	-0.0000	26.5729
1	2	0.0000	-6.0937	0.0000	18.2812
2	1	0.0000	6.1424	26.5729	8.1458
2	2	0.0000	8.9062	18.2812	-8.4375
3	1	0.0000	1.3576	8.1458	0.0000
3	2	0.0000	-1.4062	-8.4375	0.0000

As a partial, and again inadequate check, a listing of output for data of beam1.dat and beam2.dat is shown in the following listings. It is compared with LC1 and LC3 of output for beam3.dat. First, response to beam1.dat.

Output in response to beam1.dat

Member stress resultants:

Member	LC	Axial	Shear	B.M. 1	B.M. 2
1	1	0.0000	-6.0937	0.0000	18.2812
2	1	0.0000	8.9062	18.2812	-8.4375
3	1	0.0000	-1.4062	-8.4375	0.0000

Next, the program response to data of beam2.dat.

Output in response to beam2.dat

Member stress resultants:

Member	LC	Axial	Shear	B.M. 1	B.M. 2
1	1	0.0000	-8.8576	-0.0000	26.5729
2	1	0.0000	6.1424	26.5729	8.1458
3	1	0.0000	1.3576	8.1458	0.0000

Additional material on Plane Frames

Statically Determinate Frames
Statically Indeterminate Frames
Flexibility Solution - One degree of indeterminacy
Virtual Displacements
Master and Slave Nodes
Checking of Results
External Forces on Members

Chapter 6

Space Structures

6.1 Stiffness Solution of 3D Trusses

6.1.1 Digression

I wish to digress here. Here, because I have not yet found a good place where to put this material. The development of the program in the overall picture of the work has been slow and sporadic. At times, all else, all the worries were pushed aside to do the work of programming and then there were long periods of inactivity in this area. Whilst my hobby of personal computers is mainly responsible for the periods of inactivity of programming, often times it is the other aspects of life that caused interruptions. Not the least of which has been the health of those near and dear to me, particularly the serious illness of my wife of 60 years, who has been my supporter and my love throughout our long and varied life.

All that not withstanding, the program has been completely rewritten recently. It is now basically in the form where it can be readily developed into a finite element program without too much effort. It is now able to be applied to structures made up of different types of elements. Currently it implements only two types of elements - truss bars and frame beams. In principle, it is easy to extend it by adding additional types of elements. I am very tempted to add a space truss element and test that with a few problems.

Actually, the new program is significantly shorter than the original - about 40% shorter. It is now only a little over 1,500 lines long! It is more comprehensive and shorter, so how did this come about? Basically, by the extensive use of Python's dictionaries, which proved to be such a boon to the development.

As I have stressed repeatedly, to really grasp the subject one needs to study the programs as much or even more than the text. Dictionaries is a

relatively new concept in programming. So to read and understand the new version of the program one needs to be familiar with *Python Dictionaries.*

6.1.2 Member Stiffness

Stiffness solution of space trusses mirrors closely the solution of plane trusses. The solution follows exactly the same steps. First, we determine stiffnesses of the individual members. These member stiffnesses are then assembled into a structure stiffness. The loads of each load case are assembled into a load vector. The relationship between the load vector, the structure stiffness and the structure displacements yields the *Structure Stiffness Equations* analogously to a plane truss.. The solution of these equations yields the values of all displacements.

Once the displacements are determined from the Structure Stiffness Equations, the displacements of each member are extracted and the forces in individual members determined.

6.1.3 Member axes

We analyse member properties in *member axes*, which pass through the centroids of the members and along the axes of members. Unfortunately, we can not yet effectively draw 3 dimensional members that visually look 3 dimensional. So we shall refer user to the figure 3.1 and describe the third dimension in words.

The Figure shows a member with *member axes* $\bar{x}$ and at right angle $\bar{y}$. Member axis $\bar{x}$ measures the distance on the member from its start point. The 3rd member axis $\bar{z}$ is always assumed to be at right angle to the plane of $\bar{x}$ and $\bar{y}$ The $\bar{y}$ axis is always at the right angle to the $\bar{x}$ and $\bar{z}$ are also at right angles. The axes follow the right hand rule: when x is rotated towards y a right hand screw would move in the direction of the z axis; if y axis is rotated to z axis, a right hand screw thus rotated, would move in the direction of the x axis; if z axis is moved towards x, a right handed screw would move to the y direction.

The forces, applied on the member, $\bar{P}_1$, $\bar{P}_2$ and $\bar{P}_3$ are positive when they act in the same direction as the $\bar{x}$, $\bar{y}$ and $\bar{z}$ axes. The tension in the bar, N consists of two forces, which are in equilibrium and therefore act in opposite directions.

The orientation of the bar is defined by its projections on the structure x, y and z axes and are denoted by dx, dy and dz respectively. The length of the bar is then given by $\sqrt{(dx)^2 + (dy)^2 + (dz)^2}$.

The ends of a member are always assumed to be pinned to the truss as a whole. Clearly the force at each end must pass through the pin, just as in a two dimensional truss. Again, as in plane trusses, the sign convention is that tension in the member is positive. Therefore a positive member force N at the end 2 of the member acts in the direction of $\bar{x}$ whilst at the end 1 - in the direction opposite of $\bar{x}$.

Again, exactly as in the plane truss, consider a member held in position by the pin at end 1. The stresses in the member are uniform of magnitude $\sigma = \frac{N}{A}$, where A is the area of the member. The member extension, e is simply

$$e = \frac{L}{EA} \cdot N \tag{6.1}$$

where E is the *Modulus of Elasticity (Young's Modulus).* L is the member length. The term $\frac{L}{EA}$ is the *flexibility* of the member. The inverse of member flexibility is the end stiffness of the member, $\bar{k}$. In terms of member end stiffness, the *force-displacement* relation becomes

$$N = \frac{EA}{L} \cdot e \tag{6.2}$$

where $\dfrac{EA}{L}$ is the *member end stiffness in member axes*, $\bar{k}$. Generally, $\bar{k}$ is a matrix, but even for a pin jointed member space truss member it is still a scalar, or a (1 x 1) matrix. We can write a formal expression for $\bar{k}$ as follows:

$$\bar{k} = \frac{EA}{L} \tag{6.3}$$

6.1.4 "Global" Member Axes

In order to ensure the equilibrium of each joint, it will be necessary to augment forces of all members meeting at any one joint. For this reason we will need to work in axes for each member that are in the direction of *structure axes.* The global member axes numbering for plane truss member was shown in figure 3.2. In a space truss, there are a total of 6 truss member end forces. They are numbered 1,2,3 at and 1. At end 2, the end forces are numbered locally 4,5 and 6. As one would expect, the forces 1 and 4 are parallel to and in the direction of the $\bar{x}$ axis, forces 2 and 5 - $\bar{y}$ and 3 and 6 - $\bar{z}$.

A vector at the end 2 of a member can be resolved in the direction of $\bar{x}, \bar{y}\, and\, \bar{z}$ axes as a *scalar product* of the vector with the respective axis. If the vector is of unit length (i.e. it is a *unit vector*), its three components can be conveniently written as a column matrix:

$$\begin{aligned} cos\alpha &= \frac{dx}{L} \\ cos\beta &= \frac{dy}{L} \\ cos\gamma &= \frac{dz}{L} \end{aligned} \tag{6.4}$$

Furthermore,

$$\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} = (N) \begin{bmatrix} -dx/L \\ -dy/L \\ -dz/L \\ dx/L \\ dy/L \\ dz/L \end{bmatrix} \tag{6.5}$$

And the (6x1) Transformation Matrix B is as follows:

$$\mathbf{B} = \begin{bmatrix} -dx/L \\ -dy/L \\ -dz/L \\ dx/L \\ dy/L \\ dz/L \end{bmatrix} \tag{6.6}$$

In terms of this Transformation Matrix, member end forces in global axes are given, in terms of member forces in local axes N, in exactly the same way as in planar trusses as follows:

$$\mathbf{P} = (N) \cdot \mathbf{B} \tag{6.7}$$

For each force component there is a corresponding displacement component. The displacement components are so chosen that the work product is equal to the sum of every force P multiplied by the corresponding displacement u. In matrix notation that can be written as

$$\mathbf{P}^T \cdot \mathbf{u}$$

We now look at the end displacements. In the same manner as for end forces, consider first the displacements of end 2 in global x. y and z directions. If we look at the geometry of the bar end displacements, bearing in mind that they are small and we can approximate the direction of displaced bar as the same as the direction of the bar before the displacement, by similar triangles we find that

$$e = (dx/L).u_4 + (dy/L).u_5 + (dz/L).u_6$$

To get the bar extension due to displacements of both end nodes, we need to add the effect of the displacements at end 1 to obtain:

$$\begin{gathered} e = \\ -(dx/L).u_1 - (dy/L).u_2 - (dz/L).u_3 \\ +(dx/L).u_4 + (dy/L).u_5 + (dz/L).u_6 \end{gathered}$$

If we look at the above expression, we see that the terms of matrix **B** appear in it. They do appear in the same order, but in a row, rather than a column. The displacement componets $u_1, u_2, u_3, u_4, u_5, u_6$ written in a column are a (6x1) matrix **u**. In matrix terms the above expression becomes:

$$e = \mathbf{B^T}.\mathbf{u} \tag{6.8}$$

Comparing the equation for the force N (equation 6.7) with the equation for the elongation e (equation 6.8), we see a remarkable similarity of "Contragredience".

It is useful to record these two equations side by side:

$$\mathbf{P} = \mathbf{B} \cdot (N)$$

$$e = \mathbf{B^T}.\mathbf{u}$$

Notice that **P** corresponds to **u** as N corresponds to e. The second equation looks a kind of transposition of the first. That makes these equations easy to remember. We can not fail to notice that in terms of matrices, the expressions are identical to those of planar trusses - the difference is in the size of matrices.

It is tempting to leave the rest for the reader to obtain by reference to planar trusses. However, there is no harm to summarise the expressions here, much the same as for the planar trusses. It will serve as a convenient reference for writing a program for space trusses.

We now write the expression for member stiffness in global axes:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot e$$

.

Substituting the second contragredience equation into the above, we have:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot \mathbf{B^T}.\mathbf{u}. \tag{6.9}$$

Member stiffness in global axes, **k** defines the relationship between member forces in global axes and member displacements in global axes, as follows:

$$\mathbf{P} = \mathbf{k} \cdot \mathbf{u}. \tag{6.10}$$

So the member stiffness in global axes $\mathbf{k}$ in terms of the transformation matrix $\mathbf{B}$ and the member stiffness in member axes, $\bar{\mathbf{k}}$ is:

$$\mathbf{k} = \mathbf{B} \cdot \bar{\mathbf{k}} \cdot \mathbf{B^T}, \tag{6.11}$$

where the member stiffness in member axes is actually a scalar and is equal to $\bar{\mathbf{k}} = \frac{EA}{L}$ It is clear that the member stiffness $\mathbf{k}$ is symmetrical, as its transpose is equal to itself.

6.1.5 Displacements and Forces

Truss Displacements

The structure stiffness matrix is assembled in the same manner as for planar trusses and frames. Though the assembled struture stiffness matrix is often very sparse matrix, we simply use a precompiled module 'numpy', 'Numeric' or "numarray', written mainly in C.

A good feature to check is to ascertain that the stiffness matrix is *positive definite* and also symmetrical, so that stiffness matrix elements are symmetrically placed along the leading diagonal, thus $K_{ij} = K_{ji}$

Member Displacements

Member displacements are extracted by the same *location vectors* into the member end displacements in global axes. For any member, the i-th term of member end displacements is equal to:

$$u[i] = u_s[loc[i]] \tag{6.12}$$

Actually, in the program that has been renamed fem.py, which accepts negative location vector values, the extraction algorithm is somewhat modified. The negative value of a term in the location vector signifies by convention that the value of the displacement is in a changed convention where the negative signifies that the force and displacement positive direction is in the opposite of the usual convention. The need for this peculiarity arrises when symmetry and anti-symmetry is taken into account as a short cut. The following equation for the extraction of element displacements takes this feature into account. Similar, though somewhat more complex modification is also applied in the assembly process. The reader is urged to look at examples, as description may not be clear enough.

$$u[i] = sign(loc[i]) \times u_s[abs(loc[i])] \tag{6.13}$$

In the above equation, $sign(x)$ returns $+1$ if x is positive and -1 if x is negative. The functions $abx(x)$ returns the positive value of x, regardless of the sign. Please notice that in Python, function $sign(x)$ has a totally different meaning and in the program we provide our own function that is named $sgn(x)$ to do the job (sgn so it is not a Python keyword $sign$).

These scalar terms (u[i]) are augmented into the member displacement vector in global axes $\mathbf{u_s}$.In order to calculate member forces, we need to change those displacements into the local member axes. We do this using the contragredience equations, 3.7:

$$e = \mathbf{B^T} \cdot \mathbf{u} \tag{6.14}$$

Evidently, the above matrix expression is again identical to the matrix expression in planar trusses. The only difference is in the size of the matrices. The same is true also in the remainder of this section on Displacements and Forces.

Member Forces

Once member end displacements in member axes are known, we know member extension (or shortening) and hence the member force:

$$N = \frac{EA}{L} \cdot e \tag{6.15}$$

6.1.6 Stress Matrix

Stress Matrix is simply an alternative expression for internal forces in terms of element displacements. It is defined as matrix which post-multiplied by the element displacements yields stress resultants. In case of truss member, there is only one stress resultant, namely N. Substitution of equation 6.1.5 into the above equation 6.1.5 yields the required expression:

$$N = \frac{EA}{L} \cdot \mathbf{B^T} \cdot \mathbf{u} \tag{6.16}$$

So we can formally write the expression for the *Stress Matrix* as:

$$\mathbf{StressMatrix} = \frac{EA}{L} \cdot \mathbf{B^T} \tag{6.17}$$

Summary

It may be useful to summarise the whole process again as for planar trusses:

- $\mathbf{k} = \mathbf{B} \cdot (\frac{EA}{L})\mathbf{B}^T$
- Assemble all $\mathbf{k}$ to structure stiffness $\mathbf{K}_s$
- Assemble load vector, $\mathbf{P_s}$
- Solve stiffness equations, $\mathbf{P_s} = \mathbf{K}_s \cdot \mathbf{u_s}$
- Extract member displacements $\mathbf{u}$ from $\mathbf{u_s}$
- Find member forces, $N = \frac{EA}{L} \cdot \mathbf{B^T} \cdot \mathbf{u}$

Chapter 7

Generalisation

7.1 Contragredience

This is a name for the remarkable relationship of equilibrium and geometry. A set of forces $\mathbf{P}$ can be transformed into another set of statically equivalent forces, $\bar{\mathbf{P}}$ with a matrix $\mathbf{B}$ as follows:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{P}} \tag{7.1}$$

$\mathbf{P}$ and $\bar{\mathbf{P}}$ are vectors [1] of force components, which we indicate by bold text. $\mathbf{B}$ is a matrix, which we also show by bold text.

For practical purposes, $\bar{\mathbf{P}}$ are often forces in member coordinate axes and $\mathbf{P}$ in axes parallel to global axes, which for brevity we refer to as global axes.

Let $\bar{\mathbf{u}}$ and $\mathbf{u}$ be displacements that correspond to $\bar{\mathbf{P}}$ and $\mathbf{P}$ respectively. Since $\bar{\mathbf{P}}$ and $\mathbf{P}$ are statically equivalent (or indeed the same forces in different axes), the work product of $\bar{\mathbf{P}}$ and $\mathbf{P}$ during any displacement $\bar{\mathbf{u}}$ and $\mathbf{u}$ are the same. Notice that we have not invoked any stress strain relationship - only equilibrium. In matrix notation the work product can be written as

$\bar{\mathbf{k}}$

$$\mathbf{P}^T \cdot \mathbf{u} = \bar{\mathbf{P}}^T \cdot \bar{\mathbf{u}} \tag{7.2}$$

Transposition is indicated by the superscript T. Substitution of the first equation into the second yields:

$$\bar{\mathbf{P}}^T \cdot \mathbf{B}^T \cdot \mathbf{u} = \bar{\mathbf{P}}^T \cdot \bar{\mathbf{u}} \tag{7.3}$$

[1] By vector here we mean a matrix of n rows and one column. It is convenient to refer to it as *n dimensional vector*.

Since the equation applies to any $\bar{\mathbf{P}}$ and therefore to any $\bar{\mathbf{P}}^T$, $\bar{\mathbf{P}}^T$ can be cancelled on both sides of the equals sign, yielding a relationship of geometry as follows:

$$\bar{\mathbf{u}} = \mathbf{B}^T \cdot \mathbf{u} \tag{7.4}$$

It is worth stressing that the matrix B is derived from considerations of equilibrium and nothing else but equilibrium. The resulting above equation is a relationship of geometry of displacements $\mathbf{u}$ and $\bar{\mathbf{u}}$ that correspond to forces $\mathbf{P}$ and $\bar{\mathbf{P}}$ respectively. This relationship is often called *contragredience.* It is easier to remember the equations if they are written side by side:

$$\mathbf{P} = \mathbf{B} \cdot \bar{\mathbf{P}}$$

$$\bar{\mathbf{u}} = \mathbf{B}^T \cdot \mathbf{u} \tag{7.5}$$

Contragredience plays an important role in simplifying the derivations of stiffness equations for the analysis of structures. Furthermore, it can be used to solve problems of geometry by solving the corresponding equilibrium equations.

Generally we are dealing with small displacements only. If the displacements were large, one would have to take into account the change of angles between the member axes, which as a rule are taken to be along the member axis, and the global axes, which do not change during displacement. To put it in other words, the equations are derived assuming that the members of the structure are of the same, or nearly the same, direction before and after the deformation.

7.2 Member Stiffness

Member stiffness stiffness is in two forms - $\bar{\mathbf{k}}$. member stiffness in member axes; and $\mathbf{k}$, member stiffness in global axes. The following equation shows the meaning of member stiffness in member axes:

$$\bar{\mathbf{P}} = \bar{\mathbf{k}} \cdot \bar{\mathbf{u}} \tag{7.6}$$

Member axes are so chosen that the calculation of $\bar{\mathbf{k}}$ is as simple as possible. The determination of $\mathbf{k}$ done by the basic principles of applied mechanics or by method of virtual forces. Conceptionally simplest is to assume that

one end of the member is held firmly and the forces at the other end cause deformations. By equilibrium considerations, member forces in member axes are related to the member forces in global axes, ref. (7.1) We chose to pre-multiply both sides of 7.2 by $\mathbf{B}$ to yield:

$$\mathbf{B}.\bar{\mathbf{P}} = \mathbf{B}.\bar{\mathbf{k}}.\bar{\mathbf{u}} \tag{7.7}$$

Substituting the contragredience equations (7.1) in to the above equation we have:

$$\mathbf{P} = \mathbf{B}.\bar{\mathbf{k}}.\mathbf{B}^T \cdot \mathbf{u} \tag{7.8}$$

and thus:

$$\mathbf{k} = \mathbf{B}.\bar{\mathbf{k}}.\mathbf{B}^T \tag{7.9}$$

The above equation is the general equation for the member stiffness. The stress strain relations are involved only in the determination of $\bar{\mathbf{k}}$.

Appendix A

Appendices

A.1 Python

All programs in this text are written in Python programming language. It is an interesting interpreter, capable of a very varied use. It is well documented with many books and reference material. Once you, the reader, begin the journey into Python programming language, you will wonder why you did not use it before.

A word of caution: please be patient. Python often upsets users of more traditional programming languages. Its structure is based on *indentation of program source.* It imposes on programmer the need to write – readable programs, with the blocks of indentation that do correspond to *Program Blocks.* In Pascal we would use 'begin' to start a block and 'end' to finish a block. In C the corresponding symbols are '{' and '}'. In those languages the recommended practice is to indent those blocks for readability. In Python it is mandatory to indent those blocks - no freedom to write hard to read code!

On the first encounter with Python the above described feature can cause upset and confusion. Sometimes the knee-jerk reaction is *Is this a return to the long forgotten Fortran?* Rest assured, it is not. Python is versatile and is organised to minimise the amount of typing. The consiceness of Python can be offputting for the beginner.

There are many sources in both the printed form and the freely accessible learning material on the internet. I will endeavour to concentrate on the latter.

Perhaps the best known introduction to Python is the *Tutorial*, written by the autho or Python, Guido van Rossum. It is somewhat unusual 'tutorial', because it is written very concisely and yet it covers many aspects of Python programming. It is freely available at http://docs.python.org/tut/.

Another source of basic material about Python is a book *How to Think Like a Computer Scientist* by Allen B. Downey, Jeffrey Elkner and Chris Meyers. It is available at http://www.ibiblio.org/obp/thinkCSpy/. There are numerous other sources for learning Python, but even with these two there is more than enough material to get you going.

Traditionally, the first task in a new programming language is to write a program that outputs a message 'Hello world!'. Simple enough task that gets you to install the program ans some kind of interaction with it. For interaction with it we will use python-idle. Idle is a simplest programming environment, which is available on any platform that has Python installed. That is the main reason for sticking with it. The next section deals with installation of Python under Linux and the second-next section is the installation of Python under win32 platform. On first reading, windows users may skip the next section, whilst the Linux users may postpone reading of the second-next section.

A.1.1 Python for Linux

As Python was developed in the free software world, free as in *free speech*, it is sensible to discuss Python in a Linux (or Unix) environment.

A.1.2 Python for Windows

There are many sources of information about Python under various flavours of Windows. IMHO, installation of Python interpreter and the numpy module are simpler under Windows than under Linux.

A.2 Program for Structures

In the revision of the text, all mainlines of Structural analysis programs that were written as part of this text, were merged into one program, named *fem.py*. It turns out that the modified program could deal with any Finite Element Method element, provided that the element properties were handled by a suitable subroutine. Hence, it seems sensible to call it *Finite Element Method*, in full *fem.py*.

The computer programs are an integral part of this introduction to computer oriented structural analysis. The reader is urged to read them carfully and to solve examples, using the program. A minimal familiarity with Python interpreter is a prerequisite. Python interpreter is available freely for all major operating systems

The listing is omitted to save paper and trees that the paper is made from.

A.3 Notation

Coordinates - Symbols for coordinate axes: $\bar{x}$ member coordinate along the member axis from end 1 to end 2

$\bar{y}$ member coordinate in member axes at the right angle to the member

x member coordinate in structure axes, usually horizontal

y member coordinate in structure axes, usually vertical

Forces - symbols for force vectors:

$\mathbf{P}$ Force vector:

$\bar{\mathbf{P}}$ Force vector in member axes (not global);

$\mathbf{P}$ Force vector in "global" axes (member axes in the global direction);

$\mathbf{P_s}$ Structure force vector.

Displacements - symbols for displacement vectors:

$\bar{\mathbf{u}}$ Member displacement vector in member axes;

$\mathbf{u}$ displacement vector in "global" axes (member axes in global direction);

$\mathbf{u_s}$ Structure displacement vector.

Matrices - symbols for matrices.

$\mathbf{X}$ Matrix X

$\mathbf{X}^T$ Matrix X transposed.

$\mathbf{X}^{-1}$ Inverse of Matrix X.

$\bar{\mathbf{k}}$ Member stiffness matrix in member axes.

$\mathbf{k}$ Member stiffness in "global" axes.

$\mathbf{K_s}$ or simply $\mathbf{K}$ Structure Stiffness

A.4 Version Table

Vers.	Date	Comment
0.00	2006-04-27	First contact with LaTeX and latex2html.
0.01	2006-05-11	A version suitable for upload.
0.02	2006-05-22	Added checking of trusses.
0.03	2006-05-27	Editing of some LaTeX commands.
0.04	2006-05-28	New brackets $[\cdots]$ for matrices.
0.05	2006-05-31	Virtual work for trusses added.
0.06	2006-06-23	Plane Frames, incl stiffness added
0.07	2006-08-14	Added section on SMAP
0.08	2006-10-31	Corrections by Lex Mulcahy
0.09	2006-12-28	Moved version table, example solution added
0.10	2009-03-17	Added space trusses and a note on new data format
0.11	2009-04-15	Transferred all tex files to the repository

todo ... RSN ... prove Virtual work for frames. Add an example of flexibility method for single degree of redundancy in trusses to reflect plane frames. ... Urgently add the very short introduction to Python.

Index