Mode-Jumping with B2000

J.J.C. Remmers

Delft University of Technology
Faculty of Aerospace Engineering

Graduation Committee

• Dr. ir. E. Riks [1]	Thesis advisor
• Prof. Dr. J. Arbocz [1]	
• Ir. G. Rebel [1]	
• Dr. ir. A. de Boer [2]	

[1] University of Technology Delft
[2] National Aerospace Laboratory (NLR)

The graduation project which is described in this thesis is performed at the department of Aerospace Structures and Computational Mechanics of the faculty of Aerospace Engineering at Delft University of Technology in the period July 1997-October 1998.

This report contains 178 pages.

Opdracht

Contents

Glossary of Symbols

$\mathbf{A}$	amplifying factor; arbitrary, non-singular matrix
$\hat{\mathbf{A}}$	Antman parameters transformation parameters
A	beam's cross section
$\mathbf{B}$	arbitrary, non-singular matrix
$\mathbf{C}$	damping matrix
$\bar{\mathbf{C}}$	constitutive matrix (material description)
$\bar{\mathbf{c}}$	constitutive matrix (spatial description)
C_{2D}	set of possible configurations of 2 dimensional beam
C_{3D}	set of possible configurations of 3 dimensional beam
$\mathbf{d}$	additional position vector (3D beam)
$d_{i,i=1\ldots4}$	Antman's alternative deformation parameters (2D)
$\mathbf{E}_i$	fixed base vectors
$\mathbf{e}_i$	frame attached to cross-section (undeformed beam)
E	Young's modulus
$\mathbf{f}^{\mathrm{ext}}$	external forces vector
$\mathbf{f}^{\mathrm{int}}$	internal forces vector: $\mathbf{f}^{\mathrm{int}}(\mathbf{u}) = \mathbf{K}\mathbf{u}$
G	shear modulus
$\mathbf{G}_i$	global coordinate axes
$\mathbf{g}$	dynamic force vector
h	time step
h_β	modified time-step
$\mathbf{h}$	history vector
I	moment of inertia (2-dimensional beam)
I_x	moment of inertia about x-axis
I_y	moment of inertia about y-axis
i	iteration number
$i_{\max}$	maximum number of iterations per step
J	moment of gyration, Jacobian
$\mathbf{K}$	stiffness matrix
L	initial beam length
$\mathbf{M}$	mass matrix
N	number of degrees of freedom of the total construction
N_i	Lagrange polynomial
n	step number
$n_{\max}$	maximum number of steps

nel	number of nodes per element
Q	dissaptive energy
$\mathbf{r}$	position vector of beam mid axis
r_i	Gauss quadrature sampling points
r	radius of gyration
S	beam's cross section
s	arclength coordinate
$\mathbf{T}$	transformation matrix (element local to branch global)
$\mathbf{t}_i$	frame attached to beam cross section (deformed beam)
T	kinetic energy
t	time
$\mathbf{u}$	displacement vector, incremental displacement (3D)
$\dot{\mathbf{u}}$	velocity vector
$\ddot{\mathbf{u}}$	acceleration vector
$\mathbf{v}$	auxiliary vector used in Jensen algorithm
$\dot{\mathbf{v}}$	first differentiation of auxiliary vector

α	Rayleigh's damping coefficient, $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$
α_i	linear multi-step iteration constants; Gauss quadrature weight factors
β	Rayleigh's damping coefficient $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$
β_i	linear multi-step iteration constants
ϵ	axial strain
$\boldsymbol{\Gamma}$	strain vector (spatial description)
$\boldsymbol{\gamma}$	strain vector (material description)
γ	shear strain
$\mathbf{K}, \boldsymbol{\kappa}$	curvature tensor, axial vector (material description)
$\boldsymbol{\Lambda}$	rotation tensor
λ_l	l^{th} eigenvalue of free vibrating structure
λ	load factor
η	path parameter
μ_l	eigenvector of amplification matrix
ν	Poisson's ratio
ρ	mass density
$\boldsymbol{\Omega}, \boldsymbol{\omega}$	curvature tensor, axial vector (spatial description)
ω_l	eigenfrequency of free vibrating structure
Ψ	strain energy
$\boldsymbol{\psi}$	eigenvector of free vibrating structure
$\boldsymbol{\Theta}, \boldsymbol{\theta}$	incremental rotation tensor, axial vector
$\bar{\boldsymbol{\Theta}}, \bar{\boldsymbol{\theta}}$	Rodrigues rotation tensor, axial vector
$\boldsymbol{\Xi}$	differential tensor
ξ	normalized arc-length parameter; damping ratio
$\boldsymbol{\Upsilon}$	generalized lumped mass matrix

Subscripts

d	dynamic solution
e	element local coordinates
G	(branch) global coordinates
I	Node number
n	step number
s	static solution

Superscripts

f	free (not prescribed) vector
i	iteration number
p	prescribed vector
SKEW	skew-symmetrical part of matrix
SYM	symmetrical part of matrix

Abbreviations

d.o.f.	degree of freedom
EEM	eindige elementen methode
FEM	finite element method
LMS	linear multi-step
ODE	ordinary differential equation

Abstract

In the past years a lot of research has been done to analyse buckling behavior of structures in general and post-buckling behavior in particular. A lot of attention is paid to the mode-jumping phenomenon. When a structure is loaded beyond the limit point, the deformation mode jumps to another stable mode. This attends large velocities and is therefore a purely dynamic process. In finite element (FEM) analysis this behavior can be simulated with a transient solving routine.

In this thesis the numerical aspects of the mode-jumping process of beam structures are considered. For this purpose, two new features are implemented in the finite element package `B2000`; a set of beam elements and a nonlinear transient solver. In principle, the development of both can be seen as two different things.

The three beam elements can be divided into two categories. The 2-dimensional beam elements `B2.EP` and `B2.EP+` are based on papers by Eriksson and Pacoste [5, 6]. The strain and bending is based on a model by Reissner. The behavior of this type of beam is even in the post buckling region still correct as opposed to the beams governing the Lagrange-Green strain formulation. Furthermore the Bernoulli hypothesis is applied. This implies that the beams cannot deform in a pure shear state. Due to the absence of out-of-plane deformations and the corresponding 3-dimensional rotation tensor, the theoretical background is rather straightforward. The 2-dimensional beam elements serve as a starting point for the development of a fully 3-dimensional beam.

The 3-dimensional beam element `B2.NL` is based on papers by Simo *et al.* [29, 30]. It is a so-called finite rotations element: the nonlinear behavior of these elements is correct for large rotations. It can therefore be added to the same family of elements to which Rebel's shell elements are reckoned. The description of the rotation is based on the classical formulation which is proposed by Rodrigues [3]. The strains are described using Reissner's formulation, just as with the 2-dimensional elements. The Bernoulli hypothesis is not used. At the moment, the material properties are fully elastic.

As opposed to the 2-dimensional beam element, the 3-dimensional element does not work properly in nonlinear calculations at the moment. There may be two reasons for this. First, there might be some locking phenomena (membrane locking) which has not been tackled properly. Second, the description of the three dimensional rotations might be wrong.

All beam elements have the same formulation for the mass matrix. The total mass of the beam is divided over the 2 nodes. The inertia terms (the rotational mass) are neglected, Although this formulation is simplified, it appeared to be very effective in transient analyses. This mass representation is therefore also applied to Rebel's shell elements and the cable elements in B2000.

The implicit transient processor B2TRANS is based on the B2IDTI processor, written by K. Yildirim [34]. The equations of motion are solved by a special group of implicit time integration methods, the linear multi-step (LMS) methods. Using Jensen's algorithm [13] the system of second order differential equations is transformed into a first order one. Park's LMS method is used to transform this ODE into a nonlinear system of equation, which is solved iteratively, using the Newton Raphson method.

The simulation of mode-jumping phenomena is described by Riks *et al.* [26, 27, 28]. In his papers he suggests fast and accurate simulation technique. The stable, pre-buckling deformation behavior, up to the limit points is calculated according to the ordinary Riks' path-following method [25]. As soon as the equilibrium becomes unstable (this can be checked by examining the decomposed stiffness matrix), the response is calculated with the transient solver. The load is raised to a level that is higher than the limit load. The structure is released from this point. Since the corresponding equilibrium is unstable, the structure will leave this primary path with high velocities and go to a new, stable equilibrium on the secondary stable path with a different deformation mode. Apart from the 2-dimensional beam element, it is also possible to perform simulations with the the finite rotation shell elements.

A number of numerical examples is used to show the reliability of the beam elements and the transient macro-processor. There are also some examples that show mode-jumping simulations. As can be seen from the results of these tests, the new features (except for the nonlinear 3-dimensional element) work decently. The performances of the new, simplified mass description in mode-jumping simulations are also good.

In the near future the solution of the problems with the 3-dimensional beam element has got the highest priority. Since the element works perfectly in linear analyses, it is most likely that just some small fixes will be enough. When this is done some other aspects can be improved. All beam elements can be equipped with a plasticity model. The LMS algorithm as implemented in the transient processor can be adapted in order to be able to calculate dynamic behavior with large rotations of structures. This also requires a closer look to the nonlinear mass matrix. The improved elements and processor can be applied in the continuing research of the mode-jumping phenomenon. Especially the influence of the initial conditions and the damping constants must be regarded.

Samenvatting

De laatste jaren is er veel onderzoek gedaan naar knik gedrag in het algemeen en naknik gedrag in het bijzonder. Het 'mode-jumping' verschijnsel krijgt hierbij steeds meer aandacht. Wanneer een constructie wordt belast tot voorbij zijn limietpunt slaat het vervormingspatroon (de 'mode') razendsnel om (de 'jump') naar een andere, stabiele vervorming. Dit alles gaat gepaard met hoge snelheden en is daarom een puur dynamisch proces. In eindige elementen (EEM) berekeningen kan dit gedrag gesimuleerd worden met een transiente oplosmethode.

In dit afstudeerrapport worden de numerieke aspecten van mode-jumping van balkconstructies beschouwd. Hiervoor zijn twee nieuwe onderdelen in de EEM code `B2000` geïmplementeerd: een serie balkelementen en een niet-lineaire transiente oplosmethode. De ontwikkeling van deze twee zaken staat in principe los van elkaar.

De drie balkelementen kunnen worden onderverdeeld in twee catagorieën. De twee 2-dimensionale balkelementen `B2.EP` en `B2.EP+` zijn gebaseerd op artikelen van Eriksson en Pacoste [5, 6]. De rek en de buiging is gebaseerd op een model van Reissner. Het gedrag van dit type balk in het na knik gebied is nog steeds correct, in tegenstelling tot de balken die uitgaan van de klassieke Lagrange-Green rek formulering. Tevens is de Bernoulli hypothese toegepast. Dit betekent dat zuivere afschuif vervorming niet meer mogelijk is maar dat deze wordt omgezet in buiging. Door het ontbreken van uit het vlak verplaatsingen en de daarbij behorende 3 dimensionale rotaties is de theoretische achtergrond van deze balken tamelijk eenvoudig. De 2-dimensionale balken vormen dan ook een goed uitgangspunt voor de volledige 3-dimensionale balk.

Het 3-dimensionale balk element `B2.NL` is gebaseerd op artikelen van Simo c.s. [29, 30]. Dit element is een zogenaamd *eindige rotaties* element. Het niet-lineaire gedrag van dit element is ook bij zeer grote rotaties correct. Het element kan daarom in principe worden toegevoegd aan de familie van elementen waartoe ook G. Rebel's schaal elementen behoren [23]. De beschrijving van de rotatie is gebaseerd op de klassieke formulering zoals die is voorgesteld door Rodrigues [2]. Net als bij de 2-dimensionale balken is er gebruik gemaakt van Reissner's beschrijving van de rek. De Bernoulli hypothese is niet toegepast. Op het moment zijn alle materiaal eigenschappen van het materiaal lineair elestisch.

In tegenstelling tot het 2-dimensionale balkelement werkt het 3-dimensionale balkelement nog niet in het niet-lineaire geval. Dit kan twee oorzaken hebben.

Er kan een locking probleem zijn dat niet met de methode van gereduceerde numerieke integratie opgelost kan worden. Het kan ook zijn dat de beschrijving van de 3-dimensionale rotatie tensor niet geheel correct is.

Alle balkelementen hebben dezelfde formulering voor de massa matrix. Hierin is de totale massa verdeeld over de beide knooppunten. De traagheidstermen (de massa in de rotatie vrijheidsgraden) zijn verwaarloosd. Dit sterk vereenvoudigde massa model blijkt echter zeer effectief bij transiente analyses.

De impliciete, transiente processor `B2TRANS` is gebaseerd op de `B2IDTI` processor van K. Yildirim [34]. De basis van deze processor wordt gevormd door een speciale groep implicite tijdsintegratie methoden, de lineaire meer stappen (LMS) methode. Met behulp van het algoritme van Jensen [13] wordt het stelsel tweede orde differentiaal vergelijkingen omgezet in een stelsel eerste orde vergelijkingen. Met de methode van Park [7, 20, 21] worden deze differentiaal vergelijkingen getransformeerd naar een niet-lineair stelsel, dat wordt opgelost met de Newton-Raphson methode.

Simulatie van mode-jumping verschijnselen is veel beschreven door Riks c.s. [26, 27, 28]. Zijn methode is erop gericht om zo snel en zo nauwkeurig mogelijk het verschijnsel te simuleren. Dit betekent dat het stabiele vervormings gedrag, tot aan het knik punt, met de gebruikelijke Riks' padvolg methode wordt berekend [25]. Zodra het statische evenwicht instabiel wordt (wanneer de stijfheidsmatrix singulier is), wordt er overgegaan op de transiente oplosmethode. De belasting wordt verhoogd tot boven de limietwaarde en de constructie wordt losgelaten. Omdat bij deze belasting het evenwicht instabiel is, zal de constructie zich met grote snelheid verwijderen van dit evenwicht en tot rust komen op een ander stabiel pad met een ander vervormingspatroon. Dit type berekeningen vereist een aantal aanpassingen in zowel de padvolgmethode (continuatie routine) `B2CONT` als de transiente processor `B2TRANS`. Behalve met de balkelementen is het ook mogelijk om mode-jump analyses uit te voeren op schaalconstructies met Rebel's eindige rotatie schaalelementen.

Met behulp van een aantal numerieke voorbeelden is tenslotte de werking van de balkelementen en de transiente oplosmethode uitgebreid getest. Een aantal voorbeelden heeft betrekking op het mode-jumping verschijnsel. Uit deze voorbeelden blijkt dat alle nieuwe toevoegingen, het 3-dimensionale balkelement uitgezonderd, naar behoren werken. Ook de mode-jump analyses kunnen nu worden uitgevoerd, op zowel de 2-dimensionalebalk elementen als de eindige rotatie schaalelementen.

In de nabije toekomst zullen eerst de problemen met het 3-dimensionale nietlineaire balkelement opgelost moeten worden. Omdat het element in het lineare geval werkt zal het slechts om een kleine aanpassing gaan. Daarna kunnen er andere zaken verbeterd worden. Het balk element kan worden uitgerust met een plasiticiteits model. Het LMS algoritme achter de transiente processor moet zodanig worden aangepast dat het ook voor grote rotaties nauwkeurige resultaten levert. Hierbij moet ook gekeken worden naar de mogelijkheden van een niet-lineare beschrijving van de massa matrix. De verbeterde elementen en de transiente processor kunnen worden ingezet bij het verdere onderzoek naar mode-jumping. Vooral de invloed van beginvoorwaarden en dempings constanten moeten hierbij beschouwd worden.

Acknowledgments

The research which is described in this thesis is carried out at the department of Aerospace Structures & Computational Mechanics at the faculty of Aerospace Engineering, Delft University of Technology with the help of a number of people. First of all, I would like to thank my supervisor, Dr. ir. E. Riks for his guidance during this project and his insights in nonlinear computational analysis. I would also like to thank ir. G. Rebel. Without his help, the implementation of the elements and processor into the computer program `B2000` would be much more complicated. Furthermore I would like to thank the other two members in the committee, Prof. Dr. J. Arbocz and Dr. ir. A. de Boer, representing the Dutch Aerospace Laboratory NLR.

There are some other people that deserve credits; ir. R. Hordijk for the maintenance of the computers and the installed software; all the students and AIO's of the department for the their support and pleasant atmosphere to work in and Prof. Dr. ir. Clement for taking his time introducing me into functional analysis. Finally I would like to thank my parents for supporting me during my study at this university and Irene for her patience and care.

Joris Remmers

1 Introduction

Simulation of nonlinear phenomena, such as buckling, post-buckling behavior, dynamic buckling and mode-jumping problems are an important field in computational mechanics nowadays. The finite element method (or FEM in short) is frequently used to simulate these nonlinear problems.

The main idea behind the finite element method is the division of a structure into a number of small, but finite elements, of which the mechanical properties can be derived analytically. This division is often called *discretization* of the structure. Due to this discretization, the mechanical behavior of the structure can be described by a system of equations. The number of equations in this system (or so-called *degrees of freedom*) is proportional to the number of elements that is used in the discrete model. In very accurate models the number of degrees of freedom can exceed 100,000.

A system of equations of this size cannot be calculated by hand. It is therefore not surprising that the first serious applications arose together with the arrival of computer technology in the mid fifties. These computer programs were designed by NASA and were used in research projects. The first commercial platforms arose around 1970. In the years after, these platforms were further developed in order to satisfy specific needs. Nowadays, a large number of different programs is available. Some of them can be used for designing purposes in combination with CAD systems, like `MSC NASTRAN`, other ones are intended for analysis or scientific research, e.g. `B2000` and `STAGS`.

The finite element method is a classical example of an engineering method. It is rather simple and straightforward and does not require mathematical skills of the user. At the moment it is the most widely used method in computational mechanics. Also in thermodynamics and acoustics it has become the leading calculation method. Strangely enough, finite element applications penetrated other fields of physics more slowly. For example in fluid dynamics, the finite difference method is still preferred to the finite element method.

1.1 The B2000 Platform

The B2000 finite element package has been developed in the mid eighties by S. Merazzi and P. Stehlin. They developed B2000 using their experiences with other programs they previously worked with. All these programs had the unpleasant circumstance that they could not be manipulated at all. The implementation of new elements or solving methods was impossible.

To overcome these problems, the B2000 package is not placed in a rigid format. It is modular, hardware independent and most important it can be customized by the user to serve its specific needs. New solution techniques as well as element descriptions can be implemented in the platform rather easily using the B2000 building blocks. The presence of an independent database manager, post-process utilities and a robust equation solver have even lead to the development of FEM related packages like the explicit B2ETA solver and the optimization platform B2OPT.

Because of these properties, the platform is mainly used in a research environment. B2000 is used as a testbed at a number of universities and aeronautical institutes in western Europe. Approximately 30 people, among them a number of students, are improving and adding new features to the package.

1.1.1 The Processors

The actual building blocks of B2000 are the processors, which are able to perform all necessary operations in finite element calculations. For example, there is a processor that solves systems of equations by an LDL decomposition (B2ES), there are also processors that assemble the stiffness matrix of a structure (B2EP and B2EPN).

Macroprocessors are integrated sets of algorithms built with the processors mentioned above. They can be used independently within the B2000 platform. A very important and indispensable macro-processor is the input processor B2IP. It translates the ascii text input file into the *archival database* which is written in a standard B2000 file format. Every analysis must be started with a B2IP session. The obtained database is used by the analysis macro processors, that perform the actual computations, for example B2CONT. This macro processor is designed for nonlinear analysis based on Riks' path-following technique.

1.1.2 The Elements

A variety of elements has already been implemented in the package and their number is still increasing. Besides the traditional elasticity elements (beam, shell and volume elements), acoustic and heat transfer (Laplace-type) elements are available as well. Some of the elasticity elements are equipped with plasticity or crack models.

A new generation of elasticity elements is currently under development by G. Rebel [23]. These finite rotation shell elements are capable of handling large strains and rotations and are the first ones that deal with the problems concerning the so-called drill stiffness.

1.1.3 The MEMCOM data base manager

All data produced by the B2000 platform is managed by MEMCOM [18], an independent database manager, developed by S. Merazzi. The data is stored in a binary format, instead of ascii text. The biggest advantage of this binary format is that it is much more compact than ascii and less sensitive for I/O errors.

All MEMCOM operations, e.g. read and right tasks, can be controlled in the B2000 source code by using ordinary C or Fortran subroutine calls. Within the database, all data is organized in folders, the *datasets.* The B2000 package uses a hierarchical notation method. For example, the displacements in the 6^{th} time step are stored in the dataset,

```
DISP.GLOB.6
```

where DISP is the name of the data set. The second entry is reserved for the name of the coordinate system in which the data is expressed, in this case a global coordinate system. The last entry in this example represents the cycle number. Additional information to a certain dataset can be stored in a *description table.*

Besides the standard read and write functions, MEMCOM can also be used to manipulate data in the database. For example, when two datasets need to be summed, it is not necessary to read both datasets, add them and write the result back to the database. The summation can be done within the database itself.

Since all data is stored in a binary format, it is not possible to view the contents of the database by using an ordinary text-editor. An additional program that can be used to look in the database is the monitor program or its graphical version Xmon. These programs can also be used to manipulate the datasets.

1.1.4 Post Processing

The results in the database can be presented in a graphical lay-out using the post-processing programs B2BASPL and B2XY. The B2BASPL program visualizes all possible results (e.g. deformations and stresses) by means of various techniques such as colored contour plots. The package is optimized for high performance graphic workstations, like Silicon Graphics systems. History functions or xy plots can be made with the B2XY package.

It is of course possible for the user to write his own post-processing program within the B2000 environment. Data from the database can be read using MEMCOM commands and written in a different format. In this report, the data is obtained from the computational database with a new macroprocessor B2GNU. The gnuplot package is used to plot the results.

1.2 Current Developments in B2000

As said before, the B2000 package is mainly used in a research environment. As a result of this, the package is always under development and new features are added continuously. In the last years the number of institutes participating in the B2000 project has increased rapidly.

At SMR, a consulting company founded by S. Merazzi and P. Stehlin, all developments on B2000 are gathered for further distribution. Furthermore, the MEMCOM package as well as B2BASPL is under development at this company. The National Aerospace Laboratory (NLR) coordinates development on the package in the Netherlands and collaborates with both Universities in Delft and Twente. The optimization routine B2OPT is created here as well as the macro-processor B2TEST. This last package can be used to validate new versions of the B2000 package. Germany's national aerospace research center (DLR) uses B2000 as a platform to do their research of thermal analyses.

At the École Polytechnique Fédérale de Lausanne, optimization and parallelization of the B2000 source code is one of the main topics. Currently, a parallel version of the explicit time integration method B2ETA is under development. The University of Twente uses the package for their work in the field of acoustics. A number of acoustic and viscous elements are developed in cooperation with the NLR. At the faculty of Aerospace Engineering in Delft the developments are mainly focussed on nonlinear structural analysis such as buckling behavior, plasticity models and crack propagation.

1.3 Research Objectives

The main objective of this research is the simulation of mode jumps in beam structures. This requires two new numerical models. First an appropriate nonlinear beam element must be derived. This involves the derivation of a residual (internal) forces vector and a stiffness matrix. Since mode jumps are transient (dynamic) processes a mass matrix must be derived as well. Second, in order to calculate the actual jump, a time integration method must be chosen. Since for the calculation of modejumps also quasi-static calculations are required, this method must be based on the same numerical principles as the continuation routine. This implies that the time integration method (transient method) must be an implicit method.

Both the beam elements and the time integration method will be implemented in the B2000 package. In order to improve the suitability, the new features in B2000 must be widely applicable within the package. This means that the beam elements, as well as the transient processor must also be available for other analyses.

After all, this study covers two extreme regions of computational mechanics. The development of the beam elements is mainly based on mechanical and material principles as opposed to the more mathematical founded transient method. Because of this, this report also provides a good survey of the different aspects of finite element calculations.

1.4 Overview of the Thesis

The thesis can be divided into 2 main parts. In the first part the development of the beam elements is discussed. In chapter 2 two 2-dimensional beam elements are derived. Both beams are based on papers by Eriksson and Pacoste [5, 6]. In chapter 3 the two dimensional beam element is extended to a full 3-dimensional beam element. The kinematics of this beam, as well as the numerical implementation is based on the work of Simo and VuQuoc [29, 30]. In the second part solution techniques to solve the mode-jumping process are described. Chapter 4 discusses the transient time integration method [13, 21] and the development of the `B2000` macroprocessor `B2TRANS`. In chapter 5 the techniques to calculate mode-jumps are proposed.

Numerical examples to illustrate the capacities of the beam elements and the transient processor `B2TRANS` are given in chapter 6. The numerical examples are compared to analytical solutions, results obtained from literature or other finite element packages. The conclusions and recommendations are given in chapter 7. The appendices are reserved for additional information about the developed programs. Appendix A contains a user manual for the transient processor. A short outline of the syntaxis of the source code of `B2TRANS` is given in B. Finally a list of all created and modified Fortran source files is printed in appendix C.

Throughout this thesis, a consistent notation is used in order to make a distinction between scalars, vectors and matrices. All scalars are written in thin letters, the vectors are denoted by small bold letters, matrices are in bold capitals. These conventions also holds for vectors and tensors that are prescribed on the SO(3) space which will be discussed in chapter 3: Skew-symmetric tensors are written in bold capitals, their axial vectors in bold small letters.

2 A 2-dimensional beam

In the development of a fully nonlinear, 3-dimensional beam, the 2-dimensional beam is a good starting point. Because of the small number of degrees of freedom (six, i.e. three per node) and the fact that there are no zero-energy modes (like for example in cable or rod type elements), this beam model is rather straightforward. Complications due to the description of 3-dimensional finite rotations of the beam do not appear either.

However, strictly speaking, the 2-dimensional beam is just of academic interest. It is obvious that it is not possible to use it in ordinary 3-dimensional structures. Nevertheless, due to its surveyability, the 2-dimensional beam model is often used to test new finite element solution techniques. For example, the new implicit time integration method `B2TRANS`, which will be discussed in chapter 4, is tested for its reliability with this element. Also literature provides a large amount of examples and 'bench mark tests' of nonlinear finite element calculations with plane beam structures.

The beam elements described in this chapter are finite strain beam models introduced by Reissner and further worked out by Eriksson and Pacoste [5, 6]. At this moment, these elements are considered as quasi-static beams. This means that there is no representation for the mass derived yet. In a later stage, when the time integration method will be discussed, a proper formulation for the mass will be added.

The beams are assumed to be slender. This means that, compared to their length, the radius of gyration, $r = \sqrt{I/A}$ is very small, $r/L << 1$. Furthermore, the beam is made of an isotropic linear elastic material. This implies that Young's modulus E is constant for all deformations and that there are no cross coupling terms between the various deformation modes.

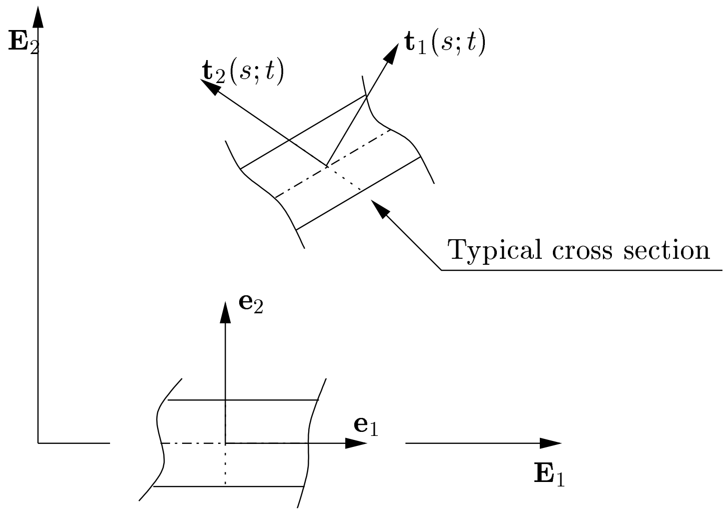


Figure 2.1: Geometrical conventions for the 2-dimensional beam

2.1 Kinematic Model

Since the beam is assumed to be slender, it can be considered as a one dimensional object in a 2-dimensional space. All variables of the beam are expressed in terms of the beam mid axis, along which the arclength parameter s is defined, $s \in [0, L]$ where L is the length of the beam. Note that in a deformed configuration this beam mid axis, the dash-dotted line in figure (2.1), can be curved.

In each point of the beam mid axis, a typical cross section is defined, denoted by the dashed line in the figure. In undeformed position, the cross section is perpendicular to the beam axis. There are two orthonormal vectors attached to this typical cross section, $\mathbf{t}_1(s;t)$ and $\mathbf{t}_2(s;t)$, both a function of the arclength s and time $t \in \mathbb{R}^+$. The vector $\mathbf{t}_1(s;t)$ is always perpendicular to the cross section. In the undeformed state, the direction of these vectors is equal to the direction of the element local base vectors $\mathbf{e}_1$ and $\mathbf{e}_2$.

$$\mathbf{t}_1(s;0) = \mathbf{e}_1; \qquad \mathbf{t}_2(s;0) = \mathbf{e}_2 \tag{2.1}$$

The vector $\mathbf{t}_1$ need not be tangential to the beam axis in deformed position. In the sequel, the time parameter t will be omitted.

2.1.1 Deformations

The deformation of the beam is fully described when the position of the beam axis and the rotation of the cross section are known as a function of the arclength s. The position of the beam axis in the 2-dimensional space is denoted by the vector $\mathbf{r}(s)$, as shown in figure (2.2). The orientation of the typical cross section (the vectors $\mathbf{t}_1(s)$ and $\mathbf{t}_2(s)$) is described by the angle $\theta(s)$. The set of possible deformed configurations $\boldsymbol{\phi}(s) = (\mathbf{r}(s), \theta(s))$ of the 2-dimensional beam is denoted by

$$C_{\text{2D}} = \{\boldsymbol{\phi} = (\mathbf{r}, \theta) | \mathbf{r} : (0, L) \to \mathbb{R}^2, \theta : (0, L) \to \mathbb{R}\} \tag{2.2}$$

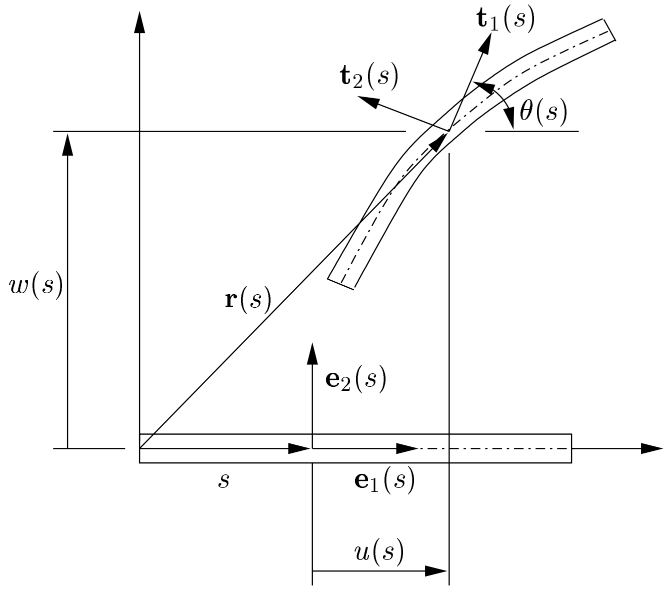


Figure 2.2: Description of the displacements for the 2-dimensional beam

In this case, it is possible to express the vector $\mathbf{r}$ in terms of its vector components with respect to the fixed base $\mathbf{e}_i$, according to the deformations in the figure.

$$\mathbf{r} = [s + u(s)]\mathbf{e}_1 + w(s)\mathbf{e}_2 \tag{2.3}$$

The direction of the coordinate axes attached to the cross-section can be expressed in terms of $\theta(s)$.

$$\mathbf{t}_1(s) = \cos\theta(s)\mathbf{e}_1 + \sin\theta(s)\mathbf{e}_2 \tag{2.4}$$

$$\mathbf{t}_2(s) = -\sin\theta(s)\mathbf{e}_1 + \cos\theta(s)\mathbf{e}_2 \tag{2.5}$$

Both $\mathbf{t}_1(s)$ and $\mathbf{t}_2(s)$ remain orthonormal.

2.1.2 Strain and Curvature

In order to describe the strains in the beam, the strain model as proposed by Reissner is chosen. The strains are first defined in the moving frame.

$$\mathbf{r}' = (1 + \epsilon)\mathbf{t}_1 + \gamma\mathbf{t}_2 \tag{2.6}$$

where ϵ is the normal strain and γ is the shear strain. The operator $(\ )'$ is the derivation of the vector with respect to the arc-length parameter $\frac{d}{ds}$. The curvature κ is by definition equal to the derivative of the rotation of the cross section.

$$\kappa = \theta' \tag{2.7}$$

The rotation θ and the curvature κ are invariant of the coordinate system. This can be seen as follows. The rotation is defined around a fictitious out-of-plane vector, say $\mathbf{t}_3$, perpendicular to both $\mathbf{t}_1$ and $\mathbf{t}_2$. However, since all deformations of the beam are defined in a 2-dimensional space, the direction of this fictitious vector does not change at all. In other words the identity $\mathbf{t}_3(s;t) \equiv \mathbf{t}_3(s;0) \equiv \mathbf{e}_3$ holds for all $s \in [0, L]$ and $t \in \mathbb{R}^+$. All variables in this direction are therefore invariant.

The expressions for ϵ and γ can be found by taking the derivative of equation (2.3) and rewriting the result in terms of the moving frame $\mathbf{t}_i$

$$(1+u')\begin{bmatrix}\cos\theta\\ -\sin\theta\end{bmatrix} + w'\begin{bmatrix}\sin\theta\\ \cos\theta\end{bmatrix} = \begin{bmatrix}1+\epsilon\\ \gamma\end{bmatrix} \tag{2.8}$$

After regrouping the terms of this equation, the strain and curvature can be written as follows

$$\begin{aligned}
\epsilon &= (1+u')\cos\theta + w'\sin\theta - 1\\
\gamma &= w'\cos\theta - (1+u')\sin\theta\\
\kappa &= \theta'
\end{aligned} \tag{2.9}$$

2.1.3 Bernoulli Hypothesis

In order to avoid shear-locking problems it is convenient to apply the Bernoulli hypothesis. The beam model becomes much simpler by assuming that the derivative of the position vector $\mathbf{r}'$ is in all cases tangential to the director $\mathbf{t}_1$ that is attached to the cross-section of the beam, i.e.

$$\mathbf{r}' = (1+\epsilon)\mathbf{t}_1 \tag{2.10}$$

This implies that the mid axis of the beam will always be perpendicular to the cross-section. As a result of this the shear strain is always equal to zero, $\gamma = 0$. Rewriting equation (2.10)

$$\begin{bmatrix}1+u'\\ w'\end{bmatrix} = (1+\epsilon)\begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} \tag{2.11}$$

This equation gives the following two, independent expressions for the axial strain ϵ

$$\epsilon = (1+u'-\cos\theta)\frac{1}{\cos\theta}; \qquad \epsilon = (w'-\sin\theta)\frac{1}{\sin\theta} \tag{2.12}$$

The curvature κ remains unchanged.

2.1.4 Timoshenko Beam

The curvature of the beam is represented by the derivation of the rotation of the cross-section. This is a quite good approach of the curvature. However, an exact model has been presented by Timoshenko [9]. He assumed the curvature to be the inverse of the radius of curvature of the beam.

$$\kappa_{\mathrm{T}} = \frac{1}{R} = \frac{\theta'}{||\mathbf{r}'||} \tag{2.13}$$

where $||\mathbf{r}'||$ is the eucledian norm of the strain vector: $(\mathbf{r}' \cdot \mathbf{r}')^{1/2}$. Evaluation of this expression yields

$$\kappa_{\mathrm{T}} = \frac{\theta' \cos\theta}{1 + u'} \tag{2.14}$$

2.2 Mechanical Model

Next, the kinematical behavior must be translated into mechanical behavior. First, the constitutive relations must be derived. Then, the strain energy can be calculated.

2.2.1 Constitutive Relations

The beam can be deformed in two ways. It can be enlarged (axial strain) and it can be bent (curvature). Since the beam is presumed to be made of a linear elastic material, there are no cross-coupling effects between these modes. This implies that the constitutive relations or stress-strain relations are rather simple. In a matrix formulation, they can be written as

$$\begin{bmatrix} N \\ M \end{bmatrix} = \begin{bmatrix} EA & 0 \\ 0 & EI \end{bmatrix} \begin{bmatrix} \varepsilon \\ \kappa \end{bmatrix} \tag{2.15}$$

where N and M are the internal axial force and bending moment respectively; E is the Young's modulus[1]. A is of course the beam's cross section and I the moment of inertia. Although the deformation of the beam is described in a 2-dimensional space, both A and I are 'ordinary' 3-dimensional quantities. The dimension of A is $[mm^2]$ and I is $[mm^4]$.

2.2.2 Internal Energy

The strain energy (or elastic energy) is the mechanical energy stored in a deformed structure. When the strains remain within the elastic boundaries of the material, which is always the case since a linear elastic constitutive model is assumed, the strain energy is equal to the work done by the external forces during their application. When the structure is not loaded (or undeformed) the strain energy is equal to zero.

The strain energy Ψ of the beam, as a function of the two deformation modes, can be written as

$$\Psi = \frac{1}{2} \int_0^L EA\epsilon^2 + EI\kappa^2 ds \tag{2.16}$$

[1]Note that both the Poisson's ratio ν and the shear modulus G are not present in this equation. This is of course a result of the Bernoulli hypothesis

Substituting the axial strain and curvature equations gives the following two integral equations:

$$\Psi = \frac{1}{2}\int_0^L EA(1+u'-\cos\theta)^2\frac{1}{\cos^2\theta} + EI(\theta')^2 ds \tag{2.17a}$$

$$\Psi_{\mathrm{T}} = \frac{1}{2}\int_0^L EA(1+u'-\cos\theta)^2\frac{1}{\cos^2\theta} + EI\left(\frac{\theta'\cos\theta}{1+u'}\right)^2 ds \tag{2.17b}$$

It is obvious that these expressions are not useful in practice, since for angles $\theta = (2k+1)\pi, k \in \mathbb{Z}$ the axial strain terms become singular. The alternative formula $\epsilon = (w' - \sin\theta)\frac{1}{\sin\theta}$ can neither be used for the same reason. It is necessary to derive a new expression for the axial strain using the two equivalent relations, as derived in equation (2.12).

In the energy equation, the axial strain just appears as a squared.

$$\epsilon^2 = (1+u'-\cos\theta)^2\frac{1}{\cos^2\theta} \tag{2.18}$$

The square of the alternative formulation yields

$$\epsilon^2 = (w'-\sin\theta)^2\frac{1}{\sin^2\theta} \tag{2.19}$$

By using the identity $\cos^2\theta + \sin^2\theta \equiv 1$, this last equation can be written as

$$\epsilon^2 = (w'-\sin\theta)^2\frac{1}{1-\cos^2\theta} \tag{2.20}$$

Regrouping yields

$$\epsilon^2 - \epsilon^2\cos^2\theta = (w'-\sin\theta)^2 \tag{2.21}$$

From equation (2.18) it can be seen that

$$\epsilon^2\cos^2\theta = (1+u'-\cos\theta)^2 \tag{2.22}$$

Substituting (2.22) into (2.21) gives the following expression for the square axial strain:

$$\epsilon^2 = (1+u'-\cos\theta)^2 + (1-\sin\theta)^2 \tag{2.23}$$

At this point the internal energy equation for both the simplified and the Timoshenko beam can be formed[2].

$$\Psi = \frac{1}{2}\int_0^L EA[(1+u'-\cos\theta)^2 + (1-\sin\theta)^2] + EI(\theta')^2 ds \tag{2.24a}$$

$$\Psi_{\mathrm{T}} = \frac{1}{2}\int_0^L EA[(1+u'-\cos\theta)^2 + (1-\sin\theta)^2] + EI\left(\frac{\theta'\cos\theta}{1+u'}\right)^2 ds \tag{2.24b}$$

[2]In literature, the Bernoulli axial strain is sometimes. wrongly, divided into a pure strain part and a 'pseudo shear strain' part, according to $\epsilon^2 = \bar{\epsilon}^2 + \bar{\gamma}^2$, where $\bar{\epsilon}^2 = (1+u'-\cos\theta)^2$ and $\bar{\gamma}^2 = (1-\sin\theta)^2$. The elastic energy is than written as $\Psi = \int_0^L EA\bar{\epsilon}^2 + EA\bar{\gamma}^2 + EI\kappa^2 ds$. It need no discussion that this notation is incorrect and confusing.

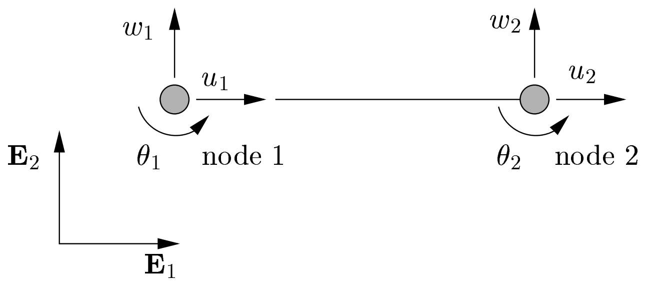


Figure 2.3: Finite element representation of the 2 node, 2-dimensional beam element

2.3 Finite Element Description

Both beam descriptions will be implemented as a 2 node beam element. Since the beam is defined in a 2-dimensional space, each node has 3 degrees of freedom (d.o.f.'s); two translations, u_I and w_I in the $\mathbf{E}_1$- and $\mathbf{E}_2$-direction resp. and one rotation, θ_I. The subscript I denotes the node number ($I = 1, 2$) as can be seen in figure 2.3.

2.3.1 Interpolation functions

Normally, linear interpolation functions are used to describe the relation between the displacements of the nodes and the displacements in an arbitrary point on the beam's mid axis. In this case a method introduced by Antman [5] is used instead. Antman proposed a set of 4 parameters $d_1 \dots d_4$ which are functions of the nodal displacements and rotations

$$d_1 = \arctan\left(\frac{w_2 - w_1}{L + u_2 - u_1}\right); \qquad d_2 = \frac{\theta_1 - \theta_2}{2}; \tag{2.25}$$

$$d_3 = \frac{\theta_1 + \theta_2}{2} - d_1; \qquad d_4 = \frac{u_2 - u_1}{L} \tag{2.26}$$

Note that Antman's alternative deformation parameters are based on linear extrapolation functions. The derivative of the axial displacement u' can be expressed in terms of these parameters.

$$u' = d_4 \tag{2.27}$$

Since the kinematics of the beams are based on the Bernoulli hypothesis the lateral displacement w can be written as a function of the axial displacement using the 2 expressions for axial strain, equation (2.12).

$$w' = (1 + u') \tan\theta \tag{2.28}$$

In terms of Antman's parameters,

$$w' = (1 + d_4) \tan d_1 \tag{2.29}$$

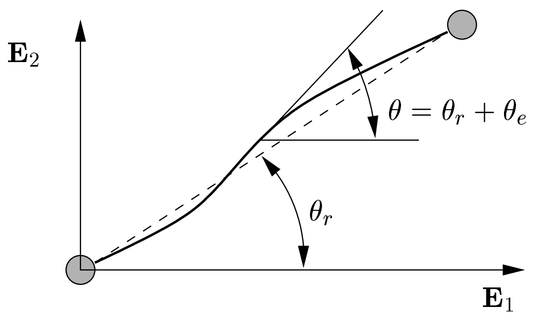


Figure 2.4: Deformed finite element model of a 2-dimensional beam

For reasons of simplicity, the rotation of the beam's cross section θ can be divided into a rigid rotation part θ_r and an elastic rotation part θ_e, according to

$$\theta = \theta_r + \theta_e \tag{2.30}$$

The rigid rotation actually describes the position of the beam in the element local coordinate frame $\mathbf{E}_i$, i.e. the angle of a fictitious line between the 2 nodes and the original position of the beam. The θ_e term is the rotation of the mid surface compared to this fictitious line, figure 2.4. In terms of Antman's parameters, these rotation can be written as.

$$\theta_r = d_1; \qquad \theta_e = \left(1 - 2\frac{s}{L}\right) d_2 + \left(1 - 6\frac{s}{L} + 6\frac{s^2}{L^2}\right) d_3 \tag{2.31}$$

The curvature θ' can be found by derivation of the expression for θ with respect to the arclength parameter.

$$\theta' = -\frac{2}{L} d_2 + \left(-\frac{6}{L} + \frac{12s}{L^2}\right) d_3 \tag{2.32}$$

All displacements, the rotation and their derivatives can be filled in the energy equation. When this is done, it will appear that there are some severe problems caused by membrane locking.

2.3.2 Membrane Locking

A very severe kind of problems in finite element development are locking problems. In principle, beam elements can suffer from two of these problems, shear locking and membrane locking. Since the current beam model have been derived using the Bernoulli hypothesis, the shear mode is neglected. Shear locking is therefore not possible. The other type, membrane locking can still occur. In order to explain this problem, a definition by Ibrahimbegović and Frey [12] is used.

The membrane locking phenomenon stems from the inability to capture a state of pure bending inextensional deformation. In an equation form, the beam

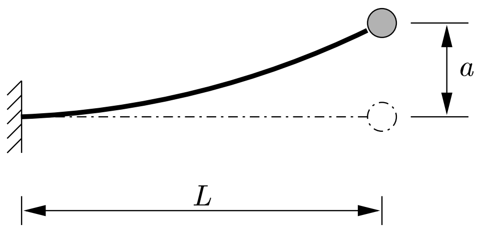


Figure 2.5: Curved 2 node beam element

is unable to bend while

$$\epsilon(s) = 0; \qquad \forall s \tag{2.33}$$

As opposed to the shear locking problem, which will be discussed in the next chapter, the membrane locking problem only occurs when the beam is already bent. In this case the angle that describes the rotation is no longer equal to 0. For the undeformed beam $\theta = 0$, the strain is equal to

$$\epsilon = u' \tag{2.34}$$

When the beam is slightly curved, the membrane strain can be written as

$$\epsilon = u' + \theta f' \tag{2.35}$$

Note that the term $\sin\theta$ is replaced by θ which is valid for small rotations. The geometry of the curved beam is denoted by the term f'. As can be seen in figure 2.5, this parameter can be written as

$$f = N_2(s)a; \qquad f' = \frac{df}{ds} = \frac{a}{L} \tag{2.36}$$

The discrete approximation of the axial strain for a curved beam becomes

$$\epsilon = \frac{1}{L}(u_2 - u_1) + \frac{a}{2L}(\theta_1 + \theta_2) + s\frac{a}{2L}(\theta_2 - \theta_1) \tag{2.37}$$

The only way to bend the beam while the axial strain remains 0 is when $\theta_1 = \theta_2$ is equal to zero. This means that the beam is not curved.

There are many methods to solve this problem. Most of them assume that the beam is integrated numerically, for example using a Gauss integration method. Since this beam will be integrated analytically, other techniques must be found. The rotation of the beam θ can said to be a linear extrapolation. By taking the average of the rotation, in principle the rotation of the beam in the mid point at $s = \frac{1}{2}L$ is calculated. Replacing the rotation terms by the average rotation θ_{av} gives the following equation

$$\hat{\epsilon} = \frac{1}{L}(u_2 - u_1) + \frac{a\theta_{\text{av}}}{L} \tag{2.38}$$

It can be seen that it is now possible to bend the beam, while the axial strain remains equal to 0.

The average of an arbitrary function $f(s)$ in a certain domain $s \in [0, L]$ can be calculated as follows

$$f_{\text{av}} = \frac{1}{L}\int_0^L f(s)ds \tag{2.39}$$

In the energy equations, θ always appears as a trigonometric function, i.e. $\cos\theta$ or $\sin\theta$. The averages of these functions are

$$\cos\theta_{\text{av}} = \frac{1}{L}\int_0^L \cos\theta ds \tag{2.40}$$

$$\sin\theta_{\text{av}} = \frac{1}{L}\int_0^L \sin\theta ds \tag{2.41}$$

Remember that θ is constructed from a rigid component θ_r and an elastic component θ_e.

In order to reduce the amount of calculations, a first simplification must be made. It can be assumed that the elastic rotation of the beam remains considerable small, $\theta_e < 0.1\ rad$. The trigonometric functions of the elastic rotations can then be represented as a Taylor expansion series.

$$\begin{aligned} \cos(\theta_e) &= 1 - \frac{1}{2}\theta_e^2 + H.O.T. \\ \sin(\theta_e) &= \theta_e + H.O.T. \end{aligned} \tag{2.42}$$

With the use of some manipulation functions, the average of the total rotation $\theta = \theta_e + \theta_r$ can be written as

$$\frac{1}{L}\int_0^L \cos\theta ds = \frac{1}{L}\int_0^L [\cos\theta_r - \theta_e \sin\theta_r - \frac{1}{2}\theta_e^2 \cos\theta_r]ds \tag{2.43}$$

$$\frac{1}{L}\int_0^L \cos\theta ds = \frac{1}{L}\int_0^L [\sin\theta_r + \theta_e \cos\theta_r - \frac{1}{2}\theta_e^2 \sin\theta_r]ds \tag{2.44}$$

Substituting of equations (2.31) and integrating the result yields:

$$\frac{1}{L}\int_0^L \cos\theta ds = \cos d_1[1 - \frac{1}{6}d_2^2 - \frac{1}{10}d_3^2] \tag{2.45}$$

$$\frac{1}{L}\int_0^L \cos\theta ds = \sin d_1[1 - \frac{1}{6}d_2^2 - \frac{1}{10}d_3^2] \tag{2.46}$$

These average rotation terms can be inserted in the strain energy equations, which can be integrated afterwards. Due to the simplicity of the formulations, this can be done by hand, or with the help of an external mathematical manipulation program such as `MAPLE V2`. The integrated strain energies are

$$
\begin{aligned}
\Psi =& \frac{EAL}{2}\left[1+d_4-\cos d_1\left(1-\frac{1}{6}d_2^2-\frac{1}{10}d_3^2\right)\right]^2+\\
&\frac{EAL}{2}\left[(1+d_4)\tan d_1-\sin d_1\left(1-\frac{1}{6}d_2^2-\frac{1}{10}d_3^2\right)\right]^2+\\
&\frac{2EI}{L}(d_2^2+3d_3^2)
\end{aligned}
\tag{2.47}
$$

and

$$
\begin{aligned}
\Psi_{\mathrm{T}} =& \frac{EAL}{2}\left[1+d_4-\cos d_1\left(1-\frac{1}{6}d_2^2-\frac{1}{10}d_3^2\right)\right]^2+\\
&\frac{EAL}{2}\left[(1+d_4)\tan d_1-\sin d_1\left(1-\frac{1}{6}d_2^2-\frac{1}{10}d_3^2\right)\right]^2+\\
&\frac{2EI}{L}\left[\frac{\cos^2 d_1(d_2^2+3d_3^2)(1-\frac{1}{6}d_2^2-\frac{1}{10}d_3^2)^2}{(1+d_4)^2}\right]
\end{aligned}
\tag{2.48}
$$

2.3.3 First and Second Variation

This expression can be used to obtain a relation between internal forces and displacements. Suppose that the strain energy of a structure is presented as a function of a single deformation u. It has been proven by Castigliano that the corresponding force F, needed for this deformation can be calculated by differentiating the strain energy equation with respect to u.

$$F=\frac{\partial\Psi(u)}{\partial u}\tag{2.49}$$

The stiffness k of the structure with respect to this deformation is equal to the derivative of the force F with respect to u

$$k=\frac{\partial F}{\partial u}\tag{2.50}$$

The stiffness can be deduced from the strain energy equation directly by combining equations (2.49) and (2.50), yielding

$$k=\frac{\partial^2\Psi}{\partial u\partial u}\tag{2.51}$$

As a result of the finite element discretization, the internal energy of the beam is expressed in terms of 4 variables, Antman's alternative parameters, which are just a function of the 6 nodal displacements and rotations. It is therefore possible to use Castigliano's theorems to determine the internal forces vector and the stiffness matrix.

For reasons of surveyability, the 6 nodal displacements and rotations u_I, w_I and θ_I are replaced by the vector $\mathbf{q} = [q_1, q_2, \dots, q_6]^t$. According to Castigliano's theorem, the internal forces vector can be calculated as follows.

$$f_i^{\mathrm{int,e}} = \frac{\partial \Psi}{\partial q_i} \tag{2.52}$$

Since the energy is expressed in terms of the alternative Antman parameters d_i, the chain rule must be used.

$$f_i^{\mathrm{int,e}} = \frac{\partial \Psi}{\partial q_i} = \frac{\partial \Psi}{\partial d_j}\frac{\partial d_j}{\partial q_i} \tag{2.53}$$

Introducing a 4×6 transformation matrix $\hat{\mathbf{A}}$, which is defined as

$$\hat{A}_{ij} = \frac{\partial d_i}{\partial q_j} \tag{2.54}$$

When the first variation of the strain energy in terms of the Antman parameters $\frac{\partial \Psi}{\partial d_i}$ is written as f_i^{a}, the correct first variation vector is equal to

$$f^{\mathrm{int,e}} = \hat{\mathbf{A}} f^{\mathrm{a}} \tag{2.55}$$

The stiffness matrix of the element ($\mathbf{K}^{\mathrm{e}}$) is the second variation of the strain energy. It can be calculated by differentiating the expression for the first variation with respect to $\mathbf{q}$

$$\mathbf{K}^{\mathrm{e}} = \frac{\partial}{\partial q_j}\left(\hat{\mathbf{A}}\mathbf{f}^{\mathrm{a}}\right) \tag{2.56}$$

This can be written as

$$K_{ij}^{\mathrm{e}} = \frac{\partial \hat{\mathbf{A}}}{\partial q_j} f_i^{\mathrm{a}} + \hat{A}_{ij}\frac{\partial f_i^{\mathrm{a}}}{\partial q_j} \tag{2.57}$$

When the chain rule is applied, this equation can be written as

$$\mathbf{K}^{\mathrm{e}} = \hat{\mathbf{A}}^t \mathbf{K}^{\mathrm{a}} \hat{\mathbf{A}} + f_1^{\mathrm{a}} \hat{\mathbf{A}}^{(1)} + f_2^{\mathrm{a}} \hat{\mathbf{A}}^{(2)} + f_3^{\mathrm{a}} \hat{\mathbf{A}}^{(3)} + f_4^{\mathrm{a}} \hat{\mathbf{A}}^{(4)} \tag{2.58}$$

Where $\mathbf{K}^{\mathrm{a}}$ is the stiffness matrix in terms of Antman's parameters

$$K_{ij}^{\mathrm{a}} = \frac{\partial \Psi}{\partial d_i \partial d_j} \tag{2.59}$$

and $\hat{\mathbf{A}}^{(k)}$ a transformation matrix.

$$\hat{A}_{ij}^{(k)} = \frac{\partial^2 f^{\mathrm{a}}}{\partial q_i \partial q_j} \tag{2.60}$$

Since both $\hat{\mathbf{A}}^{(2)}$ and $\hat{\mathbf{A}}^{(4)}$ are equal to 0 the complete equation for the 6×6 second variation matrix yields

$$\mathbf{K}^{\mathrm{e}} = \hat{\mathbf{A}}^t \mathbf{K}^{\mathrm{a}} \hat{\mathbf{A}} + f_1^{\mathrm{a}} \hat{\mathbf{A}}^{(1)} + f_3^{\mathrm{a}} \hat{\mathbf{A}}^{(3)} \tag{2.61}$$

The differentiations and the construction of the transformation matrices are rather straightforward. They can be done by hand, or with help of the mathematics manipulation program `MAPLE V2`.

2.4 Numerical Aspects

The `B2000` package is designed to analyze 3-dimensional structures. The implementation of the 2-dimensional beam elements in this environment will cause some problems. In principle, it is possible to use these elements in ordinary 3-dimensional structures. With a correct choice of the transformation matrix the local coordinate system of the 2-dimensional element can be placed arbitrarily in the 3-dimensional space. From a geometrical point of view, this solution is correct. However, the out-of-plane bending as well as the torsion of the beam are not defined. In these directions the beam suffers from zero energy modes (modes of which the stiffness of the beam is equal to zero). Locking of these modes is the only way to prevent the stiffness matrix of the structure from becoming singular.

The `B2000` package has already taken into account these kind of problems. There is an option to choose the type of geometry. Apart from the ordinary 3-dimensional coordinate system, a 2-dimensional (plane) and a quasi 2-dimensional space can be chosen as well. Unfortunately, these 2-dimensional spaces are not implemented yet. This implies that at the moment, the element cannot be used properly, unless the 2-dimensional space is implemented first. Since these beam elements are just designed for scientific use, it is better to think of a less radical solution.

The best idea is to use the element in a restricted ordinary 3-dimensional space. The elements can only be placed in the xy plane. The z-coordinate is always equal to zero. The displacements in the Z-direction as well as rotations about the x and y-axis need to be locked. In this case it is still possible to use the element in combination with 'normal' 3-dimensional elements, as long as these out-of-plane d.o.f.'s are locked for all nodes. A disadvantage of this solution is that it is easy to make mistakes.

From a programmers point of view, the internal forces vector and the stiffness matrix are considered as 3-dimensional objects. The out-of-plane d.o.f. rows and columns exist, but just contain zero values. This implies that the length of the forces vector and stiffness matrix is 12 elements long. The user will not notice this.

2.4.1 Transformation Matrix

The last step towards a workable finite element is the transformation of variables from the element local coordinate system into the global coordinate system[3].

$$\mathbf{T} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{2.62}$$

where α is the angle between the element local and the global coordinate system. The terms $\cos\alpha$ and $\sin\alpha$ can be calculated directly as can be seen in figure 2.6.

$$\cos\alpha = \frac{L_x}{L}; \qquad \sin\alpha = \frac{L_y}{L} \tag{2.63}$$

[3]In `B2000` conventions the global coordinate system is often referred to as the *branch global* coordinate system.

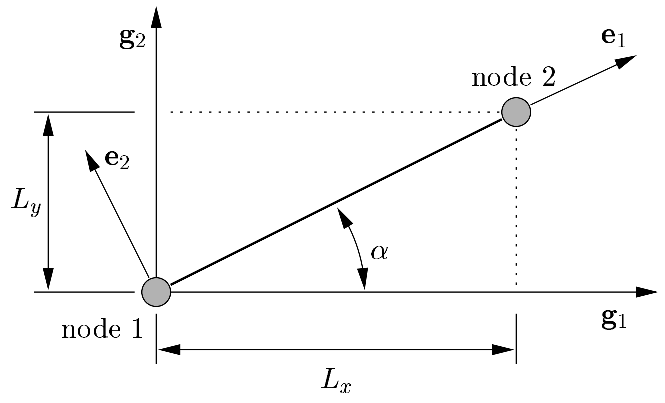


Figure 2.6: Transformation from the element local to the global coordinate system

In every analysis within the B2000 package, the displacements are calculated and stored in terms of the branch global coordinate system. In the calculations of the element local internal forces and stiffness matrices, these displacements must be transformed into element local displacements first, using the transformation matrix [1].

$$\mathbf{u}_e = \mathbf{T}^t \mathbf{u}_G \tag{2.64}$$

The transformation of the element local internal forces and stiffness matrices must be transformed to the branch global coordinate system using equivalent procedures.

$$\mathbf{f}_G^{\text{int}} = \mathbf{f}_e^{\text{int}} \mathbf{T} \qquad \mathbf{K}_G = \mathbf{T}^t \mathbf{K}_e \mathbf{T} \tag{2.65}$$

2.4.2 Implementation

As described above, the beams are implemented in the B2000 platform as ordinary 3-dimensional elements. Since both beams are almost identical, they have the same name, B2.EP. The model curvature description can be chosen using the flag NG. When NG is set to 1, the simplified version of the element is used. When NG is set to 2, the Timoshenko variant is chosen. In the sequel this particular variant will be denoted as B2.EP+. The users manual of the element can be found in appendix A. An overview of the source code is given in appendix C.

3

A 3-dimensional beam element

The 2-dimensional beam element developed in the previous section can be expanded to a 3-dimensional element by adding out-of-plane deformations. This will lead to a number of new difficulties, most of them concerning the kinematic description of the large rotations. The addition of a completely new deformation mode, the torsion, will also be considered.

Three dimensional beam elements in general are part of the so-called standard structural elements, which are essential to any finite element package. Other members are the shell elements and the volume elements. In principle any structure can be modeled with these elements. Apart from the typical beam like structures (trusses and frames), beam elements can also be used to model stiffeners and stringers on shells.

The aim is to develop a 3-dimensional beam element that, together with Rebel's shell elements [23] completes the family of nonlinear finite rotation elements in the B2000 package. Furthermore the beam must satisfy the demands stated in chapter 2, i.e. a good behavior in the post buckling area. The development is therefore completely based on proceedings by Simo *et al.* [29, 30].

3.1 Analytical Derivation

The mechanical properties of the beam are first derived analytically. The beam is considered as a one dimensional element in a 3-dimensional space. First a uniform way to describe the deformed position of the beam is regarded. Using this kinematic description, the strains and curvature of the beam can be determined. The *constitutive relations* translates the strains to stress. Finally, the derivation is completed with the formulation of the internal forces of the beam, using the linear and moment balance equations.

3.1.1 Kinematic Description

The kinematic description of the 3-dimensional beam is adapted from the description in the previous section and extended with an additional 'out-of-plane-

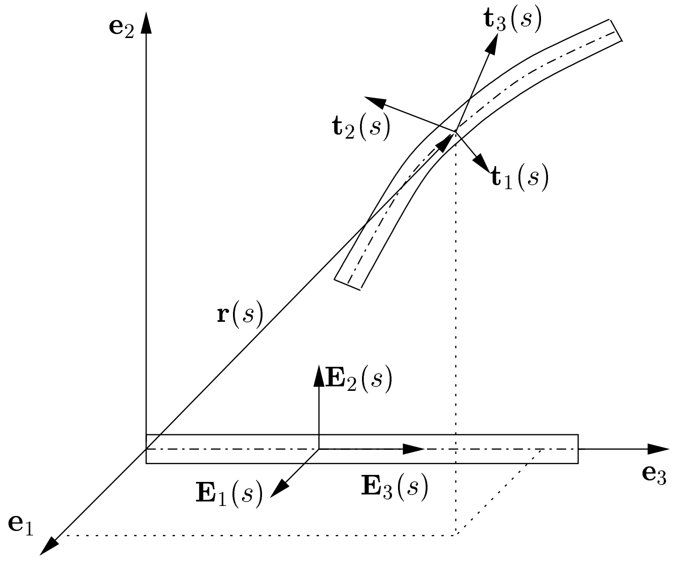


Figure 3.1: Kinematic description of the 3-dimensional beam

axis'. Let $\mathbf{t}_i(s,t)$[1] represent the three orthonormal basis vectors of a moving frame attached to a typical cross section, where $s \in [0, L] \subset \mathbb{R}$ denotes the curvilinear coordinate along the mid axis of the beam, and $t \in \mathbb{R}^+$ is a time parameter. The vector $\mathbf{t}_3(s)$ remains normal to the cross section at all times (in the derivation of the 2-dimensional element this vector was called $\mathbf{t}_1$). The fixed reference basis of the same section is denoted by $\mathbf{E}_i(s)$, so that at time $t = 0$ the following must hold: $\mathbf{t}_i(s, 0) = \mathbf{E}_i(s)$, for $s \in [0, L]$. Since the undeformed beam is straight, the orientation of fixed basis $\mathbf{E}_i$ is constant along the beam's arclength s and equal to the orientation of the element local coordinate system, $\mathbf{e}_i$.

The orientation of the moving frame can be expressed in terms of the fixed frame using the orthogonal transformation tensor $\mathbf{\Lambda}(s,t) = \Lambda_{ij}(s,t)\mathbf{E}_i \otimes \mathbf{E}_j$ which is a function of both the position on the beam axis coordinate s and the time t.

$$\mathbf{t}_i(s,t) = \mathbf{\Lambda}(s,t)\mathbf{E}_i = \Lambda_{ij}(s,t)\mathbf{E}_i \tag{3.1}$$

The position of the centroid of the cross section (i.e., the origin of the moving frame) is denoted with the vector $\mathbf{r} \in \mathbb{R}$ which is defined in the terms of the fixed frame as follows:

$$\mathbf{r}(s,t) = r_i(s,t)\mathbf{E}_i \tag{3.2}$$

Note that the position of a point on the nodal axis is expressed in terms of the fixed reference frame ($\mathbf{E}_i$). Accordingly, the set $C_{3\text{D}}$ of all possible configurations of the beam is defined by

$$C_{3\text{D}} = \{\boldsymbol{\phi} = (\mathbf{r}, \mathbf{\Lambda}) | \mathbf{r} : (0, L) \to \mathbb{R}^3, \mathbf{\Lambda} : (0, L) \to \text{SO}(3)\} \tag{3.3}$$

[1]In the sequel all latin subscripts can obtain the value $i = 1, 2, 3$. Furthermore, the Einstein convention holds, unless indicated otherwise.

Here, SO(3) is the special orthogonal (Lie) group. The transformation tensor $\mathbf{\Lambda}$ is a pure rotation tensor and defined in the SO(3) space of orthogonal transformations. This implies that the identity $\mathbf{\Lambda} \cdot \mathbf{\Lambda}^t = \mathbf{I}$ must hold. In other words, in a matrix notation, the transposed matrix is equal to the inverse matrix ($\mathbf{\Lambda}^t = \mathbf{\Lambda}^{-1}$).

In the continuing, all variables are expressed in one of these two reference frames. In principle, the complete derivation is done according to the fixed frame, the so-called *material description*. However, some variables need to be calculated in terms of the moving frame, the *spatial description*. Vectors and matrices can always be transformed from one to another frame using the identity (3.1).

3.1.2 Strain and Curvature

The strain is measured in the moving frame first. On the analogy of the Eriksson-Pacoste beam, the 3-dimensional strain vector γ is defined as follows:

$$\boldsymbol{\gamma} = \frac{d\mathbf{r}}{ds} - \mathbf{t}_3 \tag{3.4}$$

Where $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \epsilon]^t$. The first two terms denote shear strain in the $\mathbf{t}_1$ and $\mathbf{t}_2$ direction , the last term is the normal or axial strain. The strain vector can be transformed to the material reference by using equation (3.1):

$$\mathbf{\Gamma} = \mathbf{\Lambda}^{-1}\left(\frac{d\mathbf{r}}{ds} - \mathbf{t}_3\right) \tag{3.5}$$

Using the orthogonality condition of $\mathbf{\Lambda}$, this equation can be rewritten as

$$\mathbf{\Gamma} = \mathbf{\Lambda}^t \frac{d\mathbf{r}}{ds} - \mathbf{E}_3 \tag{3.6}$$

Due to the multiplication of the spatial strain vector by the rotation tensor $\mathbf{\Lambda}$ the individual terms of the material strain tensor $\mathbf{\Gamma}$ consist of both shear and axial strain terms.

The curvature κ in the 2-dimensional beam model was initially defined as the derivative of the rotation θ of the beam's cross section with respect to the arclength s, $\kappa = \frac{d\theta}{ds}$. This definition can also be used for the derivation of the curvature in the 3-dimensional beam. In terms of the spatial base $\mathbf{t}_i(s)$ and the rotation $\mathbf{\Lambda}(s)$ the curvature can be written as

$$\mathbf{\Omega} = \frac{d\mathbf{t}_i(s)}{ds} = \frac{d\mathbf{\Lambda}}{ds}\mathbf{e}_i(s) = \left[\frac{d\mathbf{\Lambda}(s)}{ds}\right]\mathbf{\Lambda}^t(s) \tag{3.7}$$

The spatial curvature tensor $\mathbf{\Omega}(s)$ is a so-called skew symmetric tensor of the following form

$$\mathbf{\Omega}(s) = \begin{bmatrix} 0 & -\omega_3(s) & \omega_2(s) \\ \omega_3(s) & 0 & -\omega_1(s) \\ -\omega_2(s) & \omega_1(s) & 0 \end{bmatrix} \tag{3.8}$$

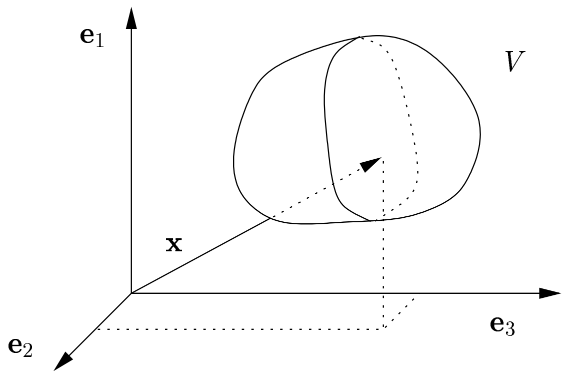


Figure 3.2: Position vector $\mathbf{x}$ in an arbitrary body V

It can be seen that the identity $\boldsymbol{\Omega} + \boldsymbol{\Omega}^t = \mathbf{0}$ must hold. Sometimes it is more convenient to use the associated axial vector of the tensor, denoted by the small letter $\boldsymbol{\omega}$. This vector is defined so that

$$\boldsymbol{\Omega}\mathbf{a} = \boldsymbol{\omega} \times \mathbf{a} \tag{3.9}$$

where $\mathbf{a}$ is an arbitrary vector. The axial vector consists of $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^t$. The first two terms are the curvature of the beam around the $\mathbf{t}_1$ and the $\mathbf{t}_2$ direction respectively. The third term ω_3 is the torsional curvature.

The translation of the spatial curvature to the fixed frame requires the same procedure as used above.

$$\boldsymbol{\kappa}(s) = \boldsymbol{\Lambda}^t(s)\boldsymbol{\omega}(s) \tag{3.10}$$

Here, $\boldsymbol{\kappa}(s)$ is the axial vector of the skew-symmetric tensor $\mathbf{K}(s)$.

3.1.3 Balance Equations

The element local equilibriums of forces and accelerations are described in the momentum balance equations, the balance of linear momentum and the balance of rotational momentum (or moment of momentum). Both equations will be derived stepwise in a general form first towards the explicit form for a 3-dimensional beam. The equations will be derived according to a spatial reference, i.e. the fixed frame. The linear momentum of an arbitrary body B is defined as

$$\iiint\limits_V \rho(\mathbf{x};t)\dot{\mathbf{x}}(\mathbf{x};t)dV \tag{3.11}$$

where $\dot{\mathbf{x}}(\mathbf{x};t)$ represents the velocity of an arbitrary particle with respect to a fixed point with position vector $(\mathbf{x};t)$ and where $\rho(\mathbf{x};t)$ is the material density. The rotational momentum of the body with respect to an arbitrary point $\mathbf{x}_0$ (in this case the origin of the fixed frame is chosen), is defined as

$$\iiint\limits_V \rho(\mathbf{x};t)\, \mathbf{x}\times\dot{\mathbf{x}}(\mathbf{x};t)dV \tag{3.12}$$

Apart from the momentum, there are external forces that work on the body. They can be divided into body forces and contact forces, acting on the surface of the body (∂V). The applied body forces and moments can be described as a function of the body force density $\mathbf{b}(\mathbf{x};t)$:

$$\iiint\limits_V \rho(\mathbf{x};t)\mathbf{b}(\mathbf{x};t)dV \tag{3.13}$$

The rotational balance in this case yields:

$$\iiint\limits_V \rho(\mathbf{x};t)\,\mathbf{x}\times\mathbf{b}(\mathbf{x};t)dV \tag{3.14}$$

A well-known example of a body force is the gravity force. In that case, the vector field $\mathbf{b}$ can be assumed to be constant over the body V. When the gravitational acceleration g acts in the negative z-direction (assuming a Cartesian coordinate system), $\mathbf{b}$ can be described as $\mathbf{b} = [0, 0, -g]^t$. A contact force does not affect each point of the body: it only effects the boundary ∂V. The linear and rotational parts are

$$\iint\limits_{\partial V} \mathbf{t}(\mathbf{x},\mathbf{n};t)dA \tag{3.15}$$

and

$$\iint\limits_{\partial V} \mathbf{x}\times\mathbf{t}(\mathbf{x},\mathbf{n};t)dA \tag{3.16}$$

The stress vector $\mathbf{t}$ is not only a function of place and time $(\mathbf{x};t)$, it also depends on the direction of the surface of the body $(\mathbf{n})$. The momentum equations together with the applied forces and moments result in the balance of momentum equations

$$\begin{aligned}\iiint\limits_V \rho(\mathbf{x};t)\mathbf{b}(\mathbf{x};t)dV + \iint\limits_{\partial V} \mathbf{t}(\mathbf{x},\mathbf{n},t)dA =\\ =\frac{d}{dt}\iiint\limits_V \rho(\mathbf{x};t)\dot{\mathbf{x}}(\mathbf{x};t)dV\end{aligned} \tag{3.17}$$

and

$$\begin{aligned}\iiint\limits_V \rho(\mathbf{x};t)\,\mathbf{x}\times\mathbf{b}(\mathbf{x};t)dV + \iint\limits_{\partial V} \mathbf{x}\times\mathbf{t}(\mathbf{x},\mathbf{n};t)dA =\\ =\frac{d}{dt}\iiint\limits_V \rho(\mathbf{x};t)\,\mathbf{x}\times\dot{\mathbf{x}}(\mathbf{x};t)dV\end{aligned} \tag{3.18}$$

The Cauchy stress formulation will be used to describe the surface forces. The stress vector $\mathbf{t}$ acts on the surface of the body in a point $\mathbf{x} \in \partial V$ in the direction $\mathbf{n}$. Cauchy proposed a new stress tensor which is independent of the normal $\mathbf{n}$.

$$\mathbf{t}(\mathbf{x},\mathbf{n}) = \mathbf{T}(\mathbf{x})\mathbf{n} \tag{3.19}$$

The tensor $\mathbf{T}$ is called the Cauchy stress tensor. In a rectangular Cartesian coordinate system, it can be decomposed as follows

$$\mathbf{T} = T_{ij}\mathbf{e}_i\mathbf{e}_j \tag{3.20}$$

and

$$\mathbf{t}_i = T_{ij}(\mathbf{x})\mathbf{n}_j \tag{3.21}$$

The first component ($\mathbf{e}_i$) defines the direction of the force, the second one ($\mathbf{e}_j$)) the direction of the normal. It can be shown by combining the linear and rotation momentum balances, that the Cauchy stress tensor is symmetric, $\mathbf{T} = \mathbf{T}^t$. Substituting the Cauchy stress tensor into equations (3.17) and (3.18) and using the divergence theorem, these equations can be written as

$$\iiint\limits_V \rho(\mathbf{x};t)\mathbf{b}(\mathbf{x};t) + \mathrm{div}[\mathbf{T}(\mathbf{x};t)]dV = \frac{d}{dt}\iiint\limits_V \rho(\mathbf{x};t)\dot{\mathbf{x}}(\mathbf{x};t)dV \tag{3.22}$$

and

$$\iiint\limits_V \rho(\mathbf{x};t)\,\mathbf{x}\times\mathbf{b}(\mathbf{x};t) + \mathrm{div}[(\mathbf{x}\times\mathbf{T}(\mathbf{x};t)]dV = \frac{d}{dt}\iiint\limits_V \mathbf{x}\times\dot{\mathbf{x}}(\mathbf{x};t)dV \tag{3.23}$$

The beam element may be considered as a one dimensional element in a 3 dimensional space. Beam properties can therefore be assumed to be constant over the beam cross-section. The divergence

$$\mathrm{div}[\mathbf{T}] = \begin{bmatrix} \frac{\partial}{\partial e_i}T_{i1} \\ \frac{\partial}{\partial e_i}T_{i2} \\ \frac{\partial}{\partial e_i}T_{i3} \end{bmatrix} \qquad \text{Summation over } i = 1,2,3 \tag{3.24}$$

can be written as (note that e_3 will be considered as the arclength variable s)

$$\hat{\mathbf{t}}_{,s} = \frac{\partial}{\partial s}\begin{bmatrix} T_{31} \\ T_{32} \\ T_{33} \end{bmatrix} \tag{3.25}$$

For the same reasons, the integration space V can be replaced by Ads, where A is the beam cross section. The balance equations can be written as

$$\int\limits_L \left[\frac{\partial \mathbf{n}}{\partial s} + \bar{\mathbf{n}}\right] ds = \int\limits_L \rho(s;t)A\ddot{\mathbf{r}}ds \tag{3.26}$$

$$\int\limits_L \left[\frac{\partial \mathbf{m}}{\partial s} + \frac{\partial \mathbf{r}}{\partial s} \times \mathbf{n} + \bar{\mathbf{m}}\right] ds = \int\limits_L \left[\rho\mathbf{I}\dot{\mathbf{w}} + \mathbf{w} \times (\rho\mathbf{I}\mathbf{w})\right] ds \tag{3.27}$$

where $\mathbf{n}$ and $\mathbf{m}$ are the internal forces and moments, the product of the internal stresses and the beam cross section. In the right hand sides of this equation is $\mathbf{I}$ the mass inertia tensor and $\mathbf{w}$ is a spin tensor, defined by Simo and VuQuoc [30].

In this chapter, a closer look is taken at the internal forces of the beam. The external (body) forces are of less importance. The dynamic (inertial) part of these equations (the right-hand-sides) are omitted for the time being. The same is done to the body forces that describe pre-stress, thermal expansion etc. In later developments these terms can be evaluated and added to the finite element definition of the beam.

$$\frac{\partial \mathbf{n}}{\partial s} = \mathbf{0} \tag{3.28}$$

$$\frac{\partial \mathbf{m}}{\partial s} + \frac{\partial \mathbf{r}}{\partial s} \times \mathbf{n} = \mathbf{0} \tag{3.29}$$

3.1.4 Constitutive Equations

The relation between strains and stresses are embedded in the *constitutive equations*, or the so-called stress strain equations. The beam considered so far is isotropic and fully elastic. As a result of this, there are no coupling effects. For example, a pure shear deformation has a pure shear force as a result. The constitutive relations are regarded in the moving frame first. They can be presented in the form of a matrix equation.

$$\begin{bmatrix} \mathbf{N} \\ \mathbf{M} \end{bmatrix} = \bar{\mathbf{C}} \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{\kappa} \end{bmatrix} \tag{3.30}$$

where $\bar{\mathbf{C}}$ is the 6×6 constitutive matrix and $\mathbf{N}$ and $\mathbf{M}$ are the forces and moments in material notation. Since there is no coupling, the matrix $\bar{\mathbf{C}}$ is a diagonal matrix. The distribution of stresses caused by axial strain, bending and torsion are linear over the beam cross section. The stresses as a result of the shear strain are distributed nonlinearly over the cross section. The distribution depends on the shape of the cross section as well.

Up to now, the complete derivation can be used for beams with an arbitrary cross section. However, in order to describe the constitutive relations of shear deformation, the shape of the cross section determines the stress distribution. In this case, it is convenient to consider the most simple and common cross section, a rectangular cross section. A method to determine the exact shear strain relation for this shape can be found in Bathe, [1]. In this book, the constitutive relation is presented as the ordinary linear relation GA (shear modulus and area) with a correction factor k.

The shear strain energy Ψ_{shear} per unit length of the beam is defined as follows.

$$\Psi_{\text{shear}} = \int_A \frac{1}{2G} \tau_a^2 dA = \int_{A_s} \frac{1}{2G} \left(\frac{V}{A_s} \right) dA_s \tag{3.31}$$

where τ_a is the actual shear stress, V the total shearing force and A_s the relative shear cross section area. Using $k = A/A_s$, the equation can be written as

$$k = \frac{V^2}{A \int_A \tau_a^2 dA} \tag{3.32}$$

For a rectangular cross section, the shear stress can be written as

$$\tau_a = \frac{3V}{2A}\left[\frac{(h/2)^2 - y^2}{(h/2)^2}\right] \tag{3.33}$$

giving the correction factor $k = 5/6$. Using the correction factor, the complete constitutive relations for the rectangular beam in material description become

$$\bar{\mathbf{C}} = \text{diag}\left[\frac{5}{6}GA, \frac{5}{6}GA, EA, EI_{xx}, EI_{yy}, GJ\right] \tag{3.34}$$

This material form of the constitutive matrix can be transformed to a spatial base, according to

$$\bar{\mathbf{c}} = \mathbf{\Pi}\bar{\mathbf{C}}\mathbf{\Pi}^t \tag{3.35}$$

where $\mathbf{\Pi}$ is a 6×6 transformation matrix assembled using the 'ordinary' transformation matrix $\mathbf{\Lambda}$.

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{\Lambda} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda} \end{bmatrix} \tag{3.36}$$

The same transformation can be executed for the forces and moments

$$\begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix} = \mathbf{\Pi} \begin{bmatrix} \mathbf{N} \\ \mathbf{M} \end{bmatrix} \tag{3.37}$$

where $\mathbf{n}$ and $\mathbf{m}$ are the forces and moments in spatial description, which can be substituted in the balance equations, (3.28) and (3.29).

3.2 Finite rotations

In a 3-dimensional configuration, the rotation space is a nonlinear manifold: the rotation of a vector is a nonlinear operation and as a result of this subsequent rotations cannot be added in a normal fashion. One of the manifestations of these nonlinearities is shown in figure (3.3). The final orientation of the box is determined by the order in which the three consecutive rotations are executed.

In many applications, for example older finite element descriptions, the rotations are assumed to remain small (< 0.1 *rad.*). As a result of this the three-dimensional rotations can then be expressed with a linear rotation tensor and the current reference from (the coordinate system) need not be updated. The beam element that will be derived in this chapter must be able to handle large deformations. The linear rotation tensor cannot be used then. Furthermore the reference frame must updated constantly.

In this section the rotation tensor for large rotations will be outlined according to the conventional description for small rotations. A special attention is paid to the Rodrigues rotation vector [3], which will be used in this specific case.

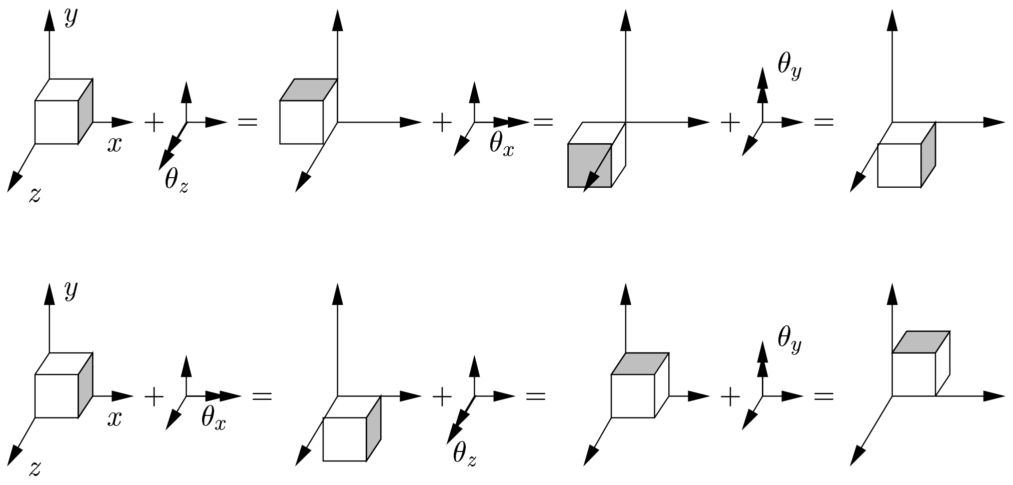


Figure 3.3: The non-communitativity of 3-dimensional vector rotations

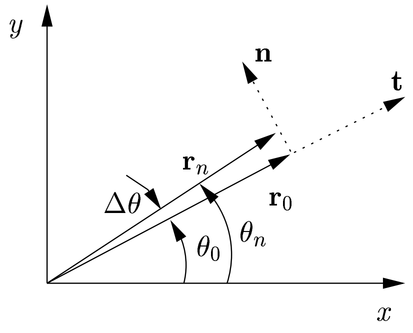


Figure 3.4: Small rotation in a 2-dimensional space

3.2.1 Small Rotations

Consider an arbitrary vector $\mathbf{r}_0$ in the 2-dimensional space as shown in figure (3.4) is rotated by an angle $\Delta\theta$ to become $\mathbf{r}_1$. The new vector $\mathbf{r}_1$ can be written as

$$\mathbf{r}_1 = ||\mathbf{r}_0|| \begin{bmatrix} \cos(\theta + \Delta\theta) \\ \sin(\theta + \Delta\theta) \\ 0 \end{bmatrix} \tag{3.38}$$

where $||\mathbf{r}_0||$ is the length of the vector $\mathbf{r}_0$. Since the rotations remain small, this equation can be simplified to:

$$\mathbf{r}_1 = \mathbf{r}_0 + \Delta\mathbf{r} = ||\mathbf{r}_0||(\mathbf{t} + \Delta\theta\mathbf{n}) \tag{3.39}$$

where $\mathbf{t}$ is a unit vector tangent to $\mathbf{r}_0$, $\mathbf{n}$ is a unit normal vector. In this situation, $\mathbf{n}$ can be written in the following form

$$\mathbf{n} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \tag{3.40}$$

Note that the relation $\mathbf{n} \cdot \mathbf{t}$ is satisfied. In a tensor notation, equation (3.38) can be written as

$$\mathbf{r}_1 = \begin{bmatrix} 1 & -\Delta\theta & 0 \\ \Delta\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r}_0 \tag{3.41}$$

This equation can be rewritten with a rotation tensor $\mathbf{\Lambda}_{\text{lin.}}$

$$\mathbf{r}_1 = \mathbf{\Lambda}\mathbf{r}_0; \qquad \mathbf{\Lambda}_{\text{lin.}} = \mathbf{I} + \mathbf{\Theta}_{\text{2D}} \tag{3.42}$$

where $\mathbf{I}$ is the unit tensor, $\mathbf{I} = \text{diag}[1, 1, 1]$, and $\mathbf{\Theta}$ is the skew-symmetric spin-tensor;

$$\mathbf{\Theta}_{\text{2D}} = \begin{bmatrix} 0 & -\Delta\theta & 0 \\ \Delta\theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.43}$$

It need no proof that when this equation is extended for an arbitrary rotation in a 3-dimensional space, this equation can be written as:

$$\mathbf{r}_1 = \mathbf{r}_0 + \Delta\mathbf{r} = \mathbf{r}_0 + (\Delta\boldsymbol{\theta} \times \mathbf{r}_0) \tag{3.44}$$

where $\Delta\boldsymbol{\theta}$ is the 3-dimensional rotation tensor. The spin-tensor $\mathbf{\Theta} = \Delta\boldsymbol{\theta}$ can be written in tensor notation.

$$\mathbf{\Theta} = \begin{bmatrix} 0 & -\Delta\theta_3 & \Delta\theta_2 \\ \Delta\theta_3 & 0 & -\Delta\theta_1 \\ -\Delta\theta_2 & \Delta\theta_1 & 0 \end{bmatrix} \tag{3.45}$$

Because of the simplifications made in this derivation, this linear rotation vector cannot be used for large rotations. The equivalent of equation (3.44) will be derived using Rodrigues formulation for large rotations.

3.2.2 Large Rotations

Assume that the rotation of an arbitrary vector $\mathbf{r}_0$ to a new situation $\mathbf{r}_1$ involves the vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^t$. This rotation is still an incremental rotation. For reasons of convenience the Δ term has been omitted. This vector can be decomposed into a unit vector $\mathbf{e}$.

$$\boldsymbol{\theta} = ||\boldsymbol{\theta}||\mathbf{e} \tag{3.46}$$

where θ is the length of the pseudo vector. Assume that the vector $\mathbf{r}_0$ is rotated around vector $\mathbf{e}$ over an angle $||\boldsymbol{\theta}||$ to the new direction $\mathbf{r}_1$, figure (3.5a). After examining the *rotation disc* as shown in figure (3.5b) the following identity can be seen:

$$\Delta\mathbf{r} = \Delta\mathbf{a} + \Delta\mathbf{b} \tag{3.47}$$

where $\Delta\mathbf{b}$ is orthogonal to $\Delta\mathbf{a}$. The length of vector $\Delta\mathbf{b}$ ($||\Delta\mathbf{b}||$) can also be derived form the figure;

$$||\Delta\mathbf{b}|| = R \sin ||\boldsymbol{\theta}|| \tag{3.48}$$

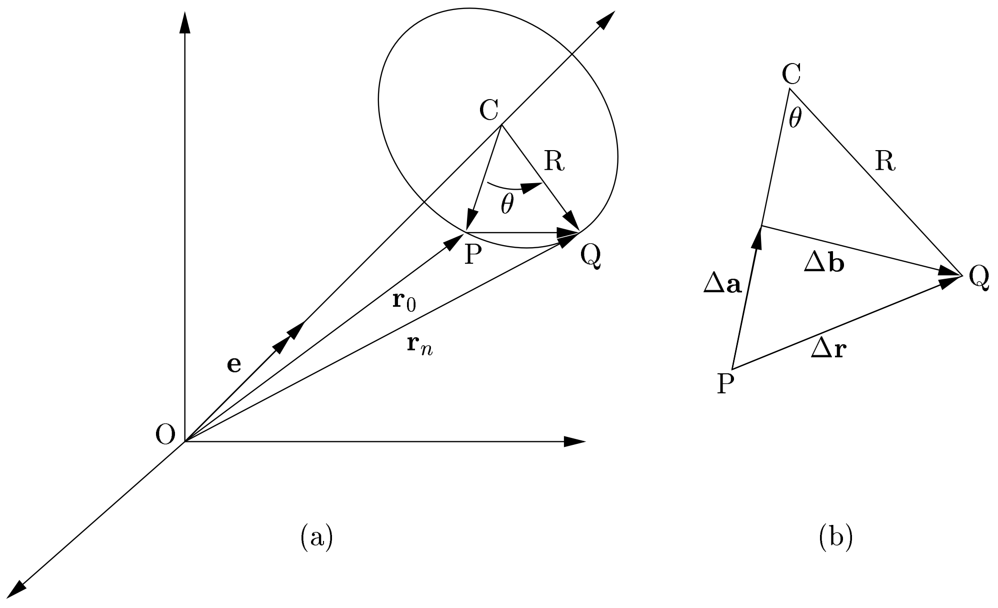


Figure 3.5: Three dimensional rotation (a) Detail (b)

The same can be done for $\Delta\mathbf{a}$

$$||\Delta\mathbf{a}|| = R(1 - \cos||\boldsymbol{\theta}||) \tag{3.49}$$

Vector $\Delta\mathbf{b}$ is perpendicular to both $\mathbf{r}_0$ and $\mathbf{e}$. Its direction can be calculated by the expression

$$\mathbf{b}^* = \mathbf{e} \times \mathbf{r}_0 \tag{3.50}$$

Note that the length of $\mathbf{b}^*$ is not equal to the length of $\mathbf{b}$. Dividing equation (3.50) by its own length $||\mathbf{b}^*||$ and multiplying this with equation (3.48) gives

$$\Delta\mathbf{b} = \frac{||\Delta\mathbf{b}||}{||\mathbf{r}_0 \times \mathbf{e}||}(\mathbf{e} \times \mathbf{r}_0) = \frac{R\sin||\boldsymbol{\theta}||}{||\mathbf{r}_0 \times \mathbf{e}||}(\mathbf{e} \times \mathbf{r}_0) \tag{3.51}$$

Using the following relation

$$||\mathbf{e} \times \mathbf{r}_0|| = ||\mathbf{r}_0||\ ||\mathbf{e}||\sin\alpha \tag{3.52}$$

Since $||\mathbf{e}|| = 1$ this can be written as

$$||\mathbf{e} \times \mathbf{r}_0|| = ||\mathbf{r}_0||\sin\alpha = R \tag{3.53}$$

Substituting this into equation (3.51) gives

$$\Delta\mathbf{b} = \sin||\boldsymbol{\theta}||(\mathbf{r}_0 \times e) = \frac{\sin||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}(\boldsymbol{\theta} \times \mathbf{r}_0) \tag{3.54}$$

Next, $\Delta\mathbf{a}$ can be determined since this vector is orthogonal to both $\Delta\mathbf{b}$ and $\mathbf{e}$.

$$\Delta\mathbf{a}^* = \sin||\boldsymbol{\theta}||(\mathbf{e} \times (\mathbf{e} \times \mathbf{r_0})) \tag{3.55}$$

In a similar way as to $\Delta\mathbf{b}$, vector $\Delta\mathbf{a}$ can be determined using equation (3.49)

$$\Delta\mathbf{a} = \frac{(1-\cos||\boldsymbol{\theta}||)}{||\boldsymbol{\theta}||^2}(\boldsymbol{\theta}\times(\boldsymbol{\theta}\times\mathbf{r}_0)) \tag{3.56}$$

The equations for $\Delta\mathbf{a}$ and $\Delta\mathbf{b}$ can now be substituted into equation (3.47).

$$\mathbf{r}_n = \mathbf{r}_0 + \Delta\mathbf{r} = \mathbf{r}_0 + \frac{\sin||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}(\boldsymbol{\theta}\times\mathbf{r}_0) + \frac{1-\cos||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||^2}(\boldsymbol{\theta}\times(\boldsymbol{\theta}\times\mathbf{r}_0)) \tag{3.57}$$

The definition of the spin-tensor

$$\boldsymbol{\theta}\times\mathbf{r}_0 = \boldsymbol{\Theta}\mathbf{r}_0 \tag{3.58}$$

holds so that the equation above can be written in a more general form.

$$\mathbf{r}_n = \boldsymbol{\Lambda}\mathbf{r}_0 \tag{3.59}$$

where $\boldsymbol{\Lambda}$ is the yet defined rotation tensor.

$$\boldsymbol{\Lambda} = \mathbf{I} + \frac{\sin||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\boldsymbol{\Theta} + \frac{(1-\cos||\boldsymbol{\theta}||)}{||\boldsymbol{\theta}||^2}\boldsymbol{\Theta}\boldsymbol{\Theta} \tag{3.60}$$

It is obvious that this expression is equal to the formula for the rotation tensor in for small rotations (3.42) with an additional nonlinear term.

3.2.3 Rodrigues Formula

For a number of reasons, the formula for finite rotations in a 3-dimensional space is often expressed in an alternative way. In this case an alternative expression based on Rodrigues' treatment is used. The incremental rotation $\boldsymbol{\theta}$ can be replaced by a so-called pseudo vector

$$\bar{\boldsymbol{\theta}} = \frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\boldsymbol{\theta} \tag{3.61}$$

Since the incremental rotation $\boldsymbol{\theta}$ can be decomposed in a direction vector $\mathbf{e}$ and its length, $\boldsymbol{\theta} = ||\boldsymbol{\theta}||\mathbf{e}$, the pseudo-vector can be written as

$$\bar{\boldsymbol{\theta}} = \tan\frac{1}{2}||\boldsymbol{\theta}||\mathbf{e} \tag{3.62}$$

After some manipulations, the rotation tensor $\boldsymbol{\Lambda}$ can be written as

$$\boldsymbol{\Lambda} = \mathbf{I} + \frac{2}{1+||\bar{\boldsymbol{\theta}}||}\left(\bar{\boldsymbol{\Theta}} + \bar{\boldsymbol{\Theta}}^2\right) \tag{3.63}$$

Note that all trigonometric terms have disappeared in this expression.

3.2.4 The Exponential Form

In equation (3.60) the rotation tensor is not just a function of the spin-tensor $\mathbf{\Theta}$, but also of the expressions $\sin\theta$ and $1-\cos\theta$. Using a Taylor expansion for these sine and cosine terms, it is possible to exclude them.

$$\sin||\boldsymbol{\theta}|| = ||\boldsymbol{\theta}|| - \frac{||\boldsymbol{\theta}||^3}{3!} + \frac{||\boldsymbol{\theta}||^5}{5!} - \dots \tag{3.64a}$$

$$\cos||\boldsymbol{\theta}|| = 1 - \frac{||\boldsymbol{\theta}||^2}{2!} + \frac{||\boldsymbol{\theta}||^4}{4!} - \dots \tag{3.64b}$$

Using the relationship

$$\mathbf{\Theta}^{2n-1} = (-1)^{n-1}||\boldsymbol{\theta}||^{2(n-1)}\mathbf{\Theta} \tag{3.65a}$$

$$\mathbf{\Theta}^{2n} = (-1)^{n-1}||\boldsymbol{\theta}||^{2(n-1)}\mathbf{\Theta}^2 \tag{3.65b}$$

the exponential form looks like

$$\mathbf{\Lambda} = \exp[\mathbf{\Theta}] = \mathbf{I} + \mathbf{\Theta} + \frac{\mathbf{\Theta}^2}{2!} + \frac{\mathbf{\Theta}^3}{3!} + \dots \tag{3.66}$$

The rotation tensor formulated in equation (3.63) can also be written in this form

$$\mathbf{\Lambda} = \exp[\mathbf{\Theta}] = \mathbf{I} + \frac{2}{1+||\bar{\boldsymbol{\theta}}||}\left(\bar{\mathbf{\Theta}} + \bar{\mathbf{\Theta}}^2\right). \tag{3.67}$$

This exponential notation can be very useful in the linearization process which will be tackled in section 3.3.2. However, it should be mentioned that this exponential formulation is just a symbolic notation. The rotation tensor *cannot* be calculated by using this notation, for the simple reason that the exponent of a tensor is not defined.

3.2.5 Compound Rotations

An already deformed beam element must be able to be deformed with a new, incremental rotation. Since the rotation is described in a nonlinear space, these compound rotations cannot be added in a traditional manner. The new incremental rotations must be applied to the current rotation tensor. In this section a formulation for compound rotations will be derivated. Consider a specific initial rotation $\boldsymbol{\theta}_0$

$$\mathbf{r}_1 = \mathbf{\Lambda}(\boldsymbol{\theta}_1)\mathbf{r}_0 \tag{3.68}$$

This rotation is followed by an incremental rotation $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^t$ that rotates the vector $\mathbf{r}_1$ into a new state ($\mathbf{r}_2$) following

$$\mathbf{r}_2 = \mathbf{\Lambda}(\boldsymbol{\theta}_2)\mathbf{r}_1 \tag{3.69}$$

Substituting equation (3.68) gives the following relationship between the new vector $\mathbf{r}_2$ and the reference vector $\mathbf{r}_0$.

$$\mathbf{r}_2 = \mathbf{\Lambda}(\boldsymbol{\theta}_2)\mathbf{\Lambda}(\boldsymbol{\theta}_1)\mathbf{r}_0 \tag{3.70}$$

The operator in the equation above can be replaced by a new operator $\mathbf{\Lambda}_{12}$.

Summarizing, the rotation of a vector $\mathbf{r}_0$ to $\mathbf{r}_2$ via the intermediate result $\mathbf{r}$ involves two rotation tensors. In all cases the rotation pseudo-vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ must be translated into a rotation tensor $\mathbf{\Lambda}_1 = \exp[\mathbf{\Theta}_1]$ and $\mathbf{\Lambda}_2 = \exp[\mathbf{\Theta}_2]$ respectively.

In the next chapter, the development of the transient processor, the concept of compound rotations is also used. In that case however a slightly different variant of the Rodrigues notation is used to describe the rotations. Doing so, the compound rotation vector can be fully derived as a function of the two rotation vectors, see section (4.6.3).

3.3 Internal Forces and Stiffness

The balance equations as derived in section 3.1.3 are used to determine the internal forces vector of the beam[2]. The stiffness of the beam can be found by taking the first derivative of the internal forces with respect to the displacements and rotations. In this case, since the expression of the internal forces is highly nonlinear, it will be done by linearization of the weak form of the balance equations. Before the balance equation can be linearized, an admissible variation must be superposed to the current deformed state.

3.3.1 Admissible Variations

A deformed configuration of the beam is specified by the position of the mid axis and the orientation of the moving frame, i.e.

$$\boldsymbol{\phi}(s) = (\mathbf{r}(s), \mathbf{\Lambda}(s)) \qquad \in \quad C_{\mathrm{3D}}. \tag{3.71}$$

Suppose that the configuration is perturbed relative to this configuration $\boldsymbol{\phi}(s)$ by a set $\boldsymbol{\eta}(s)$ consisting of a superposed infinitesimal displacement $\mathbf{u}(s)$ and a infinitesimal rotation $\boldsymbol{\theta}(s)$.

$$\boldsymbol{\eta}(s) = (\mathbf{u}(s), \boldsymbol{\theta}(s)) \tag{3.72}$$

The new, perturbed configuration is denoted by $\boldsymbol{\phi}_\varepsilon(s)$ and is still an element of the set C_{3D} of all possible configurations, equation (3.3).

$$\boldsymbol{\phi}_\varepsilon(s) = (\mathbf{r}_\varepsilon(s), \mathbf{\Lambda}_\varepsilon(s)) \qquad \in \quad C_{\mathrm{3D}} \tag{3.73}$$

Note that the additional incremental rotation $\boldsymbol{\theta}(s)$ is a compound rotation. Using equation (3.70) the new position and rotation of the mid axis can be expressed as

$$\mathbf{r}_\varepsilon(s) = \mathbf{r}(s) + \varepsilon\mathbf{u}(s); \qquad \mathbf{\Lambda}_\varepsilon(s) = \exp[\varepsilon\mathbf{\Theta}(s)]\mathbf{\Lambda}(s). \tag{3.74}$$

[2]In the derivation of the 2-dimensional beam, the internal forces are determined by derivation of the internal energy equation of the beam. The alternative name ‘first variation of the beam’ is used instead. The balance equation can be considered as the first variation of the energy equation as well.

3.3.2 Linearization

The linearization will be done stepwise. First, the configuration variables are linearized. After this the strain and curvature expressions will be tackled. The results can be used to linearize the weak form of balance equations. A *directional derivative* or so-called *Frechet derivative* is used to do so.

Configuration Variables

The set of configuration variables $\boldsymbol{\phi}(s) = (\mathbf{r}(s), \boldsymbol{\Lambda}(s))$ is linearized first. The perturbed configuration can be written as

$$\boldsymbol{\phi}_\varepsilon(s) = \mathbf{r}(s) + \varepsilon\mathbf{u} \tag{3.75}$$

Differentiation of this expression with respect to ε and setting $\varepsilon = 0$, the directional derivative of the vector $\mathbf{r}$ is determined.

$$\mathrm{D}\mathbf{r}\cdot\mathbf{u} = \left[\frac{d(\mathbf{r}+\varepsilon\mathbf{u})}{d\varepsilon}\right]_{\varepsilon=0} = \mathbf{u}(s) \tag{3.76}$$

which is not surprising since the position vector is a linear manifold. The (nonlinear) rotation of the configuration $\boldsymbol{\Lambda}$ is more complicated. Using the exponential form of the incremental rotation tensor $\exp[\boldsymbol{\Theta}]$

$$\mathrm{D}\boldsymbol{\Lambda}\cdot\boldsymbol{\Theta} = \left[\frac{d(\exp\varepsilon\boldsymbol{\Theta}\boldsymbol{\Lambda})}{d\varepsilon}\right]_{\varepsilon=0} = \left[\boldsymbol{\Theta}\exp[\varepsilon\boldsymbol{\Theta}]\boldsymbol{\Lambda}\right]_{\varepsilon=0} = \boldsymbol{\Theta}\boldsymbol{\Lambda} \tag{3.77}$$

The linearization of the transposed rotation tensor $\boldsymbol{\Lambda}^t$ can be found using an identical derivation

$$\mathrm{D}\boldsymbol{\Lambda}^t\cdot\boldsymbol{\Theta}^t = -\boldsymbol{\Theta}\boldsymbol{\Lambda}^t \tag{3.78}$$

Strain and Curvature

First the spatial description of the strain vector will be tackled. The strain of the perturbed situation is equal to

$$\boldsymbol{\Gamma}_\varepsilon = \boldsymbol{\Lambda}_\varepsilon^t\left(\frac{d\mathbf{r}_\varepsilon}{ds} - \mathbf{E}_3\right) \tag{3.79}$$

Using the linearized configuration variables, the linearized strain is equal to

$$\mathrm{D}\boldsymbol{\Gamma}\mathbf{u} = \Big[\boldsymbol{\Gamma}_\varepsilon]_{\varepsilon=0} = \boldsymbol{\Lambda}^t[\frac{d\mathbf{u}}{ds} - \boldsymbol{\Theta}\frac{d\mathbf{r}}{ds}\Big] \tag{3.80}$$

The linearized form of the spatial curvature tensor can be calculated in an identical fashion. Using equation (3.7), the curvature in the perturbed situation is equal to

$$\boldsymbol{\Omega}_\varepsilon = \frac{d\boldsymbol{\Lambda}_\varepsilon}{ds}d\boldsymbol{\Lambda}_\varepsilon = \left(\frac{d\exp[\varepsilon\boldsymbol{\Theta}]\boldsymbol{\Lambda}}{ds}\right)\exp[-\varepsilon\boldsymbol{\Theta}]\boldsymbol{\Lambda}^t \tag{3.81}$$

After derivation of the first term, this can be written as

$$\boldsymbol{\Omega}_\varepsilon = \left(\frac{d\exp[\varepsilon\boldsymbol{\Theta}]}{ds}\boldsymbol{\Lambda} + \exp[\varepsilon\boldsymbol{\Theta}]\frac{d\boldsymbol{\Lambda}}{ds}\right)\exp[-\varepsilon\boldsymbol{\Theta}]\boldsymbol{\Lambda}^t \tag{3.82}$$

Using the identity $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^t = \mathbf{I}$:

$$\boldsymbol{\Omega}_\varepsilon = \left(\frac{d\exp[\varepsilon\boldsymbol{\Theta}]}{ds}\right)\exp[\varepsilon\boldsymbol{\Theta}] + \exp[\varepsilon\boldsymbol{\Theta}]\boldsymbol{\Omega}\exp[-\varepsilon\boldsymbol{\Theta}] \tag{3.83}$$

This equation can be linearized completely except for the first term. It is shown by Simo and VuQuoc [30] that this term can be replaced by

$$\left(\frac{d\exp[\varepsilon\boldsymbol{\Theta}]}{ds}\right)\exp[\varepsilon\boldsymbol{\Theta}] = \frac{2}{1+||\bar{\boldsymbol{\theta}}||^2}\left(\bar{\boldsymbol{\Theta}}' + \bar{\boldsymbol{\Theta}}'\bar{\boldsymbol{\Theta}} - \bar{\boldsymbol{\Theta}}\bar{\boldsymbol{\Theta}}'\right) \tag{3.84}$$

When $\boldsymbol{\Theta}$ is replaced by $\varepsilon\boldsymbol{\Theta}$, the alternative spin tensor $\bar{\boldsymbol{\Theta}}$ must be replaced by $\frac{1}{2}\boldsymbol{\varepsilon}\bar{\boldsymbol{\Theta}}$. The tensor $\bar{\boldsymbol{\Theta}}'$ is the skew-symmetric tensor of the axial vector $\bar{\boldsymbol{\theta}}'$. This vector can be calculated by differentiating equation (3.61).

$$\bar{\boldsymbol{\theta}}' = \left[\frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\right]'\boldsymbol{\theta} + \frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\boldsymbol{\theta}' \tag{3.85}$$

Note that the derivative of $||\boldsymbol{\theta}||' = \mathbf{e}\cdot\boldsymbol{\theta}'$,

$$\bar{\boldsymbol{\theta}}' = \left[(1+\tan^2\frac{1}{2}||\boldsymbol{\theta}||)(\mathbf{e}\cdot\boldsymbol{\theta}') - \frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}(\mathbf{e}\cdot\boldsymbol{\theta}')\right]\mathbf{e} + \frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\boldsymbol{\theta}' \tag{3.86}$$

Manipulation of the quadratic tangent term gives

$$\bar{\boldsymbol{\theta}}' = \left[\frac{1}{\cos^2\frac{1}{2}||\boldsymbol{\theta}||} - \frac{2\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\right](\mathbf{e}\cdot\boldsymbol{\theta}')\mathbf{e} + \frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{||\boldsymbol{\theta}||}\boldsymbol{\theta}' \tag{3.87}$$

which can finally be rewritten as

$$\bar{\boldsymbol{\theta}}' = \frac{1}{2}\frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{\frac{1}{2}||\boldsymbol{\theta}||}\left[\boldsymbol{\theta}' - \left(1 - \frac{||\boldsymbol{\theta}||}{\sin||\boldsymbol{\theta}||}\right)(\mathbf{e}\cdot\boldsymbol{\theta}')\mathbf{e}\right] \tag{3.88}$$

Substituting equations (3.88) and (3.84) into (3.82) and taking the derivative with respect to ε, it follows that the linearized expression for the curvature is equal to

$$\mathrm{D}\boldsymbol{\Omega}\boldsymbol{\Theta} = [\boldsymbol{\Omega}_\varepsilon]_{\varepsilon=0} = \boldsymbol{\Theta} \tag{3.89}$$

3.3.3 Weak Form of Balance Equations

The balance equations (3.28) and (3.29) must hold for all possible deformations $\boldsymbol{\phi} = (\mathbf{r}, \boldsymbol{\Lambda})$. Any admissible variation with respect to this deformation must also hold, with the exception that they disappear on the boundary of the beam, i.e. at $s = 0$ and $s = L$.

The balance equations are multiplied by the admissible variations $\boldsymbol{\eta}(s) = (\mathbf{u}(s), \boldsymbol{\theta}(s))$, yielding

$$G(\boldsymbol{\phi}, \boldsymbol{\eta}) = \int\limits_L \left[\frac{d\mathbf{n}}{ds} \cdot \mathbf{u} + \left(\frac{d\mathbf{m}}{ds} + \frac{d\mathbf{r}}{ds} \times \mathbf{n} \right) \cdot \boldsymbol{\theta} \right] ds = 0 \tag{3.90}$$

Since the variation $\boldsymbol{\eta}(s)$ vanishes at the boundary, integration by parts of this equation leads to the spatial version of the weak form of the balance equations.

$$G(\boldsymbol{\phi}, \boldsymbol{\eta}) = \int\limits_L \left[\mathbf{n} \cdot \left(\frac{d\mathbf{u}}{ds} - \boldsymbol{\theta} \times \frac{d\mathbf{r}}{ds} \right) + \mathbf{m} \cdot \frac{d\boldsymbol{\theta}}{ds} \right] ds = 0 \tag{3.91}$$

The stiffness of the beam, previously referred to as the second variation can be obtained by linearization of the weak form of balance equations. This process is completely adapted from Simo and Vu-Quoc [30].

3.3.4 Linearization of the Weak Form

The stiffness of the structure, previously referred to as the second variation of the internal energy, can be calculated from the internal forces vector by differentiating this term with respect to the displacement and rotations. In this case, with the nonlinear description of rotations, it is done by linearizing the weak form of the balance equations. This procedure is equal to the linearization of the configuration variables and the strain and curvature measures. The complete derivation can be found in a paper by Simo and VuQuoc [30].

The linearization of the weak form of the balance equations can be presented in a Taylor series expansion notation.

$$\mathrm{L}[G(\boldsymbol{\phi}, \boldsymbol{\eta})] = G(\boldsymbol{\phi}, \boldsymbol{\eta}) + \mathrm{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]\Delta\boldsymbol{\eta} \tag{3.92}$$

The last term $\mathrm{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]\Delta\boldsymbol{\eta}$ is the Frechet derivative, which is equal to the stiffness of the beam. It consists of two parts. The first part contains the linearization of the internal forces vector $[\mathbf{n}, \mathbf{m}]^t$. Recall the spatial version of the equation that describe the internal forces,

$$\begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Lambda} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda} \end{bmatrix} \bar{\mathbf{C}} \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{\kappa} \end{bmatrix} \tag{3.93}$$

Linearization of this equation and substituting the result in the weak form of balance equation, gives

$$\mathrm{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]_1 \Delta\boldsymbol{\eta} = \int\limits_L \left[\boldsymbol{\Xi} \ \hat{\mathbf{c}} \ \boldsymbol{\Xi}^t \right] ds \tag{3.94}$$

where $\boldsymbol{\Xi}$ is a differential matrix, which can be written as

$$\boldsymbol{\Xi} = \begin{bmatrix} \frac{d}{ds}\mathbf{I} & \mathbf{0} \\ -\mathbf{r}' \times \mathbf{I} & \frac{d}{ds}\mathbf{I} \end{bmatrix} \tag{3.95}$$

Since the deformations are prescribed in a nonlinear manifold, the second part of the vector contains the linearization of the geometry variables. This part, often called the geometric stiffness matrix, can be written as

$$\mathrm{D}[G(\boldsymbol{\phi},\boldsymbol{\eta})]_2\Delta\boldsymbol{\eta} = \int\limits_L \left[\boldsymbol{\Psi}^t\mathbf{B}\boldsymbol{\Psi}\right] ds \tag{3.96}$$

where $\boldsymbol{\Psi}$ is a second differential operator defined as

$$\boldsymbol{\Psi} = \begin{bmatrix} \frac{d}{ds}\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{d}{ds}\mathbf{I} & \mathbf{I} \end{bmatrix} \tag{3.97}$$

and $\mathbf{B}$ the so-called geometric stiffness matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & [-\mathbf{n}\times\mathbf{I}] \\ \mathbf{0} & \mathbf{0} & [-\mathbf{m}\times\mathbf{I}] \\ [-\mathbf{n}\times\mathbf{I}] & \mathbf{0} & \mathbf{n}\otimes\mathbf{r}' - (\mathbf{n}\cdot\mathbf{r}')\mathbf{I} \end{bmatrix} \tag{3.98}$$

where $(\mathbf{n}\otimes\mathbf{r}')_{ij} = n_i\mathbf{r}_i$. This part of the tangent matrix is zero when the beam is undeformed.

It is shown by Simo and VuQuoc [30] that it is non-symmetric when the configuration $\boldsymbol{\phi}(s) = (\mathbf{r}, \boldsymbol{\Lambda})$ is not in an proper equilibrium. Methods to overcome these symmetry problems are discussed in the next section.

3.4 Numerical implementation

The analytical derivation in the previous sections can be used to define a finite element description. Just as in the development of the 2-dimensional beams, two components are required to obtain a workable nonlinear 3-dimensional finite element, i.e. a first variation vector (i.e. the internal forces vector) and a second variation matrix (the stiffness matrix).

The analytical description holds for arbitrary beam structures. It is therefore just a small step to apply it to a beam of finite length L. This can be either a two node or a three node element, often referred to as linear and quadratic elements respectively. Higher order elements can be deduced using the same techniques but are not very common. The discretization is first executed for an arbitrary finite element beam model with nel nodes. The specific implementation of the 2 node beam will be considered afterwards.

3.4.1 Discretization

A typical element is set up using nel nodes and has the initial length L. The nodal incremental displacements $\mathbf{u}_I$ and rotations $\boldsymbol{\theta}_I$ are interpolated in terms of shape functions (interpolation functions).

$$\mathbf{u}(s) = \sum_{I=1}^{nel} N_I(s)\mathbf{u}_I, \qquad \boldsymbol{\theta}(s) = \sum_{I=1}^{nel} N_I(s)\boldsymbol{\theta}_I \tag{3.99}$$

Here, nel represent the total number of nodes of the beam element, $N_I(s)$ the shape function associated with node I, and $\mathbf{u}_I, \boldsymbol{\theta}_I$ are the nodal incremental displacement and rotation of the element in node I.

Internal Forces Vector

The element contribution to the internal forces vector is obtained from the discrete approximation to the weak form of the balance equations. Proceeding in an element fashion, by introducing the interpolation functions, the discrete approximation to $G(\boldsymbol{\phi}, \boldsymbol{\eta})$ may be written as

$$G(\boldsymbol{\phi}, \boldsymbol{\eta}) = \sum_{e=1}^{E} G_e(\boldsymbol{\phi}, \boldsymbol{\eta}) \tag{3.100}$$

where E is the total number of elements. The element notation of the weak form of balance equation, $G_e(\boldsymbol{\phi}, \boldsymbol{\eta})$ can be rewritten as.

$$G_e(\boldsymbol{\phi}, \boldsymbol{\eta}) = \boldsymbol{\eta}_e \mathbf{f}_e^{\text{int}}(\boldsymbol{\phi}) = \sum_{I=1}^{nn} \boldsymbol{\eta} \cdot \mathbf{f}_{eI}^{\text{int}}(\boldsymbol{\phi}) \tag{3.101}$$

Here, $\mathbf{f}_{eI}^{\text{int}}(\boldsymbol{\phi})$ denotes the residual force vector in the I^{th} node of a typical element. It can be computed using the discrete approximations of the differential operator $\boldsymbol{\Xi}$. Let $\boldsymbol{\Xi}_I$ represent the discrete differential operator associated with node I.

$$\boldsymbol{\Xi}_I = \begin{bmatrix} N_I'\mathbf{I} & \mathbf{0} \\ -N_I[\mathbf{r}' \times \mathbf{I}] & N_I'\mathbf{I} \end{bmatrix} \tag{3.102}$$

In this expression, N_I' denotes the derivative of $N_I(s)$ with respect to s, $\mathbf{I} = \text{Diag}[1,1,1]$ is the unit matrix, and $[\mathbf{r}' \times \mathbf{I}]$ is a skew-symmetric matrix whose axial vector is $\mathbf{r}'$.

The spatial stress vector $[\mathbf{n}_e, \mathbf{m}_e]$ is computed from the constitutive equations (3.30). The unbalanced nodal force $\mathbf{f}_{eI}^{\text{int}}$ for a single beam element, related to node I can be written as

$$\mathbf{f}_{eI}^{\text{int}} = \int_L \boldsymbol{\Xi}_I \begin{bmatrix} \mathbf{n}_e \\ \mathbf{m}_e \end{bmatrix} ds \tag{3.103}$$

It can easily be seen that this integral equation is a system of 6 independent equations. One for each degree of freedom.

The Tangent Stiffness Matrix

In the previous section, it is shown that the stiffness matrix consists of two parts, an 'ordinary' part due to the linearization of the internal forces and moments and a geometric part due to the linearization of the rotation tensor $\boldsymbol{\Lambda}$. The same interpolation functions can be applied to these equations, yielding

$$\text{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]\Delta\boldsymbol{\eta} = \text{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]_1\Delta\boldsymbol{\eta} + \text{D}[G(\boldsymbol{\phi}, \boldsymbol{\eta})]_2\Delta\boldsymbol{\eta} = \mathbf{S}_{eIJ} + \mathbf{K}_{eIJ} \tag{3.104}$$

where the stiffness matrix $\mathbf{S}_{eIJ}$ is equal to

$$\mathbf{S}_{eIJ} = \int_L \boldsymbol{\Xi}_I \mathbf{c}_e \boldsymbol{\Xi}_J^t ds \tag{3.105}$$

and the geometric stiffness matrix $\mathbf{T}_{eIJ}$ is equal to

$$\mathbf{T}_{eIJ} = \int_L \mathbf{\Psi}_I \mathbf{B}_e \mathbf{\Psi}_J^t ds \tag{3.106}$$

where $\mathbf{\Psi}_I$ is the discrete expression of the second differentiation tensor $\mathbf{\Psi}$.

$$\mathbf{\Psi}_I = \begin{bmatrix} N_I'\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & N_I'\mathbf{I} & N_I\mathbf{I} \end{bmatrix} \tag{3.107}$$

The *total element stiffness matrix* $\mathbf{K}_{eIJ}$ is the sum of the ordinary element stiffness matrix and the element geometric stiffness matrix.

$$\mathbf{K}_{eIJ} = \mathbf{S}_{eIJ} + \mathbf{T}_{eIJ} \tag{3.108}$$

This matrix is non-symmetrical when the deformation is not an equilibrium state. Since B2000 requires symmetrical matrices, the stiffness must be divided into a symmetric and a skew-symmetric part.

$$\mathbf{K}_{eIJ}^{\text{SYM}} = \frac{1}{2}(\mathbf{K}_{eIJ} + \mathbf{K}_{eIJ}^t); \qquad \mathbf{K}_{eIJ}^{\text{SKEW}} = \frac{1}{2}(\mathbf{K}_{eIJ} - \mathbf{K}_{eIJ}^t) \tag{3.109}$$

The symmetric part is used as the temporary stiffness matrix, the skew-symmetric part is used for convergence checking: when all terms of this matrix are (almost) equal to zero, a equilibrium state (at least in terms of the beam geometry) has been found.

For the time being, just a 2 node beam element will be programmed. This means that both I and J can obtain the values 1 and 2. Hence, the internal forces vectors $\mathbf{f}_{eI}^{\text{int}}$ is a vector with 12 elements; $\mathbf{K}_{eIJ}$ is a 12×12 matrix. Quadratic (3 node) and higher order elements can be derived using the same procedures. In the next section, the interpolation functions of a two node beam element will be discussed.

3.4.2 Interpolation Functions

Since the beam itself is a one dimensional object (the only position parameter on the beam is the arclength parameter s) all variables are a function of s. However, in the finite element description, just the displacement and rotations in the nodes are known. Interpolation functions (or shape functions) are required to achieve the relationship between these nodal displacements and rotations and the displacements and rotations on the beam.

Standard shape functions have to fulfill just two rules. First of all the 'normalized' function should have the value 1 in the specific node and the value 0 in all the other nodes and second, the sum of all the shape functions must have the value 1 in the complete integration domain, in this case $s \in [0, L]$.

In order to be able to integrate the equations numerically, the domain is often transformed and normalized. Consider a new arc length variable, ξ with the domain $\xi = [-1, 1]$. The transformation equation is equal to

$$\xi = 2\left(\frac{s}{L} - \frac{1}{2}\right) \tag{3.110}$$

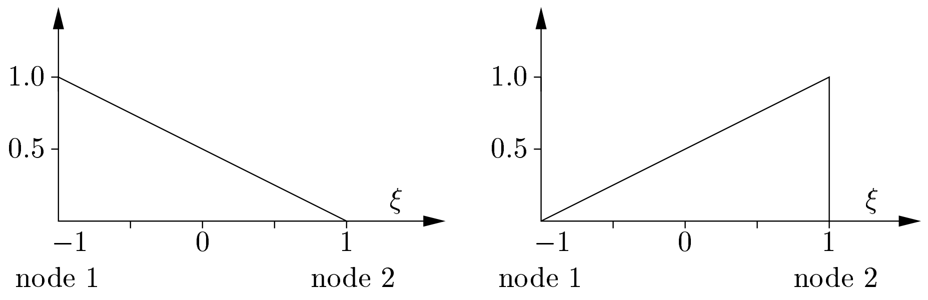


Figure 3.6: Interpolation functions for the 2 node beam element

The Jacobian of this transformation which will be needed in the integration process is defined as follows

$$J = \frac{ds}{d\xi} = \frac{1}{2}L \tag{3.111}$$

An arbitrary function set up by two points is called a linear function by definition. A 2 node beam element is therefore called a linear element. The term linear does not imply that the description of internal forces and stiffness is linear. Just the interpolation functions are linear. The 2 linear interpolation functions for the domain $[-1, 1]$ are shown in figure 3.6. In an equation form, the functions for node 1, N_1 and node 2, N_2 can be written as

$$N_1 = \frac{1}{2}(1-\xi); \qquad N_2 = \frac{1}{2}(1+\xi) \tag{3.112}$$

So the discrete equation for the displacements u_x reads

$$u_x = \frac{1}{2}(1-\xi)\,u_{x1} + \frac{1}{2}(1+\xi)\,u_{x2} \tag{3.113}$$

where u_{x1} denotes the discrete displacement in the x-direction of the first node. The differentiation of this expression with respect to s reads;

$$\frac{u_x}{ds} = \frac{u_x}{d\xi}\frac{d\xi}{s} = (-\frac{1}{2}u_{x1} + \frac{1}{2}u_{x2})J^{-1} = \frac{-1}{L}u_{x1} + \frac{1}{L}u_{x2} \tag{3.114}$$

So, the differentiated interpolation functions are

$$N_1' = -\frac{1}{L} \;\; ; \qquad N_2' = \frac{1}{L} \tag{3.115}$$

Eventually the interpolation of a three node beam can be obtained by following the same principles. The three functions (node 1, 2 and 3) should have the value 1 in the corresponding node and 0 in the other 2 nodes. The only way to achieve this is by using quadratic interpolation functions. A three node element is therefore often referred to as a quadratic element.

n	Sample point α_i	Weight r_i
1	0.00000 00000 00000	2.00000 00000 00000
2	-0.57735 02691 89626	1.00000 00000 00000
	0.57735 02691 89626	1.00000 00000 00000
3	-0.77459 66692 41483	0.55555 55555 55555
	0.00000 00000 00000	0.88888 88888 88888
	0.77459 66692 41483	0.55555 55555 55555
4	-0.86113 63115 94053	0.34785 48451 37454
	-0.33998 10435 84856	0.65214 51548 62546
	0.33998 10435 84856	0.65214 51548 62546
	0.86113 63115 94053	0.34785 48451 37454

Table 3.1: Sampling points and weight factors for Gauss quadrature integration for a one dimensional interval $\xi = [-1, 1]$

3.4.3 Numerical Integration

The equations for internal forces and stiffness are written in an integral form which have to be integrated over the interval $s \in [0, L]$. This can be done by hand, a laborious job which will result in enormous equations. A more distinguished manner is to integrate the equations numerically. A number of different techniques can be used to do so. These techniques are more or less based on the same idea. The function is evaluated at a number (k) of specific points, the so-called *sampling points*. The values of the function in these sampling points are used to set up a polynomial, by using *weight factors*. The primitive of this i^{th} order polynomial is known so that the polynomial can be integrated.

One of the most simple integration methods is the Newton-Côtes integration method. The integration domain is assumed to be divided into n intervals, the sampling points are spaced at equal distances. The Newton-Côtes procedure for 1 interval is also known as the trapezoidal integration rule, the procedure for 2 intervals as the Simpson formula. The performances of both methods are rather poor. Good results are obtained when the domain is divided into more intervals (> 4).

The previous integration schemes considered so far use equally spaced sampling points. The Gauss Quadrature integration method uses optimized sampling positions. The basic assumption of this method is that both the weight factors $\alpha_1, \ldots, \alpha_n$ and the sampling points $r_1, \ldots, r_n$ are variables. In literature [1], the weight factors and sampling points are presented for a one dimensional domain $[-1, 1]$[3]. They can be found in table 3.1.

The Gauss integration method is applied to the equations governing the internal forces and the stiffness of the beam. Every component of this 12×1 vector and 12×12 matrix respectively must be integrated apart. The integral

[3]This is the one and only reason that the domain of the integrals in the previous section is transformed from $[0, L]$ to $[-1, 1]$.

equation of such a component can be written in the following arbitrary form.

$$F = \int_{0}^{L} f(s) ds \tag{3.116}$$

First this equation is transformed to the new arc length variable ξ, according to

$$F = \int_{0}^{L} f(\xi) J d\xi \tag{3.117}$$

where J is the Jacobian of the transformation, derived in equation (3.111). In this case, since the interpolation functions vary linearly with respect to ξ, this function can be integrated using the 2-point Gauss integrating technique. The full equation can than be written as

$$F = \alpha_1 f(r_1) J + \alpha_2 f(r_2) J \tag{3.118}$$

The weight factors and sampling points in this case are respectively (table 3.1)

$$\begin{array}{rrrr} r_1 = & -0.57735\ 02691\ 89626; & \alpha_1 = & 1.0 \\ r_2 = & 0.57735\ 02691\ 89626; & \alpha_1 = & 1.0 \end{array} \tag{3.119}$$

Higher order integration methods will not result in significant more accurate results.

3.4.4 Updating the Configuration

The rotations of the nodes cannot be used directly in this finite element formulation: the update rotation tensor $\mathbf{\Theta}$ as well as the curvature $\boldsymbol{\kappa}$ must be calculated regarding the previous configuration.

Assume that the previous configuration in the point $n-1$ is known, i.e. $\boldsymbol{\phi}_{n-1}$ containing the position of the mid axis $\mathbf{r}$ and the rotation of the frame $\mathbf{\Lambda}_{n-1}$. The curvature tensor of the previous step, i.e. $\mathbf{\Omega}_{n-1}(s)$, is known too. The new configuration $\boldsymbol{\phi}_n$ can be considered as a variation of the old configuration $\mathbf{r}_{n-1}$ where $\mathbf{u}$ and $\boldsymbol{\theta}$ are the variables. This means that the following must hold

$$\mathbf{r}_n = \mathbf{r}_{n-1} + \mathbf{u}; \qquad \mathbf{\Lambda}_n = \exp[\mathbf{\Theta}]\mathbf{\Lambda}_{n-1} \tag{3.120}$$

In this perspective, the most attention is paid to the calculation of the exponential form of the rotation tensor. This has been done for arbitrary variations in section 3.3.1, but will be repeated here for the incremental rotation $\boldsymbol{\theta}$.

- The first aim is to determine the Rodrigues type rotation vector $\bar{\boldsymbol{\theta}}$ and its derivative. The normalized rotation vector $\mathbf{e}$ is needed for these calculations.

$$\mathbf{e} = \frac{\boldsymbol{\theta}}{||\boldsymbol{\theta}||} \tag{3.121}$$

$$\bar{\boldsymbol{\theta}}(s) = \tan\frac{1}{2}||\boldsymbol{\theta}||\mathbf{e} \tag{3.122}$$

$$\bar{\boldsymbol{\theta}}'(s) = \frac{1}{2}\frac{\tan\frac{1}{2}||\boldsymbol{\theta}||}{\frac{1}{2}||\boldsymbol{\theta}||}\left[\boldsymbol{\theta}' - \left(1 - \frac{||\boldsymbol{\theta}||}{\sin||\boldsymbol{\theta}||}\right)(\mathbf{e}\cdot\boldsymbol{\theta}')\mathbf{e}\right] \tag{3.123}$$

- The rotation tensor as well as its derivative can be calculated using the Rodrigues rotation terms

$$\exp[\boldsymbol{\Theta}] = \mathbf{I} + \frac{2}{1+||\bar{\boldsymbol{\theta}}||}\left(\bar{\boldsymbol{\Theta}} + \bar{\boldsymbol{\Theta}}^2\right) \tag{3.124}$$

$$\left(\frac{\mathrm{d}\exp[\boldsymbol{\Theta}]}{dS}\right)\exp[-\boldsymbol{\Theta}] = \frac{2}{1+||\bar{\boldsymbol{\theta}}||}\left(\bar{\boldsymbol{\Theta}} + \bar{\boldsymbol{\Theta}}'\bar{\boldsymbol{\Theta}} + \bar{\boldsymbol{\Theta}}\bar{\boldsymbol{\Theta}}'\right) \tag{3.125}$$

- The rotation tensor can be used to determine the new position of the moving frame $\boldsymbol{\Lambda}_{n+1}$

$$\boldsymbol{\Lambda}_{n+1} = \exp[\boldsymbol{\Theta}]\boldsymbol{\Lambda}_n \tag{3.126}$$

- The curvature in terms of the moving frame

$$\boldsymbol{\Omega}_{n+1} = \left(\frac{\mathrm{d}\exp[\boldsymbol{\Theta}]}{dS}\right)\exp[-\boldsymbol{\Theta}] + \exp[\boldsymbol{\Theta}]\boldsymbol{\Omega}_n\exp[-\boldsymbol{\Theta}] \tag{3.127}$$

- Finally the curvature and strain can be transformed to the fixed frame using the rotation tensor $\boldsymbol{\Lambda}$

$$\boldsymbol{\kappa}_{n+1} = \boldsymbol{\Lambda}_n^t\boldsymbol{\omega}_n \tag{3.128}$$

$$\boldsymbol{\Gamma}_{n+1} = \boldsymbol{\Lambda}_n^t\boldsymbol{\phi}'_{0(n)} - \mathbf{E_3} \tag{3.129}$$

3.4.5 Transformation Matrix

The beam element is defined in a local coordinate system $\mathbf{E}_i$. The variables can be transformed into branch global coordinates $\mathbf{g}_i$ in a similar way as described in section 2.4.1. Nevertheless, the construction of the transformation matrix $\mathbf{T}$ is somewhat more complicated compared to the 2-dimensional beam element, since an additional director $\mathbf{n}_x$ is involved to define the exact position of the element in the branch global space.

The position of the element is determined using 3 nodes, see figure 3.7[4]. Node 1 can be considered as the beam's origin, node 2 determines the endpoint of the local $\mathbf{E}_3$ vector and is on the other end-point of the beam. The auxiliary

[4]This method is also used positioning the linear beam element which is already implemented in B2000 [16].

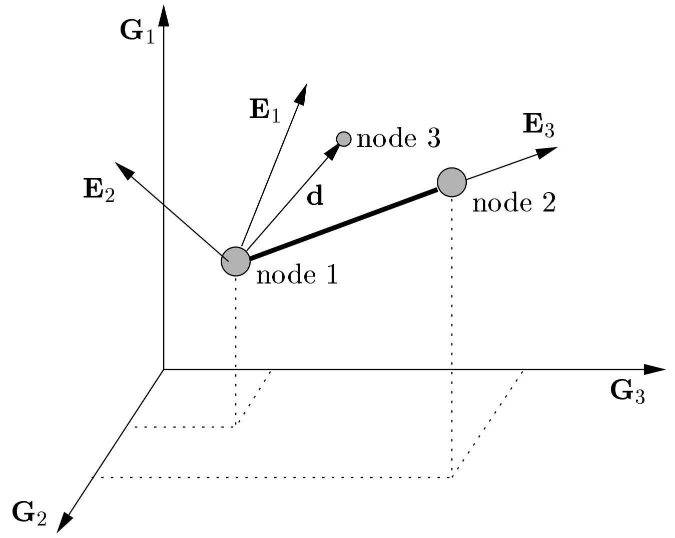


Figure 3.7: Orientation of the 3-dimensional beam element

third node is used to setup the element local E_{13}-plane. The vector $\mathbf{d}$ need not to be perpendicular to $\mathbf{E}_3$. The director in the local E_2 direction, $\mathbf{n}_2$, is normal to the E_{13}-plane and can be calculated by taking the vector product of the 2 vectors ($\mathbf{E}_3$ and $\mathbf{d}$) that are used to set up this plane.

$$\mathbf{E}_2 = \mathbf{E}_3 \times \mathbf{d} \tag{3.130}$$

$\mathbf{E}_1$ can be determined in a similar way,

$$\mathbf{E}_1 = \mathbf{E}_2 \times \mathbf{E}_3 \tag{3.131}$$

The transformation matrix is formulated as

$$\mathbf{T} = \begin{bmatrix} \cos(\mathbf{E}_1, \mathbf{g}_1) & \cos(\mathbf{E}_1, \mathbf{g}_2) & \cos(\mathbf{E}_1, \mathbf{g}_3) \\ \cos(\mathbf{E}_2, \mathbf{g}_1) & \cos(\mathbf{E}_2, \mathbf{g}_2) & \cos(\mathbf{E}_2, \mathbf{g}_3) \\ \cos(\mathbf{E}_3, \mathbf{g}_1) & \cos(\mathbf{E}_3, \mathbf{g}_2) & \cos(\mathbf{E}_3, \mathbf{g}_3) \end{bmatrix} \tag{3.132}$$

where $\cos(\mathbf{E}_i, \mathbf{g}_j)$ is the cosine of the angle between vector $\mathbf{E}_i$ and $\mathbf{g}_j$. This expression can be written as

$$\cos(\mathbf{E}_i, \mathbf{g}_j) = \frac{\mathbf{E}_i \cdot \mathbf{g}_j}{||\mathbf{E}_i||\ ||\mathbf{g}_j||} \tag{3.133}$$

When the vectors $\mathbf{E}_i$ are normalized and $\mathbf{g}_j$ are unit Cartesian base vectors, $\mathbf{E}_1 = [1, 0, 0]^t$, this equation will reduce to

$$\cos(\mathbf{E}_i, \mathbf{g}_j) = \mathbf{E}_i^{(j)} \tag{3.134}$$

where $\mathbf{E}_i^{(j)}$ is the jth component ($j = 1, 2, 3$) of the vector $\mathbf{E}_i$. The transformation matrix will be

$$\mathbf{T} = \begin{bmatrix} \mathbf{E}_1^{(1)} & \mathbf{E}_1^{(2)} & \mathbf{E}_1^{(3)} \\ \mathbf{E}_2^{(1)} & \mathbf{E}_2^{(2)} & \mathbf{E}_2^{(3)} \\ \mathbf{E}_3^{(1)} & \mathbf{E}_3^{(2)} & \mathbf{E}_3^{(3)} \end{bmatrix} \tag{3.135}$$

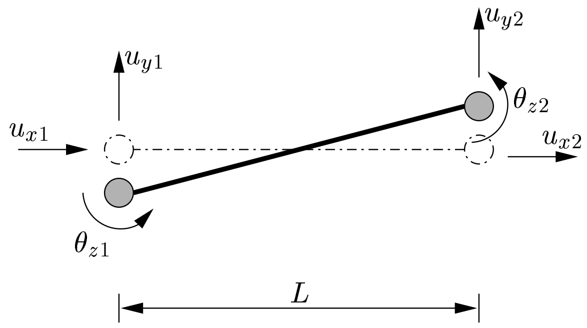


Figure 3.8: Beam deformation including shear effect

3.5 Locking Phenomena

In the previous chapter, the membrane locking phenomenon was already detected on the curved 2-dimensional beam elements. Since the Bernoulli hypothesis is not used in this 3-dimensional beam element, there is an additional locking mode, namely shear locking. In this section, the shear locking will be examined more closer and both the shear-locking as well as the membrane locking effects will be tackled on the 3-dimensional element.

3.5.1 Shear Locking

In order to get a better idea of shear deformation of a beam in general, the shear deformation of a plane beam element with shear effects will be considered. It can be assumed that the results also holds for fully 3-dimensional beams.

In figure 3.8, a beam under shear is presented. The general shear deformation of the beam γ can be written as

$$\gamma = \frac{dw}{ds} - \theta \tag{3.136}$$

For the 2 node beam element a set of linear interpolation functions have been used to establish a relationship between node displacements and continuous displacements. In this case the displacement w and the rotation θ can be written as

$$w(\xi) = N_1(\xi)w_1 + N_2(\xi)w_2 \qquad \theta(\xi) = N_1(\xi)\theta_1 + N_2(\xi)\theta_2 \tag{3.137}$$

Substituting these interpolation functions into equation (3.136), gives

$$\gamma(\xi) = N_1'(\xi)u_{y1} + N_2'(\xi)u_{y2} - N_1(\xi)\theta_{z1} - N_2(\xi)\theta_{z2} \tag{3.138}$$

And after substituting the linear interpolation functions, this equation can be written as

$$\gamma(\xi) = -\frac{1}{L}u_{y1} + \frac{1}{L}u_{y2} - \frac{1}{2}(1-\xi)\theta_{z1} - \frac{1}{2}(1+\xi)\theta_{z2} \tag{3.139}$$

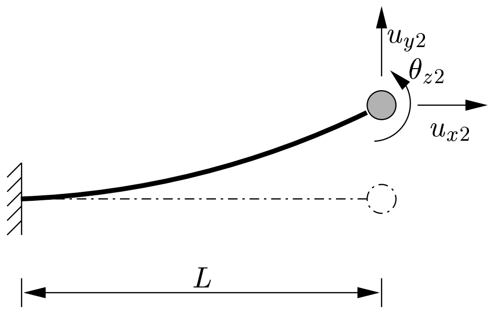


Figure 3.9: Pure bending of a clamped beam

or

$$\gamma = \frac{1}{L}(u_{y2} - u_{y1}) - \frac{1}{2}(\theta_{z1} + \theta_{z2}) + \frac{\xi}{2}(\theta_{z1} - \theta_{z2}) \tag{3.140}$$

The beam must be able to capture a pure bending state, also known as Kirchhoff's mode. One of the characteristics of a pure bending state is that the shear strain (or the shear stress) is equal to zero,

$$\gamma(\xi) = 0 \qquad \forall \quad \xi \in [-1, 1] \tag{3.141}$$

This equation can only be satisfied for all values of ξ when all terms in equation (3.140) are equal to 0. This also means that $\theta_{z1} = \theta_{z2} = 0$. Hence, a zero shear strain can only be reached when there is no deformation at all.

A number of remedies is known to overcome this problem. Just on of them will be considered in this case, the method of *reduced integration*. Other remedies can be found in a paper by Ibrahimbegović [12].

3.5.2 Reduced Integration

The reduced integration or selective integration method is the best known and the most effective remedy for locking problems. The idea is simple. Equation (3.140) can be satisfied in one point, which is $\xi = 0$. The equation reduces to

$$\gamma = \frac{1}{L}(u_{y2} - u_{y1}) - \frac{1}{2}(\theta_{z1} + \theta_{z2}) \tag{3.142}$$

This equation can be satisfied using nonzero θ_{z1} and θ_{z2}. Consider the case of a clamped beam, i.e. $u_{y1} = \theta_{z1} = 0$, as shown in figure 3.9 A pure bending state is satisfied when,

$$u_{y2} = \frac{1}{2}\theta_{z2}L \tag{3.143}$$

since there is no locking when the shear strain is evaluated in the mid point of the beam. In this point, the shear strain is equal to the average shear strain and can therefor be used for the complete beam. When the shear terms of the beam are integrated using a one-point (reduced) integration method, the shear is just evaluated at $\xi = 0$.

3.5.3 Membrane Locking

Just as the 2-dimensional beam element, this element suffers from membrane locking. The methods to overcome this problem are rather simple. As can be seen in section 2.3.2, the function for membrane strain in the 2-dimensional beam is

$$\epsilon = \frac{1}{L}(u_2 - u_1) + \frac{a}{2L}(\theta_1 + \theta_2) + s\frac{a}{2L}(\theta_2 - \theta_1) \tag{2.37}$$

Just as in the previous case, this problem can be overcome by applying the reduced integration technique on the membrane strain.

3.5.4 Implementation

The first and second variation matrices are expressed in the fixed frame. As a consequence of the transformation by the rotation matrix $\mathbf{\Lambda}$, the membrane and shear strain terms cannot be seen easily. They cannot be taken apart and integrated with the reduced method afterwards. Other ways must be found instead to apply the reduced integration method.

The membrane and shear strains are always multiplied by the stiffness parameters EA and kGA respectively. By setting these parameters equal to zero, the membrane and shear terms are omitted from the integral equation. Afterwards, when the bending en torsion stiffnesses are set equal to zero, the membrane and shear terms can be integrated with the one point Gauss integration. The complete internal forces vector and stiffness matrix, can be found by summing the results.

$$\mathbf{f}^{\mathrm{int,e}} = \mathbf{f}^{\mathrm{int,e}}_{2\ \mathrm{point}} + \mathbf{f}^{\mathrm{int,e}}_{1\ \mathrm{point}}; \qquad \mathbf{K}^{\mathrm{e}} = \mathbf{K}^{\mathrm{e}}_{2\ \mathrm{point}} + \mathbf{K}^{\mathrm{e}}_{1\ \mathrm{point}}; \tag{3.144}$$

3.6 Element Mass Matrix

In the derivation of the balance equations for a 3-dimensional beam, the inertia terms have been neglected. The beam was considered to be quasi-static. The most important reason for this decision is that the inertia terms will have a geometrically nonlinear mass matrix as a result. In the next chapter, when the equations of motion are solved using a time integration method, the mass matrix is assumed to be geometrically linear. In other words, the derivation of the mass will not be used at all in the future. A simplified mass description will be used instead.

The nonlinearity in the mass description just concerns the rotational inertia terms. When a linear description is used, all errors occur in the rotational terms. The 3-dimensional beam developed in this chapter is assumed to be slender. This means that the radius of gyration of the beam $r = \sqrt{I/A}$ is smaller than 10^{-4}. In other words, the moment of inertia is 100 times as small as the beams cross-section. The rotational mass which is proportional to the moment of inertia, is therefore much smaller than the displacement inertia.

Instead of allowing this small error in the mass matrix, the inertia terms can be omitted completely. The mass matrix will than reduce to a diagonal lumped matrix, with zero terms in the rotational terms[5]. In case of a 2 node beam element, the discretized mass can be written as

$$\mathbf{M}_I = \int_L \rho A \boldsymbol{\Upsilon} ds \tag{3.145}$$

where $\boldsymbol{\Upsilon}$ is the generalized lumped mass matrix

$$\boldsymbol{\Upsilon} = \begin{bmatrix} N_1\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.146}$$

This simple equation can be integrated analytically. For this specific two node beam element the reduced lumped mass matrix $\mathbf{M}_e$ can be written in the following form

$$\mathbf{M}_e = \mathrm{diag}[\frac{1}{2}m, \frac{1}{2}m, \frac{1}{2}m, 0, 0, 0, \frac{1}{2}m, \frac{1}{2}m, \frac{1}{2}m, 0, 0, 0] \tag{3.147}$$

where m is the total mass of the beam, $m = \rho AL$.

3.7 Closure

The beam that has been derived in this chapter is implemented in the B2000 platform as a two node beam element. Unfortunately, at the moment, this element is not working properly. An outline of the current state of the element is given in this section.

In linear cases, the performances of the beam are good. This implies that the ordinary stiffness matrix $\mathbf{S}$ is derived correctly for an undeformed beam. Furthermore, the reduced integration method in order to avoid the shear locking problem works decently.

In nonlinear cases the beam does not function right. When it is bent, the internal forces vector is wrong. The error increases with the curvature. Deformation modes where bending is not an issue, axial strain and torsion, do not have these problems. These problems can have two different causes. First, when the beam is curved, the material refence frame $\mathbf{e}_i$ is no longer equal to the spatial reference frame $\mathbf{t}_i$. As a result, there is a difference between the spatial description and the material description of the beam properties. Perhaps, these two descriptions are misinterpreted in the derivations. In linear cases, when these frames are equal, this problem is not relevant. Second, the membrane locking phenomena is not tackled correctly. Since it only occurs in curved beams, it is an indication that it may not be banned completely. The solution can be found in the application of different techniques or the implementation of new interpolation functions.

Since the quasi-static behavior of the element is rather poor, it is not tested in a nonlinear transient analysis at all. Testcases in which the static as well as the dynamic performances of the element are considered, can be found in chapter 6.

[5]This idea of developing a mass matrix is also used in STAGS [22].

4 The Implicit Time Integration Solver

The deformation of a structure due to external loads can be described by a number of mathematical models. The simplest one is the static linear model. This limited model is accurate when deformations of the structure remain small. Buckling behavior and stability analysis can be calculated using the nonlinear static equations of equilibrium. In both cases the dynamic behavior (velocity and acceleration) is neglected as well as the variation of the applied loads in the time domain. When these dynamic phenomena are included in the model, the resulting system of nonlinear equations is called the *equations of motion*[1]. This is the most complete and accurate model for the description of all mechanical behavior of structures.

All three models described above are implemented in the `B2000` platform. The static linear model in the `B2LIN` macroprocessor, the nonlinear model in `B2CONT` and the kinematic model in the `B2ETA` macroprocessor. As opposed to the *implicit* solvers `B2LIN` and `B2CONT`, the equations of motion in transient macro-processor `B2ETA` are solved using an *explicit* solution technique. Since this solution technique is based on a different finite element model, `B2ETA` cannot directly be used in combination with the implicit solvers `B2LIN` and `B2CONT`.

The *mode jumping* phenomenon (which will be discussed in Chapter 5) can be calculated using both a nonlinear static as well as a transient solver. To provide a mode-jump analysis technique in `B2000`, the need arose to implement an implicit transient. A first step was taken by K. Yildirim [34] in 1996. His solver was called `B2IDTI` and was able to solve linear responses of structures, using Park's time integration method [20, 21] and Jensen's transformation algorithm [13]. A well fixed combination that is used in many other finite element platforms, such as `STAGS` [22], currently under development at Lockheed Martin.

[1]The equations of motion are often referred to as dynamic equilibrium equations or kinematic equations. Although all terminologies are correct, in this report the first name will be used.

In this chapter the linear algorithm is evaluated as well as the development of the nonlinear solver. In the first sections a number of implicit time integration algorithms will be examined. On the basis of stability and accuracy criteria, the best algorithms will be chosen and implemented. Methods to solve the generated system of nonlinear equations are reviewed in section (4.5). Finally the solution strategies as well as the implementation of the macroprocessor will be described.

4.1 The Equations of Motion

The nonlinear equations of motion can be derived from Newton's second law and written in the following form:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}(\mathbf{u};t) = \mathbf{0} \tag{4.1}$$

where $\mathbf{M}$ and $\mathbf{C}$ are the discrete mass and damping matrices respectively; $\ddot{\mathbf{u}}$ and $\dot{\mathbf{u}}$ are the acceleration and velocity vectors. The matrices $\mathbf{M}$ and $\mathbf{C}$ are assumed to be symmetric and positive definite[2]. The vector $\mathbf{f}(\mathbf{u};t)$ is the total force vector. This vector can be divided into an internal force vector $\mathbf{f}^{\text{int}}(\mathbf{u})$, which depends on the displacements of the structure and an external, time dependent force vector $\mathbf{f}^{\text{ext}}(t)$, so that the equation can be written as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{f}^{\text{ext}}(t) \tag{4.2}$$

When the internal forces of the structure are described by a linear function, they can be rewritten as $\mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{K}\mathbf{u}$ where $\mathbf{K}$ is the linear stiffness matrix, that is constant for all displacements. In that case, the equations of motion reduce to

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}^{\text{ext}}(t) \tag{4.3}$$

This linearized formulation can only be used when displacements are assumed to remain very small.

4.1.1 Loading and Initial Conditions

The nonlinear equation of motion can be considered a second order non-homogeneous differential equation, ODE in short. A structural model governed by this equation, can be loaded in two different ways: by a prescribed force function or by prescribed displacements. A combination of both is possible as well.

Prescribed external forces:

The load is applied as discrete time dependent external forces or moments. In the ODE, these loads are present in the external load vector $\mathbf{f}^{\text{ext}}(t)$, the non-homogeneous right-hand-side of the dynamic equation. Likewise,

[2]When the mass and damping matrices are assumed to be nonlinear, i.e. they depend on the current displacement, this is not necessarily true. In some conditions, both the mass and damping matrices can become a-symmetric. However, in this report these matrices are considered linear.

body and surface forces such as gravitational forces and traction have to be translated into discrete nodal forces before they can be added to the external forces vector.

Prescribed displacements:
The structure can also be deformed by prescribing certain displacements. In real life such a deformation can be found in a laboratory testing bench, where for instance the buckling loads of specimens are tested by controlling its end-shortening. The prescribed displacements have internal forces as a result. Since the velocities and accelerations are the first and second time derivatives of the displacements also damping forces and inertia forces occur in the transient process.

A system of second order differential equations cannot be solved without initial conditions $\mathbf{u}_0$, $\dot{\mathbf{u}}_0$. In many cases the structure is undeformed and at rest, the initial conditions $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$ are equal to $\mathbf{0}$. Initial conditions that are not equal to zero can be the results of a previous calculation, which can either be a static analysis with `B2LIN` or `B2CONT`[3] or a previous transient analysis. In that case, the transient analysis can be restarted at a starting time t_0 with the initial conditions $\mathbf{u}(t_0)$ and $\dot{\mathbf{u}}(t_0)$.

4.1.2 Stiff Equation

The equation of motion is often called a stiff ODE due to the existence of greatly differing eigenfrequencies. In a second order differential equation time constant is the term used for eigenfrequencies of the construction. The number of eigenfrequencies of a discrete structure is of the same order as the number of degrees of freedom, varying from the lowest 'base' frequency up to the highest 'overtone' with a period that is just a small fraction of the base period. The importance of these overtones in the response of the structure decreases with increasing frequencies. It can be said that for a global dynamic analysis just the first frequencies have a dominant influence on the response of the structure.

4.1.3 Rayleigh's Damping Coefficient

Since transient analysis is a relatively new field in (computational) mechanics, over the years, little attention was paid to the development of a proper description of the damping. Correctly formulated damping matrices are a rarity, which is not surprising, since the damping phenomenon is a complicated mixture of many physical subdivions, for example irreversible thermodynamic processes or plasticity. Many older finite element models, does not have a description of a damping term at all. In order to have a workable tool to model damping, Rayleigh has developed a simplified formulation. The damping is considered to be a summation of the element inertia (mass) and stiffness matrices.

$$\mathbf{C}(\mathbf{u}) = \alpha\mathbf{M} + \beta\mathbf{K}(\mathbf{u}) \tag{4.4}$$

[3]When the results of a static analysis are used to restart the transient analysis, the initial velocity $\dot{\mathbf{u}}_0$ is of course equal to zero.

The positive factors α and β are the so-called Rayleigh damping factors. Note that in the nonlinear case, the damping is dependent on the displacements.

Although this formulation is not correct by far, simple global damping phenomena can be simulated quite well. In section (5.2.2) two models are presented to calculate proper values for α and β.

4.1.4 Solution Strategies

The equation of motion will be solved using a implicit integration method. Implicit integration methods can be divided into two classes: methods to solve first order ODE's and methods to solve second order ODE's. In this particular case the equations of motions will be transformed into a first order ODE first by using Jensen transformation algorithm [13]. The ODE's will be solved using a special class of implicit integration methods, the linear multi-step methods.

In the following sections, an appropriate linear multi-step algorithm will be selected after investigating its accuracy and stability. Furthermore the transformation of the ODE using Jensen algorithm will be discussed. Methods to solve the generated nonlinear system of equation will be reviewed in sections (4.5) and (4.6).

4.2 Linear Multi-step Methods

In many time integration methods, like for instance the well known Runge-Kutta method, the value of a variable $\mathbf{y}(t_n)$ at step t_n is determined by using the information obtained in the previous step, at t_{n-1}. In order to obtain a more accurate and stable iteration method, it can be useful to use the function values in more than one previous meshing point $(t_{n-1}, t_{n-2}, \dots, t_{n-k})$. Such methods are called *linear multi-step methods*[4] (*LMS* in short) or more specific: k-step methods.

Strictly speaking, i.e. from the mechanical point of view, this approach is not correct. The dynamic behavior of structure does not depend on its behavior in the past, it does not have memory capacities. In principle, every calculation can be restarted at any point by using the current displacements and velocities and neglecting all previous steps. In the the, the consequences of using also the points t_{n-1}, t_{n-2}, etc. will be pointed out.

In general, LMS schemes can be divided in two classes: the k step 1 derivative and the k step 2 derivative schemes for first and second order differential equations respectively. Since the dynamic equation will be transformed into a first order system, only the first class of LMS schemes will be considered.

[4]Note that the concept linear in the name linear multi-step methods has nothing to do with the linearity of the dynamic equation (4.1). On the contrary, the LMS method can used to solve nonlinear equations as well. The term linear stands for the linear form in which the method is expressed.

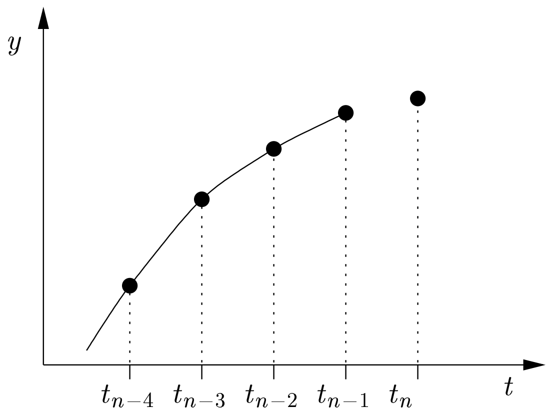


Figure 4.1: The solution of the first order differential equation in the previous time steps

4.2.1 General Form of a Linear Multi-step Scheme

Consider a system of N first order differential equations,

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) \tag{4.5}$$

where $\dot{\mathbf{y}}$ is the first derivative of $\mathbf{y}$ in time t and $\mathbf{f}(\mathbf{y})$ is an arbitrary (non)linear function of $\mathbf{y}$ ($\mathbf{f}(\mathbf{y}) : \mathbb{R}_N \to \mathbb{R}_N$). The equation is supposed to be solved numerically, up to the $n-1^{th}$ step at $t = t_{n-1}$, as indicated in figure (4.1). Since the time-step is constant, it is possible to formulate a k^{th} order polynom that connects these solutions. The solution of (4.5) at $t = t_n$ can be found by extrapolation using this polynom. The polynomials can be constructed by using several techniques, which for example can be found in [7].

The polynomial that connects the current and previous solutions can be written in the following general form.

$$\sum_{i=0}^{k} (\alpha_i \mathbf{y}_{n-i} - h\beta_i \mathbf{f}_{n-i}) = 0 \tag{4.6}$$

The coefficients α_i and β_i follow from the technique that is used to set up the polynom and determine the kind of method. The coefficient α_0 is often set to 1 resulting the following alternative form of an LMS scheme in which the current solution $\mathbf{y}_n$ and its derivative $\dot{\mathbf{y}}_n$ are separated from the previous solutions (note that the identity $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ must hold).

$$\mathbf{y}_n = h_\beta \dot{\mathbf{y}}_n + \mathbf{h}_n^{\mathbf{y}} \tag{4.7}$$

where h_β is the modified time step

$$h_\beta = h\beta_0 \tag{4.8}$$

and $\mathbf{h}_n^{\mathbf{y}}$ the historical portion containing the results in the previous steps:

$$\mathbf{h}_n^{\mathbf{y}} = \sum_{i=1} [h\beta_i \mathbf{f}(\mathbf{y}_{n-i}) - \alpha_i \mathbf{y}_{n-i}] \tag{4.9}$$

Since 1950, a number of LMS schemes have been developed for a variety of purposes. In order to select the best method to solve the the second order differential equation, two phenomena must be considered first, *stability* and *accuracy*. In the next paragraphs these notions are examined.

4.2.2 Spectral Analysis

In order to study the stability behavior of LMS schemes applied to structural dynamics, it is important to investigate the spectral properties of the equations of motion. These spectral properties describe the kinematic behavior of a construction when it is released from a certain deformation with a certain initial velocity, when no external forces are applied. In other words, the dynamic behavior of a structure is stored in the spectral properties.

The equation of motion will be examined by transforming it to a first order O.D.E. first. This will be done using a standard reduction technique, which transforms the equation of motion into a space-state equation by assuming a new vector $\mathbf{y}$ which is composed of both the displacement and the velocity vector.

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{4.10}$$

The equation of motion can be written as

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}[-\mathbf{C}\dot{\mathbf{x}} - \mathbf{K}\mathbf{x} + \mathbf{f}^{\text{ext}}] \tag{4.11}$$

where $\mathbf{M}^{-1}$ is the inverse of the mass matrix. This expression can be combined with (4.10) to yield

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{f}^{\text{ext}} \end{bmatrix} \tag{4.12}$$

or, equivalently

$$\dot{\mathbf{y}} = \mathbf{G}\mathbf{y} + \mathbf{H}(t) \tag{4.13}$$

where $\mathbf{G}(t)$ and $\mathbf{H}(t)$ are given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}; \qquad \mathbf{H}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{f}^{\text{ext}} \end{bmatrix} \tag{4.14}$$

The next step in the spectral analysis is to decompose the system of equations into N uncoupled scalar equations. Each one of these equations describes a particular deformation mode. Since every response is built of several different modes, the characteristics of the total response are embedded in the characteristics of these modes. The eigenvalue problem related to equation (4.13) can be written as:

$$(\mathbf{G} - \lambda\mathbf{1})\boldsymbol{\psi} = \mathbf{0} \tag{4.15}$$

where $\boldsymbol{\psi}$ is an eigenvector of the problem and λ the corresponding eigenvalue. Substituting the expression for $\mathbf{G}$ obtained in equation (4.14) gives the following expression for the eigenvalue problem of the reduced equation of motion.

$$\left(\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}\right) \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.16}$$

where $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ are the two elements of the eigenvector $\boldsymbol{\psi}$,

$$\boldsymbol{\psi} = \begin{bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \end{bmatrix} \tag{4.17}$$

Equation (4.16) can be rewritten as:

$$\begin{aligned} \boldsymbol{\psi}_2 - \lambda \boldsymbol{\psi}_1 &= \mathbf{0} \\ -\mathbf{M}^{-1}\mathbf{K}\boldsymbol{\psi}_1 - \left(\mathbf{M}^{-1}\mathbf{C} + \lambda \mathbf{1}\right)\boldsymbol{\psi}_2 &= \mathbf{0} \end{aligned} \tag{4.18}$$

Substituting the two equations and pre-multiplying by $\mathbf{M}$, yields a quadratic eigenvalue problem in terms of the unknown λ.

$$\left(\mathbf{K} + \lambda \mathbf{C} + \lambda^2 \mathbf{M}\right)\boldsymbol{\psi}_1 = \mathbf{0} \tag{4.19}$$

For the time being, the undamped vibration will be considered by omitting the damping matrix $\mathbf{C}$. The systems corresponding to the l^{th} and the m^{th} eigenmodes can be written as

$$\left(\mathbf{K} + \lambda_l^2 \mathbf{M}\right)\boldsymbol{\psi}_{1l} = \mathbf{0} \tag{4.20}$$

$$\left(\mathbf{K} + \lambda_m^2 \mathbf{M}\right)\boldsymbol{\psi}_{1m} = \mathbf{0} \tag{4.21}$$

Pre-multiplying the first equation by $\boldsymbol{\psi}_{1m}^t$ and the second by $\boldsymbol{\psi}_{1l}^t$ yields after subtraction

$$\left(\lambda_l^2 - \lambda_m^2\right)\boldsymbol{\psi}_{1l}^t \mathbf{M} \boldsymbol{\psi}_{1m} = 0 \tag{4.22}$$

Since by definition $\lambda_l \neq \lambda_m$ the following must hold

$$\boldsymbol{\psi}_{1l}^t \mathbf{M} \boldsymbol{\psi}_{1m} = \delta_m^l \tag{4.23}$$

where δ_m^l is the Kronecker delta. According to the same orthogonality principle, the eigenvalue for the stiffness matrix can be determined.

$$\boldsymbol{\psi}_{1l}^t \mathbf{K} \boldsymbol{\psi}_{1m} = -\lambda_{1l}^2 \delta_m^l \qquad \text{(no sum)} \tag{4.24}$$

In case of equations of motion, the eigenvalue λ_l is often referred to as the eigenfrequency belonging to the l^{th} mode ω_l.

$$\omega_l^2 = -\lambda_l^2 \tag{4.25}$$

With the model decomposition of the undamped equation at hand, the damping matrix $\mathbf{C}$ can be re-inserted. Using Rayleigh damping as presented in the introduction and the orthogonality functions (4.23) and (4.24), the damping matrix can be written as

$$\boldsymbol{\psi}_l^t \mathbf{C} \boldsymbol{\psi}_m = (\alpha - \omega_l^2 \beta)\delta_m^l \tag{4.26}$$

The characteristics of an arbitrary dynamic response of a structure can be expressed in terms of eigenfrequencies and damping ratio's. The damping of a single degree of freedom system is always expressed as

$$c = 2\xi_l \omega_l \tag{4.27}$$

where ξ is the damping ratio. In this case the damping is equal to $(\alpha - \omega_l^2 \beta)$. The damping ratio of the l^{th} mode can now be expressed in terms of the Rayleigh constants.

$$\xi_l = \frac{\alpha + \beta}{-2\lambda_l} \tag{4.28}$$

After substituting of the Rayleigh damping matrix and using the terminology above, the solution in equation (4.26) can be written as

$$\lambda^2 + 2\xi_l \omega_l \lambda + \omega_l^2 = 0 \tag{4.29}$$

The equation can be solved analytically for the eigenvalues λ. The possible solutions of this equation can be divided into four groups.

	Damping	Eigenvalues	Description of response
1.	$\xi = 0$	$\lambda_{1,2} = \pm i\omega$	Undamping vibration
2.	$0 \leq \xi < 1$	$\lambda_{1,2=} - \xi\omega \pm i\omega\sqrt{1-\xi^2}$	Underdamped vibration
3.	$\xi = 1$	$\lambda_{1,2} = -\omega$	Critically damped vibration
4.	$\xi > 1$	$\lambda_{1,2} = -\xi\omega + i\omega\sqrt{1-\xi^2}$	Overdamped vibration

The complex solutions for $\lambda_{1,2}$ can be plotted in a complex plane $\mathbb{C}$. All possible eigenvalues are in the negative half plane of this complex plane, including the imaginary axis, fig. (4.2), since by definition both ξ_l and ω_l are assumed to be positive and real.

4.2.3 Stability Regions of LMS Schemes

In the investigation of the spectral properties of an arbitrary first order LMS scheme, the following single degree of freedom, first order ODE will be considered.

$$\dot{y} = \lambda y \tag{4.30}$$

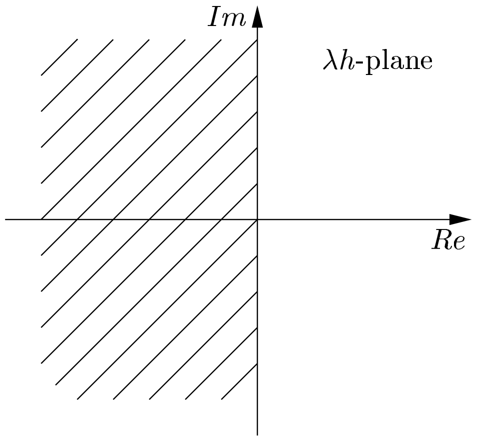


Figure 4.2: Area in the complex plane in which $\lambda_{1,2}$ occurs

where λ is the complex eigenvalue for this problem. An analytical solution of this simple equation is known. At a time $t = t_n$ ($t_n > 0$) and for an initial value $y(t_{n-1})$ the value of $y(t_n)$ is:

$$y(t_n) = e^{\lambda[t_n - t_{n-1}]} y(t_{n-1}) \qquad \forall \quad t_n - t_{n-1} > 0 \tag{4.31}$$

In the previous section it has been shown that λ is always complex and in the negative half plane of the complex space. The solution of the first order ODE, $y(t_n)$, is therefore complex and decaying as well.

$$|y(t_n)| < |y(t_{n-1})| \qquad \forall \quad t_n > 0 \tag{4.32}$$

A numerical solution can be obtained by using an LMS scheme, as presented in equation (4.6). In this case $f(y) = \lambda y$, so that,

$$\sum_{i=0}^{k} (\alpha_i + h\beta_i\lambda_l) y_{n-i} = 0 \tag{4.33}$$

In this equation the new results y_n can be connected to the previous results y_{n-i} with the amplification matrix $\mathbf{A}$.

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n+1-k} \end{bmatrix} = \mathbf{A} \begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-k} \end{bmatrix} \tag{4.34}$$

where the $k \times k$ amplification matrix can be written as:

$$\mathbf{A} = \begin{bmatrix} \frac{-(\alpha_1 - \lambda h\beta_1)}{\alpha_0 - \lambda h\beta_0} & \frac{-(\alpha_2 - \lambda h\beta_2)}{\alpha_0 - \lambda h\beta_0} & \frac{-(\alpha_3 - \lambda h\beta_3)}{\alpha_0 - \lambda h\beta_0} & \dots & \frac{-(\alpha_k - \lambda h\beta_k)}{\alpha_0 - \lambda h\beta_0} \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{4.35}$$

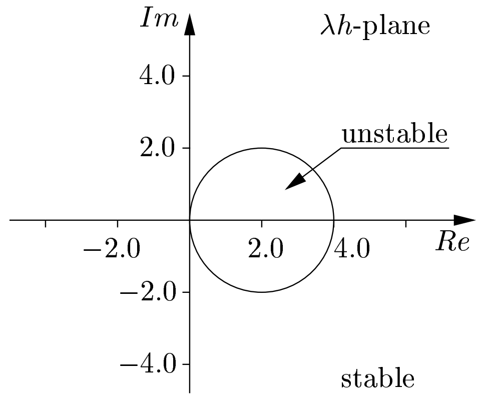


Figure 4.3: Stability region of an arbitrary LMS scheme

Since the solution of the differential equation is always decaying, the amplification of the solution may never be greater than $||1||$. In order words, the convergence of an LMS scheme can be proven when

$$||\mathbf{A}|| < 1 \tag{4.36}$$

where $||\mathbf{A}||$ is the *natural matrix norm* of the matrix $\mathbf{A}$. There are many examples of natural matrix norms that can be applied to this equation. A more applicable condition is given by Hughes *et al.* [10].

Definition 4.1 An LMS scheme is stable when all eigenvalues $\mu_{l=1,2,\ldots k}$ of the amplification matrix $\mathbf{A}$ are smaller than one in modulus.

The eigenvalues μ_l of the amplification matrix $\mathbf{A}$ can be determined using the following formula

$$||\mathbf{A} - \mu_l \mathbf{1}|| = 0 \tag{4.37}$$

The collection of complex values $\lambda_l h$, for which the eigenvalues of the amplification matrix are smaller than 1, can be plotted in the complex plane, as shown in figure 4.3. The stability behavior of LMS schemes (and time integration methods in general) can be divided into a number of classes measured to the stability properties. These classes are proposed by C.W. Gear [7, 10]. The two most important degrees of stability are *absolute stability* and *A-stability.*

Definition 4.2 The region of *absolute stability* of an LMS method is the set of $\lambda h \in \mathbb{C}$ at which the method is absolutely stable

Definition 4.3 A numerical method is said to be *A-stable* if the solution produced by the LMS scheme approaches 0 when the number of steps goes to infinity.

If an LMS scheme is stable for all possible imaginary values of λh it is called *unconditionally stable.* In practice this means that there is no restriction in the choice of the time step h. All LMS schemes where the stable region includes the complete left-hand-side of the complex plane are called unconditionally stable. When due to numerical damping (see next section) the solution decays to zero when the number of steps n go to infinity, the scheme is also A-stable. The most useful LMS schemes are those that satisfy these two conditions.

4.2.4 Accuracy

Besides the stability of the method, there are other important characteristics that influence the performances of an LMS scheme. Although the iteration process can converge, there may be errors in the calculations. These errors can be divided into 3 categories: the truncation error, numerical damping and frequency distortion or the so-called phase shift.

Truncation error

The truncation error is the difference between an analytically correct answer at a certain point t_k and the numerical solution at this point. Although the chosen time step does not influence the stability of an A-stable method, it does has its influences on the error. When the time-step is larger, the truncation error will be larger too. This error may seem small in the beginning, errors made in previous steps will accumulate the next error.

Numerical Damping

The response of a structure calculated with an A-stable LMS scheme always decays to zero, even when the there are no damping terms in the equation of motion. During the calculation, a little amount of energy is dissipated. This effect is called numerical damping and is inherent to most LMS schemes. Despite it truncates the numerical solution, it has a number of benifits.

The unstability of a time integration method can be seen as the presence of negative numerical damping. The total amount of energy grows and hence the displacements or velocities are every step a little to high compared to the real converged solution. After a number of steps, these disturbances become bigger and bigger with divergence as a concequence. When there is a little amount of numerical damping, these effects are immediately counteracted and the solution remains stable.

The numerical damping is most often proportional to the time step. Time steps that are large compared to the vibration mode periods, have a large amount of damping as a result. As a consequence of this, numerical damping has another additional benefit. The overtones of a vibration, which are recognized by their high frequencies, are damped since the chosen time step is very large compared to their periods.

It is difficult to determine the amount of numerical damping for each method analytically, so one must rely on numerical techniques. For instance, the damping can be determined by calculating the numerical behavior of a system, with a known analytical response.

Frequency distortion

Frequency distortion, or *period shift*, is the effect that the calculated period of the vibration is shorter than the actual period. Again, this effect is proportional to the time step. When the time-step is small compared to the period of the vibration, the frequency distortion will be small.

4.2.5 Nonlinear Stability

When the equations of motion are nonlinear, the eigenvalue λ_l is not constant and usually varies at each time step. It can even be possible that the eigenvalue becomes positive at some time steps. The previous conclusions concerning the stability of the LMS methods are no longer valid under all circumstances.

Since the eigenvalue is also time dependent, an alternative notation will be used. The eigenvalue at the n^{th} step will be denoted by λ_n. The amplification matrix $\mathbf{A}$ for nonlinear equations can be written as

$$\mathbf{A} = \begin{bmatrix} \frac{-(\alpha_1 - \lambda_{n-1} h\beta_1)}{1-\lambda_n h\beta_0} & \frac{-(\alpha_2 - \lambda_{n-2} h\beta_2)}{1-\lambda_n h\beta_0} & \frac{-(\alpha_3 - \lambda_{n-3} h\beta_3)}{1-\lambda_n h\beta_0} & \dots & \frac{-(\alpha_k - \lambda_{n-k} h\beta_k)}{1-\lambda_n h\beta_0} \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

The eigenvalues μ of the amplification matrix depend on the combination of the current and the previous eigenvalue λ_{n-k} of the problem. Again the LMS scheme is stable if all eigenvalues μ of the amplification matrix are smaller than 1. However, since the system of stability regions depends on more than one eigenvalue λ, it cannot be drawn in the complex plane anymore.

4.3 Selection of an Appropriate LMS

It may be clear that the choice of the LMS scheme relies on the type of equation that has to be solved. Linear or nonlinear equations require different integration methods and even the presence of a damping matrix can influence the right choice for an LMS scheme. In this section a number of LMS schemes, which have been developed for structural equations

4.3.1 Trapezoidal Rule

It has been proven by Dahlquist that the most accurate A-stable method is the *trapezoidal rule* [10]. This method is in fact a 1-step method, but it will be presented as an LMS method. The trapezoidal rule is one of the manifestations of the Newmark algorithm as presented by Bathe [1], which can be understood to be an extension of the linear acceleration method. The displacement and velocity vector can be calculated using

$$\mathbf{u}_n = \mathbf{u}_{n-1} + h[(1-\delta)\dot{\mathbf{u}}_{n-1} + \delta\dot{\mathbf{u}}_n] \tag{4.38}$$

where the free variable δ is a parameter that can be used to tune the integration accuracy and stability. In case $\delta = \frac{1}{2}$ the trapezoidal rule is obtained. In the LMS conventions, the α_i and β_i coefficients are (recalling that $\alpha_0 = 1$):

Method	β_0	β_1	α_1	α_2	α_3	Trunc. Error
Trap. rule	$\frac{1}{2}$	$\frac{1}{2}$	-1			$\mathcal{O}(\frac{1}{12}h^3)$

Table 4.1: LMS constants and truncation error for the trapezoidal rule

Stability

The implicit trapezoidal rule is unconditionally stable for linear equations. The stability region encloses the complete left hand side of the complex plane $\mathbb{C}$, the imaginary axis included, as can be seen from figure (4.4). The characteristics of this method in nonlinear analysis are rather poor due to the changing eigenvalues of the response. When one of the eigenvalues of a previous step λ_{n-1} is smaller than the current eigenvalue, the method becomes unstable. For this reason it cannot be used in a nonlinear analysis.

Accuracy

The method appears to be very accurate. As can be seen in the figures (4.5) and (4.6) the trapezoidal rule does not have any numerical damping at all and the frequency distortion is the lowest one of all known LMS schemes. The truncation error is also in proportion. Most probably, the absense of numerical damping leads to the unstability in nonlinear analysis.

4.3.2 Gear's Method

One of the first methods that were psecifically designed to deal with the stability problems of stiff equations were the k-step Gear's methods. Gear proposed a set of 6 k-step methods, but only the 2 and 3 step methods will be discussed here. The derivation of these methods is rather complicated and will not be reviewed in this context. For more information, one can refer to [7]. The LMS coefficients of these methods are:

Method	β_0	α_1	α_2	α_3	α_4	Trunc. error
Gear 2 step	$\frac{2}{3}$	$\frac{4}{3}$	$-\frac{1}{3}$			$\mathcal{O}(\frac{2}{9}h^3)$
Gear 3 step	$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$		$\mathcal{O}(\frac{3}{22}h^4)$

Table 4.2: LMS constants and truncation errors for Gear's 2 and 3 step method

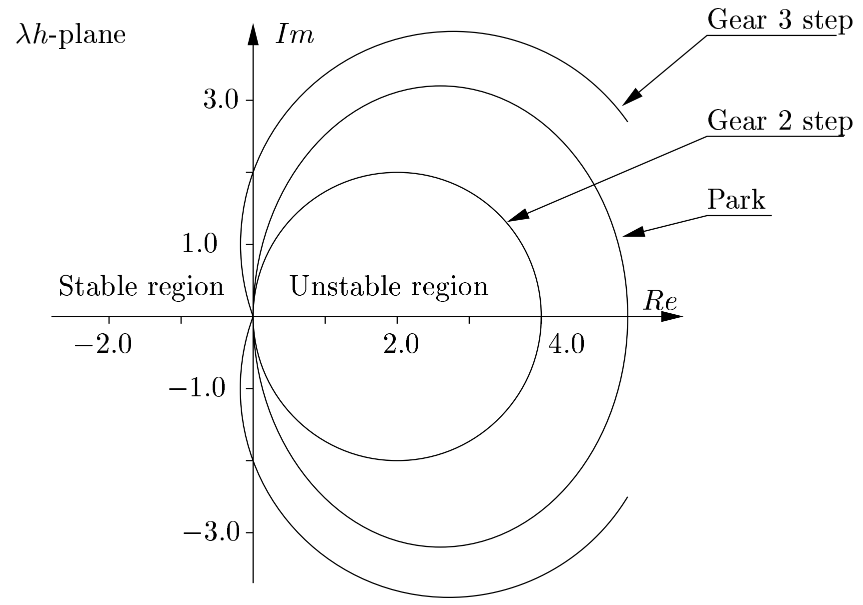


Figure 4.4: Stability regions for the trapezoidal rule, Gear's multi-step methods and Park's method.

Stability

The stability regions of the Gear 2- and 3-step methods are illustrated in figure (4.4). As may be concluded from this figure, the 2-step Gear method is in fact A-stable, that is, the region is which instability occurs is completely in the first and fourth quadrant of the complex plane. The imaginary axis is completely in the stable region. The instability region of the 3- (and higher step-) Gear methods however contains a little piece of the second and third quadrant as well as the imaginary axis. As a result of this, this LMS scheme is inappropriate for structural dynamic applications.

Accuracy

Although the 2- step Gear method is unconditionally stable, it produces a reasonable amount of numerical damping. When the time step is chosen $1/10^{th}$ of the period, the numerical damping $\bar{\xi}$ is over 0.1. The unstability of the three step method also reflects in the figure for numerical damping. For small time steps, the numerical damping is negative. This condition will always lead to unstability.

4.3.3 Park's Method

The multi-step methods described earlier all have some serious problems. Most methods have very poor stability behavior, like for instance the trapezoidal rule or Gear's 3-step method. On the other hand, the unconditionally stable Gear's 2-step method adds a large amount of numerical damping to the system.

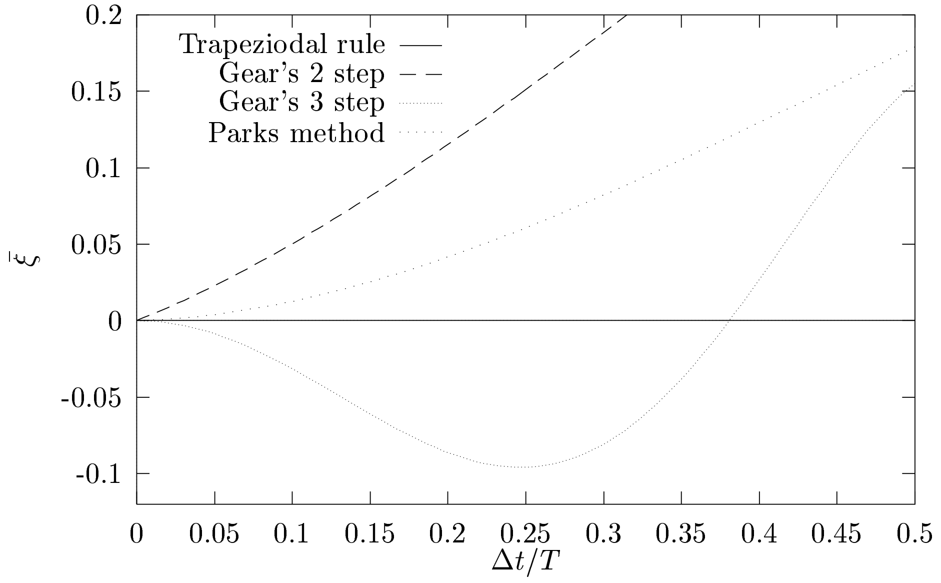


Figure 4.5: Numerical damping ratios versus relative time step h/T of the trapezoidal rule, Gear's multi-step methods and Park's method.

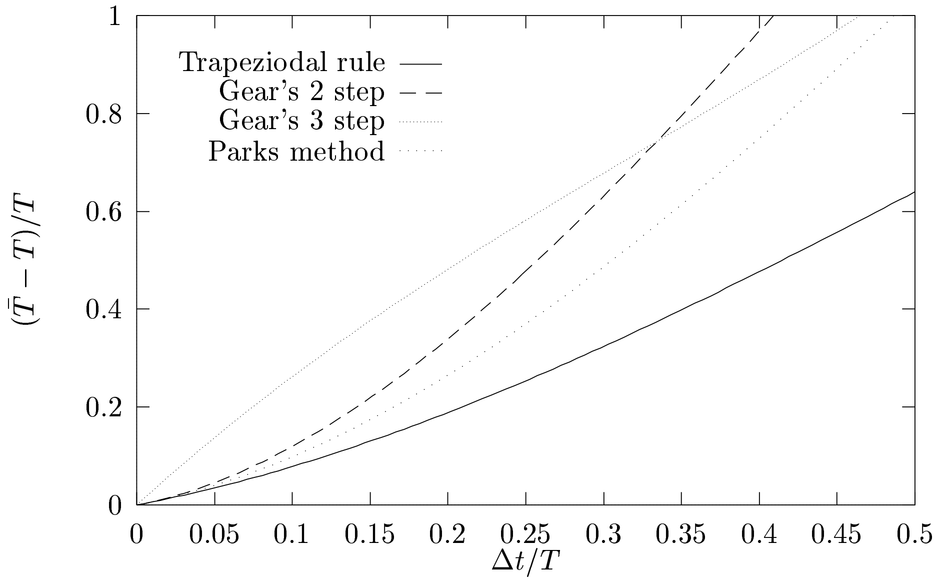


Figure 4.6: Numerical period shift or frequency distortion versus relative time step h/T of the trapezoidal rule, Gear's multi-step methods and Park's method.

Park's method intents to combine two such methods, in order to eliminate both bad characteristics with an unconditionally stable method, with hardly any damping, as a result [21]. The new method P is constructed out of two methods (m and n steps respectively) using the following equation:

$$P_{m,n} = C_m G_m + C_n G_n \tag{4.39}$$

where the multipliers C_m and C_n are constant and the identity

$$C_m + C_n = 1 \tag{4.40}$$

must hold. It need no proof that when both G_m and G_n are linear multi-step methods, $P_{m,n}$ is a linear multi-step method as well. The best result, with respect to the stability of the method, is obtained when using Gear's 2-step method (G_2 in short) first. In principle every combination of this method and an arbitrary (unstable) method, using the right multipliers, will lead to an unconditionally stable method.

$$P_{2,n} = C_2 G_2 + C_n P_n \tag{4.41}$$

The best results however can be achieved by combining the Gear's 2 and 3 step methods using equal weighting coefficients, $C_2 = C_3 = \frac{1}{2}$. The coefficients for the LMS scheme α_i and β_i become

Method	β_0	β_1	α_1	α_2	α_3	Trunc. error
Park	$\frac{6}{10}$		-$\frac{15}{10}$	$\frac{6}{10}$	-$\frac{1}{10}$	$\mathcal{O}(\frac{1}{10}h^3)$

Table 4.3: LMS constants for Park's method

Stability

According to figure (4.4) it may be clear that the Park method is indeed A-stable, the complete left-hand-side of the complex plane as well as the imaginary axis is in the stability region.

It can be shown[20] that for nonlinear equations, the stability of the method depends on the current eigenvalue only, instead of a combination of previous and current eigenvalues, which was the case for the trapezoidal rule. Since Park's method is A-stable in the linear case, it is also A-stable for nonlinear equations.

Accuracy

Furthermore, the amount of numerical damping is smaller compared to Gear's 2-step method, as well as the frequency distortion. This is the result of the combination between the overdamped 2-step method and the 3-step method with negative damping. Also the amount of frequency distortion is an average of both methods. In case of nonlinear equations the numerical damping and frequency distortion are not constant, since the eigenvalue λ changes at every step.

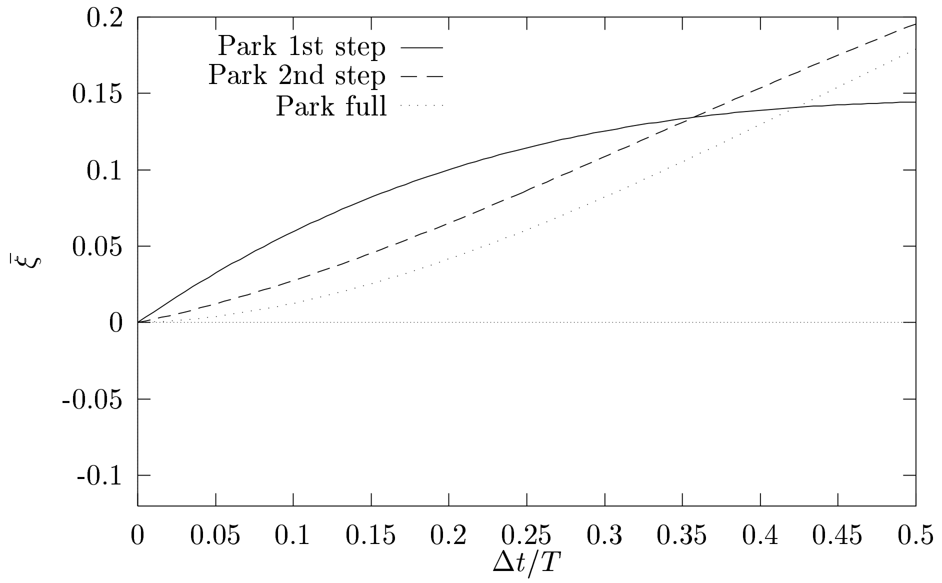


Figure 4.7: Numerical damping for Park's method, including 1^{st} and 2^{nd} step

4.3.4 Cold Restart Conditions

It is obvious that LMS schemes cannot be used when there are not enough previous points available. For instance, the 3-step Park method requires special starting algorithms for the first and second step. It is possible to use the trapezoidal rule and Gear's 2-step method for the first two steps. Nevertheless, when the equation is nonlinear, the trapezoidal rule can be become unstable. Gear's two step method produces a large amount of numerical damping.

Together with the development of his 3-step method, Park proposed a 1- and 2-step method as starting algorithms.

Method	β_0	β_1	α_1	α_2	α_3
Park (1^{st} step)	$\frac{6}{10}$	$\frac{4}{10}$	-1		
Park (2^{nd} step)	$\frac{6}{10}$	$\frac{2}{10}$	$-\frac{12}{10}$	$\frac{2}{10}$	
Park (full)	$\frac{6}{10}$		$-\frac{15}{10}$	$\frac{6}{10}$	$-\frac{1}{10}$

Table 4.4: LMS scheme for Park's method, including 1^{st} and 2^{nd} step

Although these starting algorithms have worse accuracy qualities compared to the full Park method, their stability behavior is on the same level.

4.3.5 Conclusion

The best choice of an LMS scheme depends on the spectral properties of the ODE which has to be solved. In case of the stiffly stable kinematic equation, three methods are absolutely stable, the trapezoidal rule, Gear's 2-step method and Park's method.

Gear's 2 step method is the least accurate of these. It has a rather large amount of numerical damping. The trapezoidal rule does not have any numerical damping at all, but can become unstable when the equation is nonlinear. Park's method is the most allround method. Its stability characteristics are good even in the nonlinear case, although it is less accurate than the trapezoidal rule.

When the equation is linear, the trapezoidal rule is the best choice, since the method is unconditionally stable for linear equations and it produces no numerical damping. When the equation is nonlinear, Park's 3-step method is preferred. Although it produces a notable amount of numerical damping, its stability behavior for nonlinear is much better. Gear's 2- and 3-step methods do not have this positive characteristics, but may be useful in some very specific situations.

4.4 Implementation of the LMS method

Before the LMS schemes can be used, the system of second order differential equations has to be transformed into a system of first order ODE's. In the previous section, when the spectral properties were determined, the equation was transformed into a so-called state-space. Although this method pleases for a rather straightforward, analytical evaluation of a differential equation, it has a lot of disadvantages. First of all, the number of equations will be doubled. In practice, a system of equations which is 2 times as big, needs approximately $2^2 = 4$ times as much numerical actions to solve. Also, the presence of the inverse of the mass matrix ($\mathbf{M}^{-1}$) in the formulation, will make the solution procedure more sensitive for badly formed mass matrices.

The usage of an LMS scheme offers the possibility to use an alternative method proposed by Jensen [13]. This method does not have the two problems mentioned above. It is very robust and extremely suitable for numerical applications.

4.4.1 Jensen's Transformation Algorithm

Consider an auxiliary system of equations of the form

$$\mathbf{v} = \mathbf{AM\dot{u}} + \mathbf{Bu} \tag{4.42}$$

Where the matrices $\mathbf{A}$ and $\mathbf{B}$ are arbitrary except that $\mathbf{A}$ is not singular. The derivation of this equation with respect to time t gives

$$\dot{\mathbf{v}} = \mathbf{AM\ddot{u}} + \mathbf{B\dot{u}} \tag{4.43}$$

Multiplication of the (non)linear dynamic equilibrium equation (4.3) with the matrix $\mathbf{A}$ will give

$$\mathbf{AM\ddot{u}} + \mathbf{AC\dot{u}} + \mathbf{AKu} = \mathbf{Af}^{\text{ext}}(t) \tag{4.44}$$

Substituting (4.43) into (4.44) results in the following system of equations:

$$\begin{bmatrix} \mathbf{AM} & \mathbf{0} \\ \mathbf{AC} - \mathbf{B} & \mathbf{I} \end{bmatrix} + \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \end{bmatrix} \begin{bmatrix} \mathbf{B} & -\mathbf{I} \\ \mathbf{AK} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{Af}^{\text{ext}} \end{bmatrix} \tag{4.45}$$

The general formula for a linear multi-step algorithm (4.7) can now be substituted.

$$\begin{aligned} \begin{bmatrix} \mathbf{AM} & \mathbf{0} \\ \mathbf{AC} - \mathbf{B} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} + h_\beta \begin{bmatrix} \mathbf{B} & -\mathbf{I} \\ \mathbf{AK} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \\ h_\beta \begin{bmatrix} \mathbf{0} \\ \mathbf{Af}^{\text{ext}} \end{bmatrix} + \begin{bmatrix} \mathbf{AM} & \mathbf{0} \\ \mathbf{AC} - \mathbf{B} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{h}^{\mathbf{u}} \\ \mathbf{h}^{\mathbf{v}} \end{bmatrix} \end{aligned} \tag{4.46}$$

After collecting terms

$$\begin{bmatrix} \mathbf{AM} + h_\beta \mathbf{B} & -h_\beta \mathbf{I} \\ \mathbf{A}(\mathbf{C} + h_\beta \mathbf{K}) - \mathbf{B} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{AMh}^{\mathbf{u}} \\ \mathbf{Aq} \end{bmatrix} \tag{4.47}$$

where $\mathbf{h}^{\mathbf{u}}$ and $\mathbf{h}^{\mathbf{v}}$ are the historical vectors of the displacement $\mathbf{u}$ and auxiliary vector $\mathbf{v}$ respectively and

$$\mathbf{Aq} = h_\beta \mathbf{Af}^{\text{ext}} + (\mathbf{AC} - \mathbf{B})\mathbf{h}^{\mathbf{u}} + \mathbf{h}^{\mathbf{v}} \tag{4.48}$$

This matrix equation can be solved algebraically by multiplying the second row with h_β and multiplying with $\mathbf{A}^{-1}$ to obtain

$$\mathbf{Eu} = \mathbf{Mh}^{\mathbf{u}} + h_\beta \mathbf{q} \tag{4.49}$$

where $\mathbf{E}$ is the *dynamic stiffness matrix*

$$\mathbf{E} = \mathbf{M} + h_\beta \mathbf{C} + h_\beta^2 \mathbf{K} \tag{4.50}$$

The auxiliary vector $\mathbf{v}$ can be solved using

$$\mathbf{v} = \mathbf{Aq} - [\mathbf{A}(\mathbf{C} + h_\beta \mathbf{K}) - \mathbf{B}]\mathbf{u} \tag{4.51}$$

The equations still contain the two matrices $\mathbf{A}$ and $\mathbf{B}$. Since they could be chosen arbitrary it is convenient to take $\mathbf{A} = \mathbf{I}$ and $\mathbf{B} = \mathbf{C}$, remaining the so-called *linear Jensen equation*,

$$[\mathbf{M} + h_\beta \mathbf{C} + h_\beta^2 \mathbf{K}]\mathbf{u} = \mathbf{Mh}^{\mathbf{u}} + h_\beta \mathbf{h}^{\mathbf{v}} + h_\beta^2 \mathbf{f}^{\text{ext}} \tag{4.52}$$

which can be re-written as

$$\mathbf{Eu} = \mathbf{g} \tag{4.53}$$

where $\mathbf{g}$ the *dynamic history-force vector*:

$$\mathbf{g} = \mathbf{M}\mathbf{h}^{\mathbf{u}} + h_\beta \mathbf{h}^{\mathbf{v}} + h_\beta^2 \mathbf{f}^{\text{ext}} \tag{4.54}$$

The linear Jensen equation can now be solved for $\mathbf{u}$. When a nonlinear element formulation is used, the term $\mathbf{Ku}$ in equation (4.52) must be replaced by $\mathbf{f}^{\text{int}}(\mathbf{u})$ yielding the *nonlinear Jensen equation*

$$\mathbf{Mu} + h_\beta \mathbf{Cu} + \mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{M}\mathbf{h}^{\mathbf{u}} + h_\beta \mathbf{h}^{\mathbf{v}} + h_\beta^2 \mathbf{f}^{\text{ext}}(t) \tag{4.55}$$

The solution of this particular equation requires nonlinear solution techniques, which will be described in section 4.5. After the displacement $\mathbf{u}$ has been calculated, the velocity of the system can be determined by using the original multi-step formula (4.7).

$$\dot{\mathbf{u}} = \frac{\mathbf{u} - \mathbf{h}^{\mathbf{u}}}{h_\beta} \tag{4.56}$$

The accelerations need not to be calculated in this procedure.

4.4.2 Prescribed Displacements

The prescribed displacements can be formulated as time dependent boundary conditions. Since they can be nonzero, prescribed displacements imply an internal force. In order to get a better idea of handling time dependent boundary conditions in a system of equations, a simple example, the linear static equation, is considered first.

Linear Static Equation

Consider the following linear system of static equilibrium equations

$$\mathbf{Ku} = \mathbf{f}^{\text{ext}} \tag{4.57}$$

where $\mathbf{K}$ is the stiffness matrix and $\mathbf{f}^{\text{ext}}$ the external forces vector. The total number of degrees of freedom in the equation is N. There are M $(0 \leq M < N)$ prescribed displacements. The displacement vector $\mathbf{u}$ can be split into 2 parts, $\mathbf{u}^f$ and $\mathbf{u}^p$, where the subscripts f and p stand for free and prescribed respectively. The vector $\mathbf{u}^p$ consists of zeros except for the degrees of freedom that are prescribed; in $\mathbf{u}^f$ all elements are nonzero except for the prescribed ones. Obviously, the following equation must hold

$$\mathbf{u} = \mathbf{u}^p + \mathbf{u}^f \tag{4.58}$$

Equation (4.58) can be substituted into the equilibrium equation (4.57)

$$\mathbf{K}\mathbf{u}^f + \mathbf{K}\mathbf{u}^p = \mathbf{f}^{\text{ext}} \tag{4.59}$$

The second term, $\mathbf{K}\mathbf{u}^p$, is known and can therefore be carried to the right hand side of the equation

$$\mathbf{K}\mathbf{u}^f = \mathbf{f}^{\text{ext}} - \mathbf{K}\mathbf{u}^p = \mathbf{f}^{\text{ext}} - \mathbf{f}^p \tag{4.60}$$

where $\mathbf{f}^p$ denotes the forces due to prescribed displacements. The system of equations can now be solved for $\mathbf{u}^f$. Some problems can occur since the system is undetermined: the number of degrees of freedom has been reduced to $N-M$, while the number of equations remains N. Such an 'unbalanced' system must be solved using a penalty values technique. This technique is widely used in finite element methods and therefore too trivial to discuss here.

When the solution for $\mathbf{u}^f$ has been obtained, the displacement vector must be recomposed by adding the prescribed displacements to the reduced displacement vector yielding $\mathbf{u} = \mathbf{u}^f + \mathbf{u}^p$.

The Dynamic equation

Since the velocities and the accelerations are the time derivatives of the displacement vector, these vectors can also be divided into a free and a prescribed one.

$$\ddot{\mathbf{u}} = \ddot{\mathbf{u}}^f + \ddot{\mathbf{u}}^p \tag{4.61a}$$

$$\dot{\mathbf{u}} = \dot{\mathbf{u}}^f + \dot{\mathbf{u}}^p \tag{4.61b}$$

$$\mathbf{u} = \mathbf{u}^f + \mathbf{u}^p \tag{4.61c}$$

The prescribed accelerations and velocities introduce inertia and damping forces. Following the same procedure as before, the nonlinear dynamic equation can be written as

$$\mathbf{M}\ddot{\mathbf{u}}^f + \mathbf{M}\ddot{\mathbf{u}}^p + \mathbf{C}\dot{\mathbf{u}}^f + \mathbf{C}\dot{\mathbf{u}}^p + \mathbf{f}^{\text{int}}(\mathbf{u}) + \mathbf{f}^{\text{int}}(\mathbf{u}^p) = \mathbf{f}^{\text{ext}}(t) \tag{4.62}$$

Using Jensen procedure and an LMS method, the following nonlinear equation can be obtained.

$$\mathbf{M}\mathbf{u}_n + h_\beta \mathbf{C}(\mathbf{u}_n) + h_\beta^2 \mathbf{f}^{\text{int}}(\mathbf{u}_n) = \mathbf{M}\mathbf{h}_n^{\mathbf{u}} - h_\beta \mathbf{h}_n^{\mathbf{v}} - h_\beta^2 \mathbf{f}^{\text{ext}}(t) - \mathbf{g}^p \tag{4.63}$$

where $\mathbf{g}^p$ is the force vector due to prescribed displacements

$$\mathbf{g}^p = \mathbf{M}\mathbf{u}_n^p + h_\beta \mathbf{C}(\mathbf{u}_n^p) + h_\beta^2 \mathbf{f}^{\text{int}}(\mathbf{u}_n^p) \tag{4.64}$$

or, when the equation is linear

$$\mathbf{g}^p = \mathbf{M}\mathbf{u}_n^p + h_\beta \mathbf{D}(\mathbf{u}_n^p) + h_\beta^2 \mathbf{K}\mathbf{u}_n^p \tag{4.65}$$

The reduced system can be now solved for $\mathbf{u}^f$ as described above. Afterwards, the correct displacement vector $\mathbf{u}$ must be recomposed. The velocities (and accelerations) can now be calculated using with the total history vector $\mathbf{h}_n^{\mathbf{u}}$, according to the original LMS formula

$$\dot{\mathbf{u}}_n = \frac{\mathbf{u}_n - \mathbf{h}_n^{\mathbf{u}}}{h_\beta} \tag{4.66}$$

Using this method, it is not necessary to evaluate the prescribed velocities by differentiating the prescribed displacements. The total velocity vector can be calculated using the total displacement vector.

4.4.3 The General Procedure

Summarizing, the full procedure to come to the linear or nonlinear Jensen equation is given in the table below stepwise.

```
1.    Linear:          $\dot{\mathbf{v}}_{n-1} = \mathbf{f}^{\text{ext}}(t_{n-1}) - \mathbf{f}^{\text{int}}(\mathbf{u}_{n-1})$

      Nonlinear:       $\dot{\mathbf{v}}_{n-1} = \mathbf{f}^{\text{ext}}(t_{n-1}) - \mathbf{K}\mathbf{u}_{n-1}$

2a.   if $n = 0$,
        Linear:        $\mathbf{v}_0 = \mathbf{M}\dot{\mathbf{u}}_0 + \alpha\mathbf{M}\mathbf{u}_0 + \beta\mathbf{K}\mathbf{u}_0$

        Nonlinear:     $\mathbf{v}_0 = \mathbf{M}\dot{\mathbf{u}}_0 + \alpha\mathbf{M}\mathbf{u}_0 + \beta\mathbf{f}^{\text{int}}(\mathbf{u}_0)$
2b.   elseif $n > 0$,
                       $\mathbf{v}_{n-1} = \mathbf{h}^{\mathbf{v}}_{n-1} + h_\beta\dot{\mathbf{v}}_{n-1}$

3.                     $\mathbf{h}^{\mathbf{v}}_n = h\sum_{i=1}\beta_i\dot{\mathbf{v}}_n - \sum_{i=1}\alpha_i\mathbf{v}_n$

4.                     $\mathbf{h}^{\mathbf{u}}_n = h\sum_{i=1}\beta_i\dot{\mathbf{u}}_n - \sum_{i=1}\alpha_i\mathbf{u}_n$

5.                     $\mathbf{g}_n = \mathbf{M}\mathbf{h}^{\mathbf{u}}_n - h_\beta\mathbf{h}^{\mathbf{u}}_n - h_\beta^2\mathbf{f}^{\text{ext}}(t_n)$

6.    Linear:          $\mathbf{g}^p_n = (1 + h_\beta\alpha)\mathbf{M}\mathbf{u}^p_n + h_\beta(\beta + h_\beta)\mathbf{K}\mathbf{u}^p_n$

      Nonlinear:       $\mathbf{g}^p_n = (1 + h_\beta\alpha)\mathbf{M}\mathbf{u}^p_n + h_\beta(\beta + h_\beta)\mathbf{f}^{\text{int}}(\mathbf{u}^p_n)$

7.    Solve for $\mathbf{u}_n$:

      Linear:          $[(1 + \alpha h_\beta)\mathbf{M} + h_\beta(\beta + h_\beta)\mathbf{K}]\mathbf{u}_n = \mathbf{g}_n - \mathbf{g}^p_n$

      Nonlinear:       $(1 + \alpha h_\beta)\mathbf{M}\mathbf{u}_n + h_\beta(\beta + h_\beta)\mathbf{f}^{\text{int}}(\mathbf{u}_n) = \mathbf{g}_n - \mathbf{g}^p_n$

8.                     $\dot{\mathbf{u}}_n = (\mathbf{u}_n - \mathbf{h}^{\mathbf{u}}_n)/h_\beta$

9.                     Advance step number $n = n + 1$ and time $t = t + h$

10.   If $t < t_{max}$
                       go to 1.
      else
                       stop.
      endif
```

The nonlinear equation at step number 7 must be solved using iterative techniques. These methods are discussed in the next section.

4.4.4 Additional Calculations

Apart from the displacements, velocities and in the nonlinear case internal forces some additional quantities can be calculated on request.

Kinetic Energy

In order to control the mode-jumping process (which will be discussed in the next chapter), and to check the amount of numerical damping, the kinetic, and strain energy can be calculated at each step. The total energy at a time t can be calculated using the following integral equation.

$$\int_0^t \dot{\mathbf{u}}(\tau)[\mathbf{M}\ddot{\mathbf{u}}(\tau) + \mathbf{C}\dot{\mathbf{u}}(\tau) + \mathbf{f}(\mathbf{u}(\tau);\lambda)]d\tau = 0 \tag{4.67}$$

The kinetic energy and the strain energy are:

$$T = \int_0^t \dot{\mathbf{u}}(\tau)\mathbf{M}\ddot{\mathbf{u}}(\tau)d\tau \tag{4.68}$$

$$\Psi = \int_0^t \dot{\mathbf{u}}(\tau)\mathbf{f}(\mathbf{u}(\tau);t)d\tau \tag{4.69}$$

The dissipated energy, a function of the damping matrix $\mathbf{C}$ is less important in this case. Integration of these 2 terms gives

$$T = \frac{1}{2}\dot{\mathbf{u}}_n\mathbf{M}\dot{\mathbf{u}}_n \tag{4.70}$$

$$\Psi = \frac{1}{2}\mathbf{u}_n\mathbf{f}(\mathbf{u}_n;t_n) \tag{4.71}$$

Accelerations

In the Jensen procedure, the accelerations are not calculated. In some occasions however, the accelerations need to be known. Additional calculations are required then.

There are three ways to calculate the accelerations. The first method is by using the equation of motion. Since at the end of the calculations at a time step 2 of the 3 unknowns in the equation are known $(\mathbf{u}_n, \dot{\mathbf{u}}_n)$ the third one $\ddot{\mathbf{u}}_n$ can be calculated using the equilibrium equations.

$$\mathbf{M}\ddot{\mathbf{u}}_n = -\mathbf{C}\dot{\mathbf{u}}_n - \mathbf{f}^{\text{int}}(\mathbf{u}) + \mathbf{f}^{\text{ext}}(t_n) \tag{4.72}$$

This procedure is rather time consuming, since an extra system of equations needs to be solved. When the mass matrix is semi-definite, which is possible in some cases, this equation cannot be solved at all.

In order to reduce the amount of calculations, the acceleration can be calculated just in the first time step $t = t_0$. The subsequent accelerations can be determined by extrapolation. This method is indeed less laborious but also rather inaccurate, especially after a large number of time steps.

The third method that is as accurate as Jensen's procedure and not very complicated is based on the LMS scheme that is used to calculate the velocities. The acceleration vector $\ddot{\mathbf{u}}_n$ can be calculated using a history vector that is based on the velocities in the previous time steps.

$$\ddot{\mathbf{u}}_n = \frac{\dot{\mathbf{u}}_n + \mathbf{h}_n^{\dot{\mathbf{u}}}}{h_\beta} \tag{4.73}$$

This method is fast and the estimated error of this procedure is of the same order as of the calculation of the velocity vector.

4.5 Nonlinear Solution Techniques

The nonlinear equation derived in the previous section cannot be solved directly. Numerical methods should be used instead. In general, two classes of methods for solving nonlinear equations are known, i.e. interpolation methods (method of bisection, false position method) and extrapolation methods (Newton-Raphson method). The first class of methods is most often used for single nonlinear equations since these methods are fast and very simple to implement. System of equations need to be solved using extrapolation methods. In this section we will take a closer look at extrapolation methods, Newton Raphson methods in particular.

4.5.1 Single Degree of Freedom Systems

First the methods are derived for a single degree of freedom system. Doing so, it is possible to explain the process by a simple graph. Later on, the methods are adjusted for a multi degree of freedom system and applied to the current equations.

Newton Raphson Method

An arbitrary nonlinear equation $f(x) = 0$ can be expanded into a Taylor series in an initial guess x^k close to the presumed root x^{k+1}.

$$f(x^{k+1}) = f(x^k) + \Delta x f'(x^k) + \frac{\Delta x^2}{2} f''(x^k) + H.O.T. \tag{4.74}$$

where f' is the derivative of f with respect to x and $\Delta x = x^{k+1} - x^k$. Since x^{k+1} is a presumed root of the function $f(x)$, the equation above can be written as

$$f(x^{k+1}) = 0 \tag{4.75}$$

After dropping the second order (and higher order) terms in the expansion

$$f(x^k) + \Delta x f'(x^k) = 0 \tag{4.76}$$

The increment Δx can be calculated using the linearized equation

$$f'(x^k)\Delta x = f(x^k) \tag{4.77}$$

The new value x^{k+1} can now be calculated

$$x^{k+1} = x^k + \Delta x \tag{4.78}$$

The iteration must start at a appropriate point, $x^{(0)}$ close enough to the root[5]. There are many methods to determine an appropriate initial guess (or predictor). The iteration process must be continued until the residue of the function $f(x^k)$ approaches zero, or when the increment Δx approaches zero.

Modified Newton Raphson

The method described above is often called *Full Newton Raphson*: the tangent of the equation, $f'(x^k)$, will be adjusted each iteration step. In some cases when the tangent is assumed to change very slowly, it is possible to use the old tangent $f'(x^0)$ for every subsequent iteration step.

$$x^{(k+1)} = x^k - \frac{f(x^k)}{f'(x^0)} \qquad \forall \quad k > 0 \tag{4.79}$$

It is not surprising that in this case it can take more iteration steps before a converged solution has been found. On the other hand, the tangent $f'(x)$ only has to be evaluated once in the iteration process. This aspect will be much more important when these methods are applied to large systems of equations.

4.5.2 System of Equations

The analysis described in the previous section can easily be extended to a system of N equations. Consider a nonlinear function $\mathbf{f}$.

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \qquad \mathbb{R}_N \rightarrow \mathbb{R}_N \tag{4.80}$$

For every single equation $f_1, f_2 \ldots f_N$ a Taylor expansion series can be created.

$$\begin{aligned} f_1(\mathbf{x}^{k+1}) &= f_1(\mathbf{x}^k) + \Delta x_1 \frac{\partial f_1(\mathbf{x}^k)}{\partial x_1} + \Delta x_2 \frac{\partial f_1(\mathbf{x}^k)}{\partial x_2} + \ldots + \Delta x_N \frac{\partial f_1(\mathbf{x}^k)}{\partial x_N} + H.O.T. \\ f_2(\mathbf{x}^{k+1}) &= f_2(\mathbf{x}^k) + \Delta x_1 \frac{\partial f_2(\mathbf{x}^k)}{\partial x_1} + \Delta x_2 \frac{\partial f_2(\mathbf{x}^k)}{\partial x_2} + \ldots + \Delta x_N \frac{\partial f_2(\mathbf{x}^k)}{\partial x_N} + H.O.T. \\ f_N(\mathbf{x}^{k+1}) &= f_N(\mathbf{x}^k) + \Delta x_1 \frac{\partial f_N(\mathbf{x}^k)}{\partial x_1} + \Delta x_2 \frac{\partial f_N(\mathbf{x}^k)}{\partial x_2} + \ldots + \Delta x_N \frac{\partial f_N(\mathbf{x}^k)}{\partial x_N} + H.O.T. \end{aligned} \tag{4.81}$$

This equation can be written in a matrix form

$$\mathbf{f}(\mathbf{x}^{k+1}) = \mathbf{f}(\mathbf{x}^k) + \mathbf{H}\Delta\mathbf{x} + H.O.T. \tag{4.82}$$

[5]It may be clear that this procedure will always find a root of the function. However, nonlinear functions may have multiple roots, so that the right root is found as long as the initial guess is close enough.

where $\mathbf{H}$ is the Jacobian of the function $\mathbf{f}(\mathbf{x})$ and $\Delta\mathbf{x}$ is the incremental solution vector, $\Delta\mathbf{x} = [\Delta x_1, \Delta x_2, \ldots, \Delta x_n]^t$. The sequel of derivation is similar to the single degree of freedom equation. The increment can be calculated using the following formula

$$\mathbf{H}\Delta\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{4.83}$$

Of course again a distinction can be made between the full and the modified Newton-Raphson method. In the first case the Jacobian $\mathbf{H}(\mathbf{x})$ changes at every iteration step while in case of the modified method the Jacobian is constant during the process. Before judging the performances of both these methods, it is important to know a little about the numerical methods that are used to solve system of equations.

All methods to solve the linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ are based on the same principle. First the matrix $\mathbf{A}$ is decomposed (translated) into a number of factorized matrices. The solution $\mathbf{x}$ can then be calculated directly using these factorized matrices and the vector $\mathbf{b}$. For example, the solver that is used in the B2000 platform, uses a LDLT decomposition. The matrix is decomposed in a triangular matrix $\mathbf{L}$ and a diagonal matrix $\mathbf{D}$.

These decompositions of the matrix are by far the most laborious part of the process. They can use up to 90% of the total solving time. However, as long as the matrix remains unchanged, there is no need to decompose it again: the old factorization matrices can still be used.

These considerations can be projected on the Newton Raphson methods. When the full Newton method is used, the Jacobian must be constructed and decomposed every iteration step. And although this method needs less steps to converge, the gain of speed is immediately lost due to decomposing time. In 'ordinary' nonlinear equations of motion, the nonlinearities remain small and the response of the structure is a rather smooth function. The modified method is in this case much faster than the full Newton method. However, there are cases that the full Newton method is preferred. For instance when there are contact mechanisms used in the model: the response can be less smooth and the full Newton method can be the only method that finds a converged solution.

4.6 Implementation in the Nonlinear Jensen Equation

The implementation of the iteration methods in the nonlinear Jensen equation requires some additional investigations. First of all a suitable prediction method must be sought. Furthermore, since the algorithm must be able to handle large rotations, the correct update of the rotation vectors must also be taken into account. Finally a suitable convergence criterium must be defined.

4.6.1 Prediction

The first step in the solution procedure is the determination of the predictor, a point from which the actual iteration process can be started. There is just one condition to this predictor: it needs to be in the neighborhood of the converged

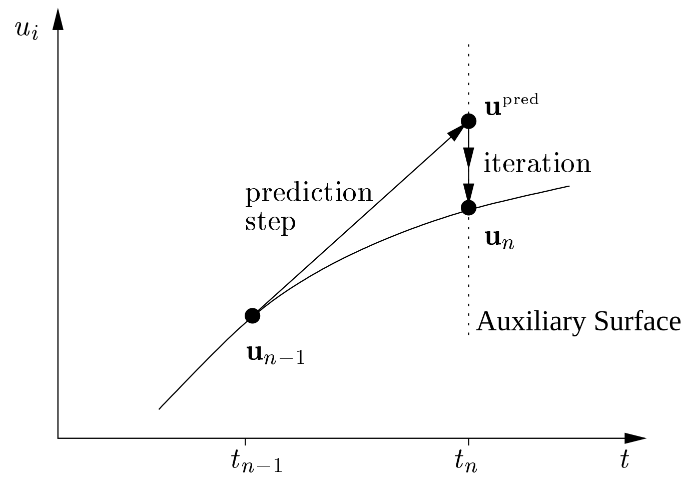


Figure 4.8: Prediction step and iteration process

solution. Although this requirement is rather vague (the solution is not known beforehand), it can be used in this case.

The solution of the Jensen equation $(\mathbf{u}_n; t)$ can be represented in a $\mathbb{R}_{N+1}$ space: N degrees of freedom plus the time parameter t. The time parameter t_n of the new converged solution $(\mathbf{u}_n; t_n)$ is already known since this is the described parameter. The condition stated above can be satisfied partly when the predictor is on the auxiliary $\mathbb{R}_N$ surface at t_n[6], see figure (4.8). In principle, there is a number of methods available to calculate the predictor. In this case just two of these will be discussed.

The Euler method is a so-called *explicit* method. The predictor is calculated regardless of the current state of the structure. Consider that the response of the structure is known up to $n-1^{th}$ step at $t = t_{n-1}$. In this point both $\mathbf{u}_{n-1}$ and $\dot{\mathbf{u}}_{n-1}$ are known. With these solutions and the time-step $h = t_n - t_{n-1}$ the next solution can be guessed using the following formula:

$$\mathbf{u}_n^{\text{pred}} = \mathbf{u}_{n-1} + h\dot{\mathbf{u}}_{n-1} \tag{4.84}$$

Note that the predictor can be calculated without solving a complete system of equations.

Another way to calculate the predictor is using the original Newton Raphson iteration procedure. In this *implicit* method, the predictor is calculated using the original Jensen equation and the previous state variables $\mathbf{u}_{n-1}$, $\mathbf{f}_{n-1}^{\text{int}}$ and $\mathbf{K}_{n-1}$. When the predictor $\mathbf{u}_n^{\text{pred}}$ is known, the iteration procedure can be continued using either the full or the modified Newton Raphson method. In this case the actual predictor is of course the previous solution $\mathbf{u}_{n-1}$.

It is obvious that out of these two methods, the Euler prediction method is by far the fastest one. However, when it comes to accuracy, the Newton method

[6]This approach has many resemblances to the nonlinear iteration procedure as implemented in the continuation macro-processor B2CONT. This processor is based on Riks' path-following technique and will be discussed in chapter 5.

for calculating the predictor is preferred. Therefore, both methods have been implemented in the transient processor.

4.6.2 Jensen Equation

Once the predictor has been calculated, the Newton iteration procedure can be started. The first thing to do is the calculation of the Jacobian of the nonlinear Jensen equation, recalling

$$(1+\alpha h_\beta)\mathbf{M}\mathbf{u}_n + h_\beta(\beta + h_\beta)\mathbf{f}^{\text{int}}(\mathbf{u}_n) = \mathbf{g}_n - \mathbf{g}_n^p \tag{4.55}$$

Since $\frac{\partial \mathbf{f}^{\text{int}}(\mathbf{u})}{\partial \mathbf{u}} = \mathbf{K}(\mathbf{u})$, the Jacobian is equal to

$$\mathbf{H} = (1+\alpha h_\beta)\mathbf{M} + h_\beta(h_\beta + \beta)\mathbf{K}(\mathbf{u}_n) \tag{4.85}$$

In terms of the Newton Raphson algorithm, the complete equation will look like

$$\mathbf{H}\Delta\mathbf{u} = \mathbf{r} \tag{4.86}$$

where the right-hand-side vector $\mathbf{r}$ (or residue vector), is the nonlinear Jensen equation (4.55).

4.6.3 Compound Rotations

The incremental solution $\Delta\mathbf{u}$ obtained by equation (4.86) must be added to the previous solution in the iteration process $\mathbf{u}_n^{(i-1)}$, according to

$$\mathbf{u}^{(i-1)} + \Delta\mathbf{u} = \mathbf{u}_n^{(i)} \tag{4.87}$$

The previous displacement $\mathbf{u}_n^{(i-1)}$ as well as the incremental array $\Delta\mathbf{u}_n$ consist of displacement and rotation increments. The 3 dimensional beam element as well as the Simo type shell elements developed by G. Rebel [23] are able to obtain large rotations. A correct update of these rotations is required. These incremental rotations may not be added to the full rotation. A compound rotation tensor must be used instead.

In section 3.2 the formulation of large rotations has been discussed as well as the principle of compound rotations. In the derivation of the compound rotation tensor, the following alternative rotation vector is used.

$$\hat{\boldsymbol{\theta}}_1 = \frac{2\sin\frac{1}{2}||\boldsymbol{\theta}_1||}{||\boldsymbol{\theta}_1||}\boldsymbol{\theta}_1; \qquad \hat{\boldsymbol{\theta}}_2 = \frac{2\sin\frac{1}{2}||\boldsymbol{\theta}_2||}{||\boldsymbol{\theta}_2||}\boldsymbol{\theta}_2 \tag{4.88}$$

It is shown by Crisfield [3] that after algebraic manipulation the compound rotation vector is equal to

$$\hat{\boldsymbol{\theta}}_{12} = \pm\left[(1-\frac{1}{4}||\hat{\boldsymbol{\theta}}_2||)^{1/2}\hat{\boldsymbol{\theta}}_1 + (1-\frac{1}{4}||\hat{\boldsymbol{\theta}}_1||)^{1/2}\hat{\boldsymbol{\theta}}_2 - \frac{1}{2}\hat{\boldsymbol{\theta}}_1\times\hat{\boldsymbol{\theta}}_2\right] \tag{4.89}$$

The correct $\pm$ sign follows from additional calculations [2]. In terms of the Jensen equation, the rotation parts of the previous solution $\mathbf{u}^{(i-1)}$ can be denoted as $\hat{\boldsymbol{\theta}}_1$ and the rotational terms of the incremental vector $\Delta\mathbf{u}$ as $\hat{\boldsymbol{\theta}}_2$. The rotational components of the updated solution $\mathbf{u}_n^{(i)}$ are equal to $\hat{\boldsymbol{\theta}}_{12}$[7].

The history vectors also contain summations of finite rotation components. In this derivation, however, these terms are not added properly using compound rotations. The resulting error can be neglected as long as the total rotations remain moderately small, say in the order of $\theta < 1$ rad. This implies that the capacities of the new beam elements as well as Rebel's shell elements are not fully utilized.

4.6.4 Converge Criteria

The sequence of iterations on the Newton Raphson process must be continued until the numerical solution of the Jensen equation in step n, $\mathbf{u}_n^k$, is close enough to the exact solution $\mathbf{u}_n$. Close enough is defined by the following convergence criterion

$$||\Delta\mathbf{u}|| < \epsilon_{\text{disp}}||\mathbf{u}_{n-1}|| \tag{4.90}$$

and

$$||\mathbf{r}_n|| < \epsilon_{\text{res}}||\bar{\mathbf{f}}_n|| \tag{4.91}$$

In the first equation is $||\mathbf{u}_{n-1}||$ the norm of the converged displacement vector in the previous time-step. In the second one is $||\bar{\mathbf{f}}_n||$ a force vector with the following value

$$||\bar{\mathbf{f}}_n|| = \begin{cases} ||\mathbf{f}^{\text{ext}}(t_n)|| & \text{if} \quad ||\mathbf{f}^{\text{ext}}(t_n)|| > 0 \\ ||\mathbf{f}^{\text{int}}(\mathbf{u}_{n-1})|| & \text{if} \quad ||\mathbf{f}^{\text{ext}}(t_n)|| = 0 \end{cases} \tag{4.92}$$

If all these vectors are null-vectors, for instance in the first step of the time integration process at $t = t_1$, an arbitrary value can be used instead. The terms ϵ_{disp} and ϵ_{res} are the so-called error factors. Normally both factors are chosen very small (in the order of 10^{-3} and smaller).

4.6.5 Time Step Control

As long as the finite element model is defined correctly and the time step is small enough to calculate at least the basetones of the structure, the iterative Newton procedure is able to obtain converged solutions without any difficulties. In some occasions however, there can be problems finding a converged solutions. The only variable that can be changed, without losing accuracy is the time-step h. A smaller time-step can lead to faster convergence. This procedure is often called *time-step cutting*. It must be done in two cases

[7]This algorithm is also implemented in the continuation macro-processor B2CONT by G. Rebel.

- The number of iterations exceeds a predefined number of maximum iterations, i.e. it takes too long before a converged solution is obtained.
- The Jacobian, **H** which is equivalent to the dynamic stiffness matrix **E**, equation (4.50), is no longer positive definite (negative roots occur on the diagonal of **H**).

By the authors knowledge, there is no such method that can be used to calculate a new, optimal time-step by evaluating the current iteration process. An ad-hoc approach is used instead.

The problem with cutting the time step in this case (time integration using Park's LMS scheme) is that after each cut, the procedure must be restarted with the less optimal 1- and 2-step method. It is therefore important to be careful in cutting the time-step. Cutting methods that adapt the time step constantly (as is implemented in the continuation procedure) are not suitable in this application. A less complicated procedure is used instead, which just splits the time-step in the critical cases mentioned above.

$$h_{\text{new}} = \frac{1}{2} h_{\text{old}} \tag{4.93}$$

It can be possible that after a while, the number of iterations per time-steps decreases. In that case the time-step can be re-enlarged. As an upper bound for the maximum time-step h_{init}, the initial time-step can be used.

4.7 Closure

The implicit time integration method as described in this chapter is implemented in the `B2000` platform as a new macro-processor `B2TRANS`. It can be used in combination with all other macro-processors like `B2LIN`, `B2BUCK` and `B2CONT`. A number of numerical examples to prove the reliability of this new macro-processor is presented in chapter 6. The users manual is given in section A.2. In appendix B a short description of the syntaxis is igven. Also the implementation of existing processors such as the nonlinear element processor `B2EPN` is discussed. A summary of all source fiules can be found in appendix C.

5 The Mode-jumping Phenomenon

The term *mode jumping* is often used to describe sudden dynamic changes in a static process. In computational mechanics, the abrupt change in wave numbers in a buckling process is indicated as a mode jump. This phenomenon was first mentioned by M. Stein [31] when describing a buckling experiment on a flat panel. A specimen as shown in figure 5.1a was loaded with an end shortening, in order to determine its limit load. However, after it had buckled, Stein noticed changes (jumps) in the wave number from 5 via 6 and 7 to 8 half waves. The load versus end-shortening diagram was of the type shown in figure 5.1b. The jumps in the figure are represented by the vertical lines where the internal force decreases for equal end-shortening.

The observed phenomena were analyzed and interpreted by several people in the years after. In their opinion, the answer to this problem should be sought in the stability of the buckling modes. In general when a structure buckles, the equilibrium state can become *unstable*[1]. When the structure enters such an unstable part of this post-buckling trajectory it will move away to the nearest *stable* equilibrium. This motion, the jump from one mode to another, attends large velocities and accelerations and is therefore a transient, dynamic process.

In the years that followed many people tried to simulate such jumps. In FEM analysis this can be done with a transient solving routine, as developed in the previous section. The pre-buckling state, before the unstable configuration has been reached, can be calculated with a path-following technique. At this stage, the velocities and accelerations are neglected. Just before the unstable path has will be entered, the calculations can be continued with the transient processor. When a new stable static equilibrium is found,i.e. when the structure

[1]The term stability has a different meaning in this context. In the previous chapter, the word stability was used to describe the ability of a time integration method to produce converged results over a number of time-steps. In this case, stability is used in combination with the ability of the structure to remain its current deformation. A more precise explanation will be given in section (5.1.3).

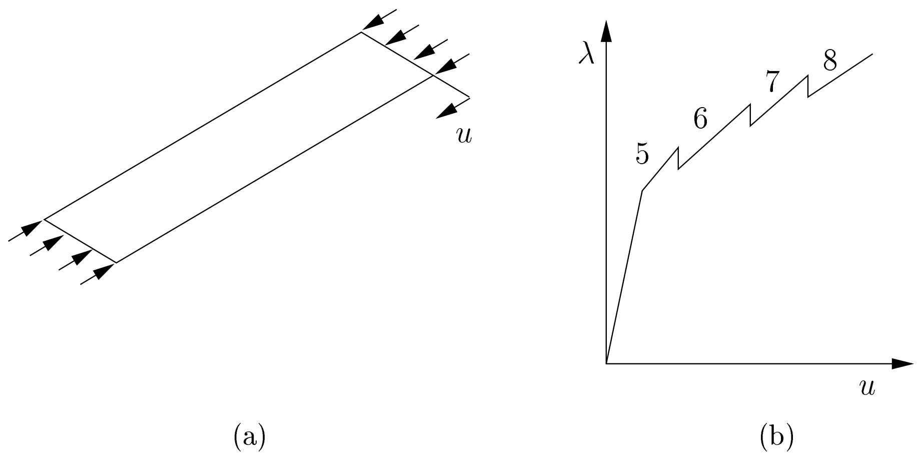


Figure 5.1: Stein's buckling experiment and load displacement relation.

is at rest, the path-following method can be restarted.

In this chapter a closer look will be taken at the mode-jumping phenomenon in general and the numerical aspects of the process in particular. The papers by Riks *et al.* [25, 27, 28] are used in these considerations. Apart from the transient part of the solution, the quasi-static part of the calculations is examined as well as the concept of loss of stability.

5.1 The Quasi-Static Solution

When the loads are applied slowly to the structure, the equations governing the equilibrium of motion, equation (4.1), can be reduced to a system of static equilibrium equations by omitting the inertia and the damping forces. This is allowed since in a slow deformation process the accelerations and velocities are assumed to be small.

$$\mathbf{f}(\mathbf{u};\lambda) = \mathbf{0} \tag{5.1}$$

In this equation the load factor is denoted by λ, which determines the magnitude of the external forces or prescribed displacements. Again, a distinction can be made between the internal and the external forces

$$\mathbf{f}(\mathbf{u};\lambda) = \mathbf{f}^{\text{int}}(\mathbf{u}) - \mathbf{f}^{\text{ext}}(\lambda) = \mathbf{0} \tag{5.2}$$

The static equilibrium equation describes the deformation of a structure under an arbitrary load. Due to the geometric nonlinearities, deformations due to large load factors can not be calculated in one step, but have to be determined in more steps, with increasing load factors. Doing so, a one dimensional curve in the space $\mathbb{R}_{N+1}$ spanned by $\mathbf{u}$ and λ can be obtained, figure 5.2.

This load-displacement curve seems to be a history plot of a deformation process, which of course it is not[2]. It is just a collection of static equilibrium

[2]The load displacement curve approaches a time history plot when the specific time is said to be infinite, for example when the loads are applied to the structure very slowly.

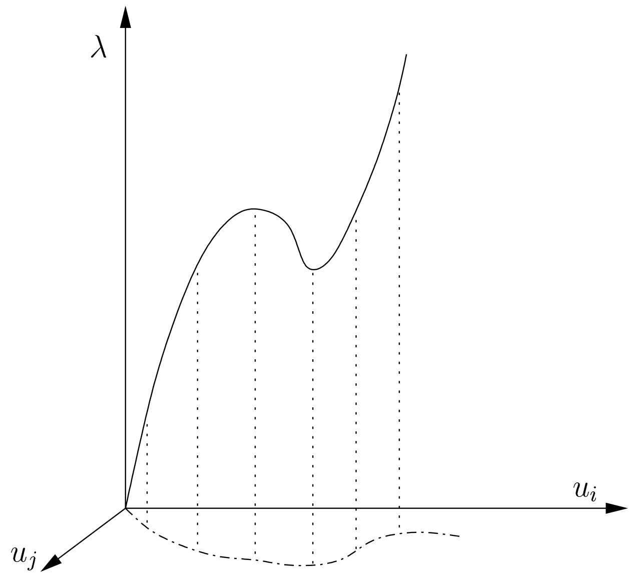


Figure 5.2: An example of a load-displacement curve in the $\mathbb{R}_{N+1}$ space

points connected by a line. However, because of its resemblance to a time plot, this solution is often called a *quasi-static response* of a structure.

The nonlinear equilibrium equation (5.1) is a system of N equations, where N is the total number of degrees of freedom of the discrete structure. Unfortunately, there are $N + 1$ unknowns. These are the N-dimensional displacement vector $\mathbf{u}$ and the scalar load-factor λ. The system is therefore undetermined. It can only be solved when the number of unknowns is equal to the number of equations.

One way to solve the system of equations is to reduce the number of unknowns to N by selecting one unknown as a constant or the so-called *path parameter*. It is possible to choose either the load-factor λ or one element in the displacement vector $\mathbf{u}$. In the first case, λ is used as a prescribed variable (denoted by the path parameter η) which is held constant at its value in the iteration process in which the corresponding displacements are calculated. This procedure is better known as the incremental load procedure. In the second case, the displacements fulfill the role of path parameter and is therefore called an incremental displacement procedure. From figure (5.3) can be concluded that both procedures can have difficulties. The incremental load procedure cannot pass the limit point A, the incremental displacement method will not pass the turning point B.

5.1.1 Riks' Path Following Technique

Instead of reducing the number of unknowns to N, the number of equations can be increased to $N+1$ by adding a new equation $f^* : \mathbb{R}_{N+1} \rightarrow \mathbb{R}_1$ to the system (5.1). A new path parameter η is also included in this equation. The

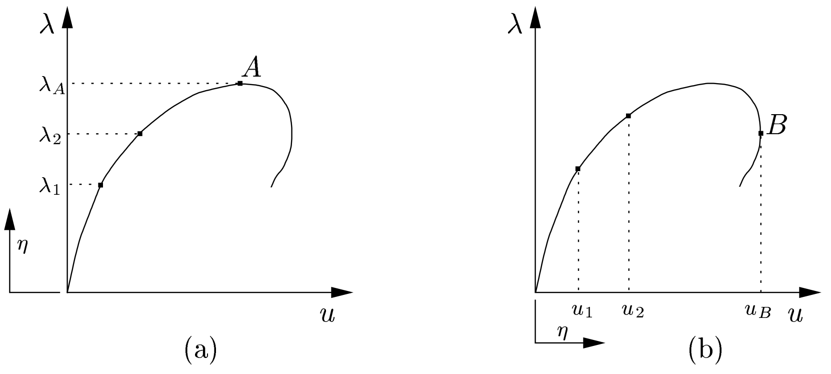


Figure 5.3: The incremental load procedure (a) and the incremental displacement procedure (b)

new, augmented system of equations can be written as

$$\hat{\mathbf{f}} = \begin{bmatrix} \mathbf{f}(\mathbf{u},\lambda) \\ f^*(\mathbf{u},\lambda,\eta) \end{bmatrix} = \mathbf{0} \qquad \hat{\mathbf{f}} : \mathbb{R}_{N+1} \rightarrow \mathbb{R}_{N+1} \tag{5.3}$$

The new equation can be chosen freely, as long as it is independent of all equations in (5.1).

The first and by far the most simple choice that will be discussed is the load parameter λ as path parameter η. The equation f^* can be written as

$$f^* = \eta - \eta_{n-1} = \lambda - \lambda_{n-1} = 0 \tag{5.4}$$

where λ_{n-1} is the load factor of the converged solution of the previous time step. It needs no proof that this equation is independent of the system of equations. In principle, this method is equal to the incremental load procedure as described before and can therefore not be used to evaluate limit points. It may be clear that there is also an alternative function f^* that is related to the incremental displacement procedure.

Another choice for the additional function is based on the principle of adaptive parameterization. Instead of using either the load factor or a displacement as the parameter, it can be useful to adapt the parameter to the properties of the solution path. A very primitive form of adaptive parameterization is a so-called mixed procedure, figure 5.4. In this procedure the path parameter switches between the load factor λ and a displacement u, depending on the direction of the path.

The path parameter can also be chosen in such a way that it is at any point tangential to the path. Riks [25] derived the following additional equation

$$f^* = \mathbf{n}^T(\mathbf{u} - \mathbf{u}_{n-1}) - \eta = 0 \tag{5.5}$$

where $\mathbf{n}^T$ is the so-called base-vector, which denotes the direction in which the new equilibrium $(\mathbf{u}, \lambda)$ is sought. There are no restrictions to this vector whatsoever.

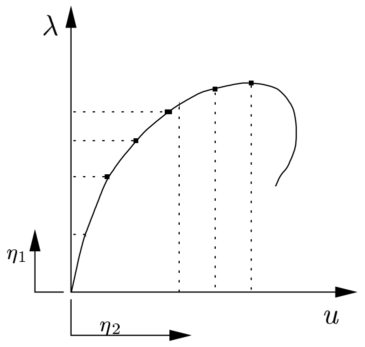


Figure 5.4: Mixed parameter procedure

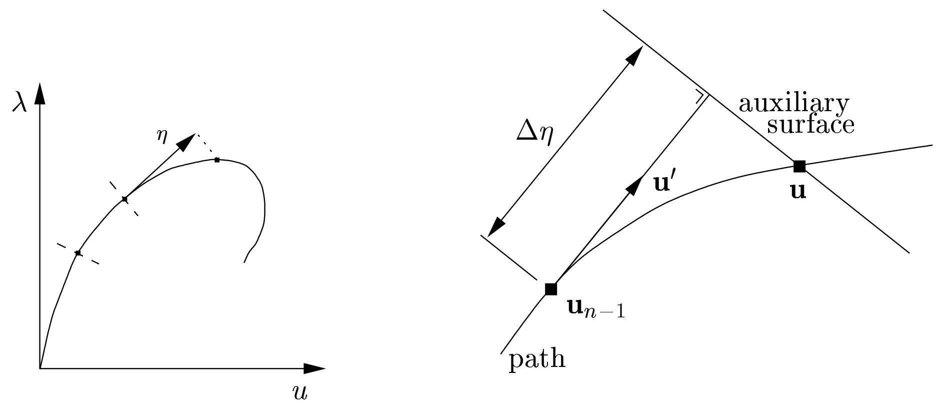


Figure 5.5: Arc-length procedure

It may be clear that this method produces the best results when the auxiliary surface is perpendicular to the path. This means that the director in equation (5.5) must be tangential to the path.

$$\mathbf{n} = \mathbf{u}_1' = \frac{d\mathbf{u}(\eta)}{d\eta}|_{\eta=\eta_1} \tag{5.6}$$

The additional function can than be written as

$$f^* = \frac{d\mathbf{u}(\eta)}{d\eta}|_{\eta=\eta_1}(\mathbf{u} - \mathbf{u}_{n-1}) - \eta = h(\mathbf{u}, \lambda) - \eta = 0 \tag{5.7}$$

where $h(\mathbf{u}, \lambda)$ is the so-called function for the auxiliary surface. The solution of the augmented system as presented in equation (5.3) is on this surface and can be found iteratively by using nonlinear solution techniques.

5.1.2 Possible Solutions

Although the path can be calculated completely using this technique, it is still important to discuss the properties of the solution of equation (5.4). In general,

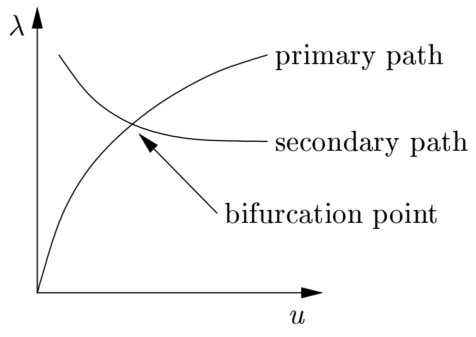


Figure 5.6: A bifurcation point, a crossing of two equilibrium paths

there are two types of points on the path with very special geometrical properties that need extra attention. The first one, which has already been mentioned, is the limit point, point A in figure 5.3a. This singularity is solved by the introduction of the additional equation. The other group of special points is formed by bifurcation points. A bifurcation point is a point on the primary path that is also an equilibrium of a different, secondary path, figure 5.6. By definition bifurcation points always have loss of stability as a result.

Limit Points

On a limit point (or stationary point), the director of the path parameter is horizontal, or $\lambda'(\eta) = 0$. The load parameter has reached a maximum[3]. When the path crosses such a limit point, the structure *can* loose stability. If so, these points are called proper limit points in the sense of the stability theory. The mathematical formulation of stability will be explained in the next section.

The loss of stability of limit points in reality can be explained as follows. Consider a structure , for example a cylindrical shell, which is clamped at one end and compressed by a uniform load on the other end. At a certain moment the exact limit point is reached. In this configuration $(\mathbf{u}_{\text{lim}}; \lambda_{\text{lim}})$ the path can only be continued when the applied load is decreased. But, decreasing the load factor will imply a relaxation of the shell and the path will be followed backwards, in the wrong direction. The path cannot be continued in reality, which indicates the presence of instability.

Bifurcation Points

As opposed limit points, the crossing of bifurcation points is not shown be the load-displacement curve. The steps are normally too big and the direction of the path is preset by the previous equilibriums. Analysis of the the condition of the stiffness matrix must give a definite answer to the question how much bifurcation points have been passed.

[3]The director is also horizontal in a minimum of a saddle-point. Although these points also occur in stability analyses, they will not be discussed here.

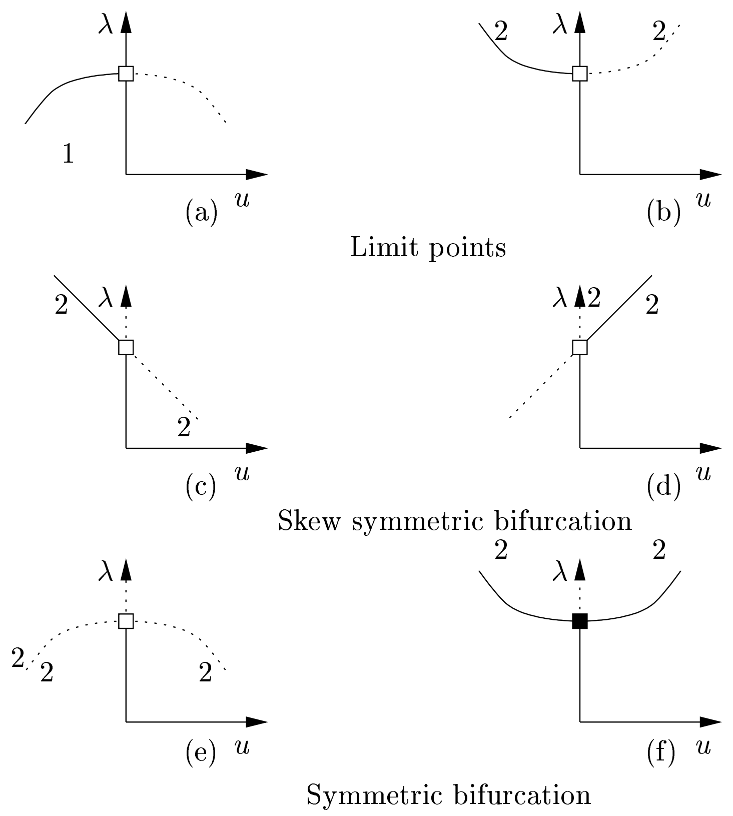


Figure 5.7: Elementary forms of loss of stability

At first sight, there are not many similarities between limit points and bifurcation points, but this is wrong. In reality, bifurcation points always occur as limit points. Due to initial imperfections in either the material boundary conditions or the geometry of the structure. The actual path will not be exact on the mathematical equilibrium path. Near a bifurcation point, the real path can also be attracted by the secondary path. In most case the path will follow this path, as can be seen in figure 5.8, with a limit point as a result. This principle is also used in FEM analyses to visualize bifurcation points. The initial imperfections are often simulated by small nodal forces.

5.1.3 Stability Analysis

As said in the previous section: the buckling path can become unstable after passing a limit point or bifurcation point. But what is stability? The answer to this question will be given in this section according to two criteria, the energy criterion and the slightly different Lyapunov criterion.

The energy criterion states that if the potential energy of the structure at the equilibrium state is a proper minimum, the equilibrium is stable. If this property does not exist, the equilibrium is unstable. More specific, a given equilibrium state $(\mathbf{u};\lambda)$ is stable if and only if the potential energy $P(\mathbf{u};\lambda)$

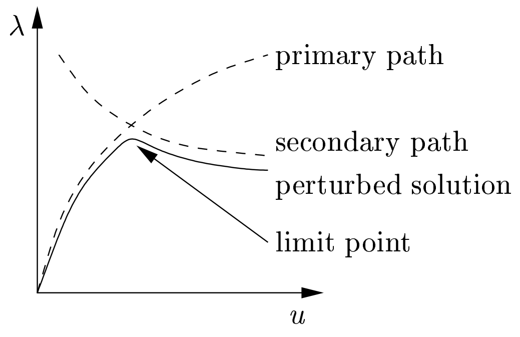


Figure 5.8: Limit point of a perturbed structure

satisfies

$$P(\mathbf{u}+\delta\mathbf{u};\lambda)-P(\mathbf{u};\lambda)>0 \qquad \forall\delta\mathbf{u}\in\mathbb{R}_N \tag{5.8}$$

The incremental displacement is assumed to be finite but small, $0<||\delta\mathbf{u}||<\epsilon^2$. If the inequality (5.8) is not satisfied, the equilibrium configuration $(\mathbf{u};\lambda)$ is unstable. The energy criterion can be expanded as a Taylor series, assuming that the equation of the potential energy can be differentiated infinitely.

$$P(\mathbf{u}+\delta\mathbf{u};\lambda)-P(\mathbf{u};\lambda)=\frac{\partial P(\mathbf{u};\lambda)}{\partial\mathbf{u}}\delta\mathbf{u}+\frac{1}{2}\frac{\partial P^2(\mathbf{u};\lambda)}{\partial\mathbf{u}\partial\mathbf{u}}\delta\mathbf{u}\delta\mathbf{u}+H.O.T. \tag{5.9}$$

Since $(\mathbf{u};\lambda)$ is an equilibrium state, the first term is equal to zero. The leading term becomes therefore the second variation of the total energy with respect to $\delta\mathbf{u}$. As derived in chapter 2, the total energy differentiated to the displacement twice is equal to the stiffness of the structure.

$$\frac{\partial P^2(\mathbf{u};\lambda)}{\partial\mathbf{u}\partial\mathbf{u}}=\mathbf{K}(\mathbf{u};\lambda) \tag{5.10}$$

The second variation of the total energy is therefore

$$\Pi_2(\delta\mathbf{u})=\frac{1}{2}\frac{\partial P^2(\mathbf{u};\lambda)}{\partial\mathbf{u}\partial\mathbf{u}}\delta\mathbf{u}\delta\mathbf{u}=\frac{1}{2}\delta\mathbf{u}^t\mathbf{K}(\mathbf{u};\lambda)\delta\mathbf{u} \tag{5.11}$$

It can now be shown that the minimum of $P(\mathbf{u};\lambda)$ and thus stability of the configuration $(\mathbf{u};\lambda)$ is ensured if the second variation of the potential energy is positive definite:

$$\Pi_2(\delta\mathbf{u})>0 \tag{5.12}$$

This equation implies that for *all* possible perturbations $\delta\mathbf{u}$ the second variation is >0. The quadratic form is called indefinite if there are some perturbation vectors thinkable for which the second variation is smaller than 0, i.e.

$$\Pi_2(\delta\mathbf{u})\lesseqgtr 0. \tag{5.13}$$

the quadratic form is called indefinite. The potential energy is not a minimum and the equilibrium is unstable. A borderline case occurs when the quadratic form is semi-positive definite, when for some $\delta\mathbf{u}$ the second variation is equal to 0.

$$\Pi_2(\delta\mathbf{u}) \geq 0 \tag{5.14}$$

Limit and bifurcation points where the path changes from stable into unstable are characterized by these points. Following criterion (5.12) in combination with equation (5.10) the potential energy is a proper minimum if the stiffness matrix $\mathbf{K}$ is positive definite. In practice, the stiffness matrix does not have any negative roots on the diagonal of the factored matrix. Since in the path-following method the stiffness matrix needs to be factored each step, the stability can be evaluated by checking the condition of the factorized stiffness matrix.

Lyapunov's criterion is equivalent to the energy criterion, but since it also refers to the dynamic behavior of the structure, it is worth while mentioning it. Consider a deformed structure at rest that is disturbed by a small perturbation. If the incipient motion grows without bounds, independent how small the initial disturbances are taken, the equilibrium is declared unstable. When for these small perturbations the motion remains bounded or damped, the equilibrium is called stable. Apparently, when an equilibrium is unstable it will move away from this equilibrium coupled with large accelerations and velocities. This behavior can be simulated with the transient algorithm developed in the previous chapter.

5.2 The Transient Solution

When the solutions of the path in the static domain have become unstable, after crossing a limit- or bifurcation point, the analysis must be continued using the transient solver in order to calculate the jump to a new stable path. One of the most important aspects is the correct initialization of the transient process. The choice of the initial conditions determine the direction in which the structure will jump. Control parameters such as the time step h and the amount of damping are other important features that make sure that the jump ends at the correct stable path in a proper way.

5.2.1 Initial Values

The critical points where mode-jumping occurs are either limit points or unstable bifurcation points, where the equilibrium on the path changes from stable to unstable. In figure (5.9) this transition from the stable trajectory occurs at point A. The unstable path is denoted with a dashed line.

In principle, these critical points are not known beforehand. A limit point is marked by a maximum in the load displacement curve and a change in the condition of the stiffness matrix $\mathbf{K}$. Bifurcation points can only be identified by evaluating the stiffness matrix at every step. In the continuing the mode-jumping of limit points will be discussed, since in practice, pure bifurcations

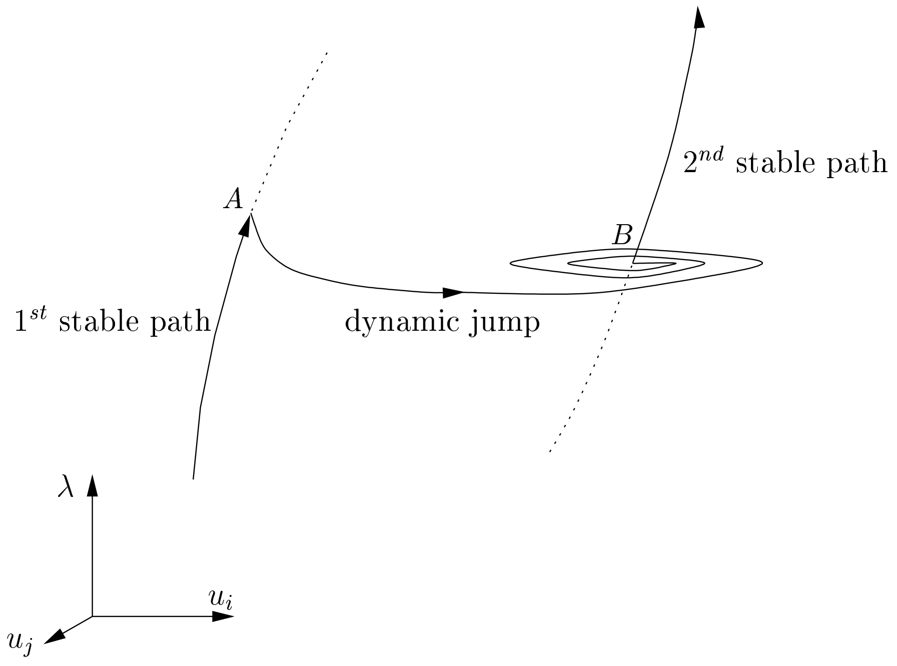


Figure 5.9: The mode-jumping process

are a rarity and can always be changed into a limit point singularity by adding an imperfection to the model.

The transient process is affected by three variables, i.e. the initial conditions $\mathbf{u}_0$ and $\dot{\mathbf{u}}_0$ and the load factor λ that is now time dependent and determines the value of either external loads $\mathbf{f}^{\text{ext}}$ or the prescribed displacements.

The first and most natural way to start the transient process is by using the last stable equilibrium on the primary path $(\mathbf{u}_s, \lambda_s)$. The subscript s means that this is a solution of the quasi-static equation. The initial displacement of the transient process at $t = t_0$ can be chosen equal to the last stable solution: $\mathbf{u}_d(t_0) = \mathbf{u}_s$. The load factor can be prescribed in such a way that it starts at the level λ_s at $t = t_0$ and increases monotonously to a level λ_d according to $\lambda(t) = \lambda_s + \frac{d\lambda}{dt}t$. In this case $\frac{d\lambda}{dt}$ is the increase of the load-factor per unit of time. Within the time domain in which the complete jump must proceed, the load-factor must exceed the limit load of this stable path.

$$\lambda_d(t) > \lambda_{\text{limit}} \qquad t < t_{\text{end}} \tag{5.15}$$

Since the jump will start from a static equilibrium, the initial velocity $\dot{\mathbf{u}}_0$ can be chosen equal to $\mathbf{0}$.

As long as the load factor is smaller than λ_{lim}, the transient analysis will behave like a quasi static analysis: the velocities and accelerations remain small. As soon as λ exceeds the limit load, the character of the process becomes more and more dynamic. The kinetic energy that was almost equal to zero increases and the structure moves away from the static branch rapidly. When the new stable path has been reached, the kinetic energy will dissipate and the structure will come at rest again. From here the continuation analysis can be restarted.

The method described above is undoubtedly the best way to simulate the jump, but not the most convenient one. Since the exact value of the limit load λ_{lim} is not known, it can take a long time before the load factor passes the limit value and the actual jump begins. The calculations that have been done so far can be considered wasted. Up to the limit point, the much faster continuation method can be used instead.

A jump can also be initiated using the Lyapunov's criterion. This criterion states that when an unstable solution is perturbated, the jump will be initiated. It can be shown that it is not possible to use an unstable static equilibrium to start the transient calculations. However, the perturbation of a stable equilibrium into the unstable area will give the same results. Again the last stable equilibrium $(\mathbf{u}_s, \lambda_s)$ will be used. By changing the load-factor λ, this equilibrium can be perturbed into the unstable area. The only condition is that the $\lambda_d(t)$ is significantly larger than the limit load λ_{lim}. In principle $\lambda_d(t)$ can be kept constant during the jump. This method will produce results very fast. Immediately after starting the transient calculation, the dynamic behavior (the increase of kinetic energy) will be visible.

In some cases, especially in shell structures, the limit or bifurcation point happens to be a knot in which more second branches are involved. In the methods above the direction in which the jump proceeds (which second branch will be followed) is completely arbitrary. In order to have full control in which direction the structure will jump, the need for an improved method arose.

The buckling modes and corresponding critical loads can be calculated by an eigenvalue analysis. In B2000 this analysis is implemented in the macroprocessor B2BUCK. This processor determines the buckling eigenvalues according to the undeformed state, $\mathbf{u} = \mathbf{0}$. It is better to perform this analysis using the last stable solution $\mathbf{u}_s$ as a starting point[4] One of the obtained eigenmodes $\mathbf{a}_j$ can be used to define the initial displacement of the transient analysis. Since the eigenmodes or often presented as normalized vectors, the must be multiplied by a scaling parameter μ first. The initial displacement can thus be written as

$$\mathbf{u}_d(t_0) = \mathbf{u}_s + \mu\mathbf{a} \tag{5.16}$$

The external load parameter $\lambda_d(t)$ must be higher than the corresponding eigenvalue λ_j. In order to give the jump an additional boost in the right direction it is possible to use initial velocities. The velocities are also based on the eigenvector of the corresponding buckling mode, say $\dot{\mathbf{u}}_d(t_0) = \epsilon\mathbf{a}$ where ϵ is a arbitrary small number.

Because the static equilibrium is perturbed, inertia and damping forces (accelerations and velocities) are needed to restore the dynamic equilibrium. This will start the jump motion. Since the deformation on the current path corresponding the load factor λ_d is not stable, the structure will move to another (second) stable path with the equilibrium $(\mathbf{u}_s^2; \lambda_d)$, figure 5.9.

[4]This solution procedure used to be a part of the B2CONT package. For reasons not known by the author, this feature has disappeared from the code. Eigenvalues analyses can only be done using a undeformed structure.

	$\mathbf{u}_d(t_0)$	$\dot{\mathbf{u}}_d(t_0)$	$\lambda(t)$
Classical method	$\mathbf{u}_s$	$\mathbf{0}$	$\lambda_s + \lambda_{,t}\, t$
Fast method	$\mathbf{u}_s$	$\mathbf{0}$	$\lambda_s + \Delta\lambda$
Improved fast method	$\mathbf{u}_s + \mu\mathbf{a}$	$\epsilon\mathbf{a}$	$\lambda_s + \Delta\lambda$

Table 5.1: Initial conditions and load factor for initiation of mode jump

5.2.2 Damping

In principle, the transient analysis of a structure is undamped when the Rayleigh coefficients α and β are equal to zero. Of course there is a small amount of numerical damping introduced by Park's method, but this amount can be neglected, especially when the time-step is small enough. As a consequence of this, the total kinetic energy during the transition will hardly decay. It can take an enormous amount of time steps before the iteration comes to a rest in the new stable branch. Furthermore, the dynamic response can become so violate, that the first stable branch can be missed and the structure reaches a next stable path. This phenomenon is called *overshooting.*

So far, damping must be simulated using Rayleigh damping. In order to obtain a realistic damping behavior of the structure, the constants α and β should be chosen so that the damping is under-damped. Recalling equation (4.28).

$$\alpha + \beta\omega_l^2 = 2\omega_l\xi_l \tag{5.17}$$

This equation describes the connection between the eigenfrequency and damping ratio of the structure and the Rayleigh constants. Unfortunately, the two constants cannot be determined from this single equation. An additional relation must be formulated first. There is a number of methods to do so, 2 of them will be discussed here.

The best way to determine α and β is to describe the relation using another eigenfrequency, for instance, the second dominant frequency. This results in the following system of 2 equations.

$$\begin{aligned} \alpha + \beta\omega_1^2 &= 2\omega_1\xi_1 \\ \alpha + \beta\omega_2^2 &= 2\omega_2\xi_2 \end{aligned} \tag{5.18}$$

In principle, two different damping ratio's ξ_1 and ξ_2 can be inserted in this equation. The best results are obtained when the damping ratio to the first eigenfrequency is smaller than the damping ratio of the second one, i.e. $\xi_1 < \xi_2$, but two identical ratio's can be used as well, $\xi_1 = \xi_2$. In this case, the ratio's are

$$\alpha = \frac{2\xi\omega_1\omega_2}{\omega_1 + \omega_2}; \qquad \beta = \frac{2\xi}{\omega_1 + \omega_2} \tag{5.19}$$

Another way to introduce a second equation is the demand that the influence of both the mass and the stiffness matrix to the damping must be equal, which results in the following system.

$$\begin{aligned} \alpha + \beta\omega^2 &= 2\omega\xi \\ \alpha - \beta\omega^2 &= 0 \end{aligned} \tag{5.20}$$

The answer to this system can be calculated immediately.

$$\alpha = \xi\omega \qquad \beta = \frac{\xi}{\omega} \tag{5.21}$$

The natural frequencies of a structure can be calculated using a linear dynamic eigenvalue analysis method, which has already been implemented in the `B2000` platform as the macro-processor `B2LIN` [16]. This analysis can be done before the buckling analysis, on the undeformed structure. Appropriate values for the damping constant ξ to simulate an under-damped vibration are in the range of $0.05 < \xi < 0.2$.

5.2.3 The Second Stable Path

It can be shown [28] that when damping is applied to the model, the kinetic energy will decrease during the transient process. When the kinetic energy approaches zero, this is an indication that the transient analysis has reached a new static stable path. When the kinetic energy T satisfies the following condition

$$T < \epsilon_T \tag{5.22}$$

the structure can said to be in rest. In reality, the structure is vibrating around the static solution of the second branch, point B in figure (5.9). The choice of the right ϵ_T is an enigma. As a rule of thumb this small constant can be chosen as $\epsilon_T = 10^{-3}T_{max}$ where T_{max} is the maximum kinetic energy during the transition, most likely reached after a few steps in the integration process.

5.2.4 Restarting the Continuation Method

The configuration of the 'construction at rest' obtained by the transient analysis can be used to restart the continuation analysis in order to calculate the next stable path. Since the transient analysis was stopped when the kinetic energy was not exactly equal to zero, there will be small velocities and accelerations. The displacement vector at this time $\mathbf{u}_r^d$ is therefore not a static solution.

The next aim is to find the corresponding static solution on the second path $\mathbf{u}_s^2$. A good restart relies on two things. First the dynamic solution $\mathbf{u}_d^r$ must be close enough to the quasi-static path to serve as a predictor in the continuation analysis. When the dynamic solution is not in the domain of convergence around the static solution $\mathbf{u}_s^2$, convergence can not be guaranteed. Secondly, the static solution must be stable, that is the stiffness matrix $\mathbf{K}(\mathbf{u}_s^2)$ must be positive definite.

As soon as the first point on the secondary path is calculated, the remainder of the path can be followed using the normal continuation routine. When this path becomes instable, the exercise must be repeated from the beginning in order to reach the third path.

5.3 Closure

In this chapter, a strategy to calculate the post buckling behavior of structures is presented. The strategy is completely based on two solution methods. The quasi-static continuation method is used to calculate the behavior of the structure in static stable conditions. The sudden, dynamic jump towards a new static stable equilibrium is calculated with a transient solution method. This last method is based on the nonlinear equations of motion which is the most accurate and complete model for the description of mechanical response of structures.

In principle, the complete behavior of the structure under loading can be calculated using the transient method instead of switching from the continuation method to the transient method and back. When the load factor λ is increased very slowly in the time domain, the transient method approaches the continuation method. This way, it is impossible to find unstable solutions: the transient algorithm will automatically jump to the new 'stable' path and continue its analysis. The post-buckling behavior of a structure will be calculated more naturally.

In spite of these advantages, this 'all-transient' method is not recommended to calculate post-buckling behavior. A number of reasons can be presented for the benefit of the 'mixed method'.

Compared to the transient method, the path-following method (quasi-static solution) is much faster for the computation of static equilibrium deformations. It is possible to walk down the path taking large steps. Regardless of the step size, the calculated path is always exact. Unfortunately it is impossible to take these large steps when the path is calculated using the transient method. The most important reason is that the load-factor λ is a function of time. The progress of time is denoted by the constant step-size h. During the process , the rate of increase of the load factor cannot easily be changed. When the load-factor as a function of time increases too fast, some uncontrollable dynamic phenomena such as overshooting can occur. When the time-step h is too large, the results become less accurate due to an enormous amount of numerical damping.

It is rather difficult to create a finite element model with loading patterns and initial conditions, that will simulate the exact dynamic behavior. Small disturbances in the initial conditions or loading will result in rather large deviations, certainly when many time-steps are required to calculate the dynamic response accurately. The transient method allows the user to *simulate* dynamic behavior rather than *analyze* the buckling properties of the structure. The results obtained with the quasi-static method may be not very close to reality, they tell a lot about the structures buckling behavior in general. Both methods can be very useful but their capabilities may not be overestimated.

6 Numerical Examples

The performances of the beam elements and the transient iteration method developed in the previous chapters are shown using a number of testcases. Most of them are obtained from the papers which have been used in the derivations of the theories. Some of them can be denoted classic: they appear in almost every publication about this subject and are therefore essential in a numerical evaluation.

Most of the examples, except from the enormous mode-jumping analyses, are rather simple. Recalculation of the results only takes a few minutes. For this reason, the can perfectly be used to validate the elements and the processor in the future, when new features have been added. They will be enclosed in the database of testcases in `B2TEST` macro-processor.

6.1 Nonlinear Beam Elements

Some specific element types can be validated with a number of prescribed test series. For example, the MacNeal-Harder series contains a number of tests to validate nonlinear shell elements and their drill rotation problems in particular. Unfortunately there are no such tests for nonlinear beam elements. The testcases presented in this section are made up by the author or collected from various articles and cover almost all applications of geometrically nonlinear beam elements.

First a set of linear deflection tests is executed to test the reliability of the elements in linear analyses. The tip displacements of a number of beam structures is compared to analytical solutions, the existing linear beam element `B2.EP` and the Rebel shell element `Q4N.REBEL`. Linear dynamic eigenvalues calculations are done as well.

The performances of the element in nonlinear analysis are tested with the continuation routine `B2CONT` and the linear buckling analysis processor `B2BUCK`. The capacities in buckling and bifurcation point analysis have extra attention. The results are compared to the beams' slenderness ratios in order to find out

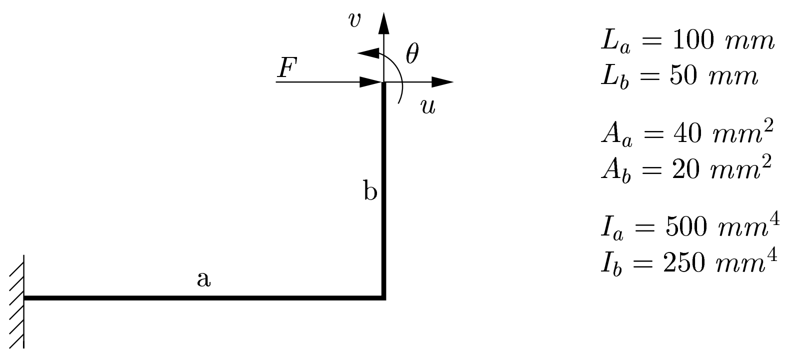


Figure 6.1: Plane beam structure for linear deflection test

	Roark	B2	B2.EP	B2.EP+	B2.NL	Q4N.REBEL
u	3.346	3.348	3.346	3.346	3.333	3.336
v	-2.500	-2.500	-2.500	-2.500	2.500	-2.520
θ	-0.075	-0.075	-0.075	-0.075	-0.075	-0.075

Table 6.1: Deflections and rotation of plane beam structure under discrete force

in which regions of slenderness the beams are still reliable.

6.1.1 Linear Deflection Test

In the first test, a plane beam structure is presented, 6.1. The two bars (a and b) have different lengths and cross sectional variables. The structure is made of an isotropic elastic material with Young's modulus $E = 2.0 \cdot 10^4\ N/mm^2$ and Poisson's ratio $\rho = 0.3$. At one end, the beam is fully clamped, at the other end a nodal force is applied with a magnitude of 100 N. The model is made of 6 beam elements, each of them 25 mm long. All available beam elements, including the linear beam element `B2` have been tested in this case. In order to calculate the tip displacements considering shear deformation, this simple beam structure is also modeled with `Q4N.REBEL` shell elements.

The deflections u and v of the tip, as well as its rotation θ can be calculated analytically with the 'slope and deflection formulas for straight elastic beams' (the *vergeet-me-nietjes* in Dutch) as presented in Roark's Formulas for Stress & Strain [35]. The analytical solutions and the numerical results can be found in table 6.1.

It can be seen that the new plane beam elements `B2.EP` and `B2.EP+` perfectly prescribe the displacements and tip deflections. This is not surprising, since the analytical beam deflection formulas are based on the principle that excludes shear deformation, in this perspective equal to the Bernoulli hypothesis. This is also the reason that the u-deflection of the 3-dimensional nonlinear beam `B2.NL` does not match the analytical estimation. The Bernoulli hypothesis is not used in the development of this element which makes it somewhat stiffer for lateral displacements. Analytical estimations of the deflections of such beams including shear deformations are not available so that an alternative way to

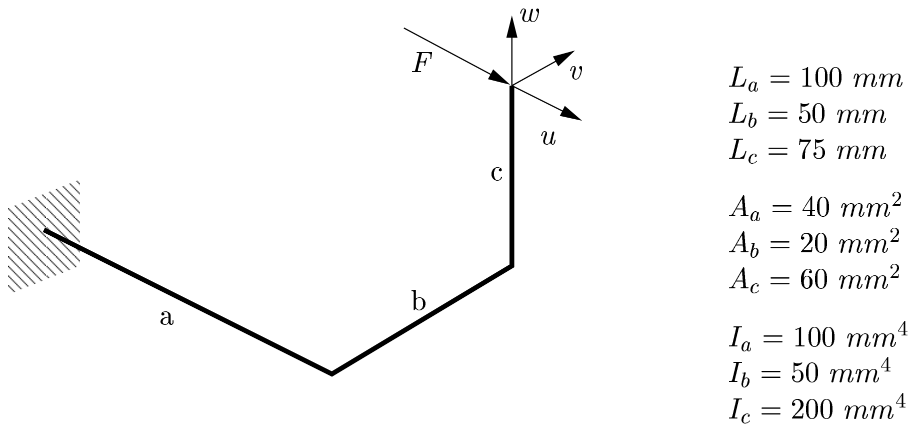


Figure 6.2: Three dimensional beam structure for linear deflection test

	Roark	B2	B2.NL
u	7.650	7.649	7.613
v	-1.250	-1.250	-1.250
w	-1.875	-1.875	-1.875
θ_x	0.000	0.000	0.000
θ_y	0.000	0.000	0.082
θ_z	-0.038	-0.038	-0.038

Table 6.2: Deflections and rotation of 2-dimensional beam structure under discrete force

check the validity of the `B2.NL` element must be sought. Shear deformation is also included in the `Q4N.REBEL` shell element and is the best element to compare the beam element with. It is therefore not surprising that the u deflections of both elements are almost identical.

The out-of-plane deformations of the 3-dimensional beam element `B2.NL` is tested with an extended version of the previous case. Due to the addition of an out-of-plane bar c the torsional stiffness of the beams participates in the tip-deflection as well, as can be seen in figure 6.2. The material parameters E and ν are the same as in the previous case. The moment of gyration J is assumed to be equal to two times the moment of inertia, $J = 2I$. The analytical solutions and the numerical results can be found in table 6.2.

Again, the deflections of the 3-dimensional nonlinear beam element are quite good compared to the analytical results and the linear element `B2`, Apart from the previously mentioned deviation due to shear deformation, the results are almost identical.

The performances of the mass description is tested by a linear dynamical eigenvalue analysis. The first three eigenfrequencies of both frames are calculated with the macro-processor `B2LIN`. The mass density of the material in both frames is assumed to be $\rho = 0.001\ kg/mm^3$. There are no numerical results available so that the results can only be compared to the existing linear beam

	freq.	B2	B2[LD]	B2.EP	B2.EP+	B2.NL	Q4.REBEL
2D model	1	0.5923	0.590	0.594	0.594	0.591	0.594
	2	2.2472	2.224	2.280	2.280	2.235	2.245
	3	5.2676	5.221	5.570	5.570	5.285	5.361
3D model	1	0.102	0.101	-	-	0.1012	-
	2	0.107	0.106	-	-	0.1063	-
	3	0.234	0.231	-	-	0.2359	-

Table 6.3: The first three eigenfrequencies $[Hz]$ of the 2 beam models using different beam elements

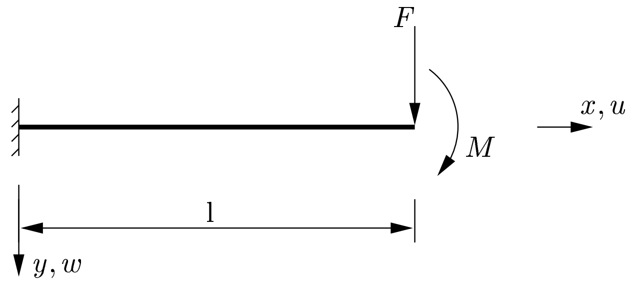


Figure 6.3: Cantilever beam under both a transverse load and a bending moment

element `B2`. This element is equipped with two descriptions for the mass: a consistent mass matrix and a lumped diagonal one. Both are used in this test. Furthermore, new models have been made. Instead of the somewhat long elements (25 mm), smaller elements (10 mm) are used instead. The result of the analyses are shown in table 6.3. It is not surprising that the performances of the reduced lumped diagonal mass matrix are somewhat behind. The eigenfrequencies of all 4 elements (`B2.EP`,`B2.EP+`,`B2.NL` and `Q4N.REBEL`) in most case too high compared to the eigenfrequencies of the old `B2` element. This is a result of the absence of mass inertia. The mass is not divided properly over all the degrees of freedom.

6.1.2 Large Deflection of a Cantilever Beam

In this simple example, the general quasi-static nonlinear behavior of the beam elements in large deformations is tested. The test is obtained from a paper by Eriksson and Pacoste [5]. A cantilever beam as shown in figure 6.3 is loaded at the tip with either a transverse load or a bending moment. The length of the beam is $l = 100$, the beam cross section properties are $A = 0.6$ and $I = 0.018$. Young's modulus is $E = 1.0 \cdot 10^8$. The beam is modeled using 4 beam elements of uniform length. On the left-hand-side, all displacements and rotations are locked. The applied unit force is 1000, the unit moment 10000. The initial load factor λ which is used to start the continuation process is 0.001 in both cases. The figures 6.4 and 6.5 show the tip deflection of the beam under force and

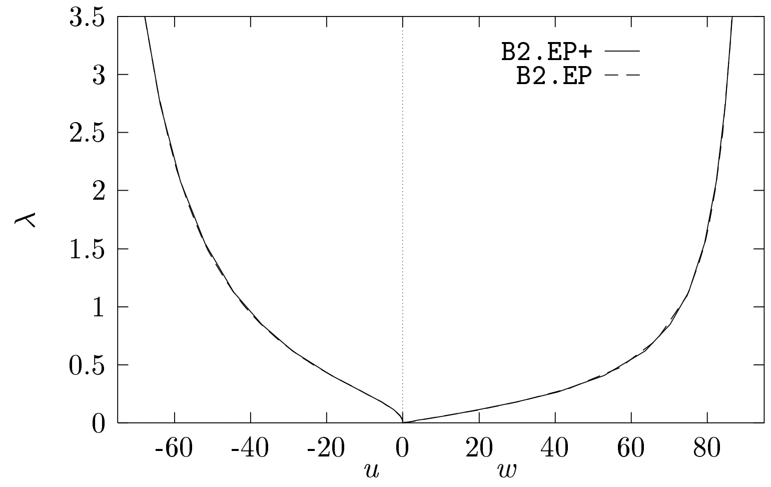


Figure 6.4: Tip displacements u and w of the cantilever beam under transverse load.

moment load respectively.

The performances of the two 2-dimensional beam elements `B2.EP` and `B2.EP+` are almost identical. The results are the same as presented in literature. It can be concluded that the error due to the simplification of the curvature is very small in this case. The choice to model the curvature this way in the 3-dimensional beam element is hence justified.

The 3-dimensional element `B2.NL` is also subjected to this test. Unfortunately due to the problems sketched in the section 3.7, the results were rather poor. Just the almost linear part of the path could be calculated, up to a load-factor $\lambda = 0.05$.

6.1.3 Pure Bending of a Cantilever Beam under a Moment Load

A classical example to prove the beam's ability to rotate over 180° is the following case obtained from an article by Simo and Vu-Quoc [29]. A straight beam of unit length $L = 1$ and bending stiffness $EI = 2$ is subjected to a concentrated end moment M, figure 6.6. The beam's cross section is, in this academic case, infinitely large, say 1000. The model is made of 20 beam elements of uniform length. The exact analytical solution to this problem is known. The beam is completely rolled up when the applied moment is 4π; when the applied moment is 8π the beam is rolled up twice. The 2-dimensional beam element with Timoshenko curvature (`B2.EP+`) is subjected to this test. The results are shown in figure 6.7.

It can be seen that the results matches the analytical solution quite well, except for the last part, where $\theta_{\text{tip}} > 5.0$ *rad*. This is a result of the simplified elastic rotation θ_e in the strain energy expression, equation (2.42). The simplification holds for small angles θ_e. In this case, the cantilever beam is uniformly bent and the elastic rotation of each element is equal to $\theta_e \approx 5.0/20 = 0.25$ *rad* which is equal to 14°. This angle is indeed to large to be evaluated by a Taylor

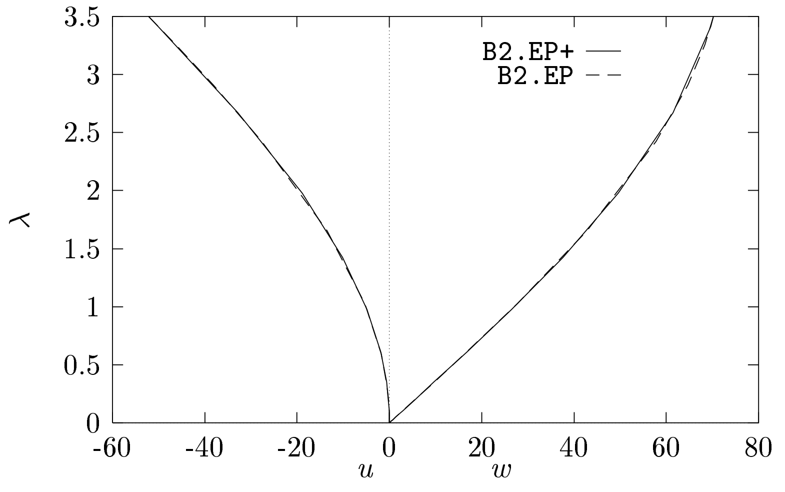


Figure 6.5: Tip displacements u and w of the cantilever beam under moment load.

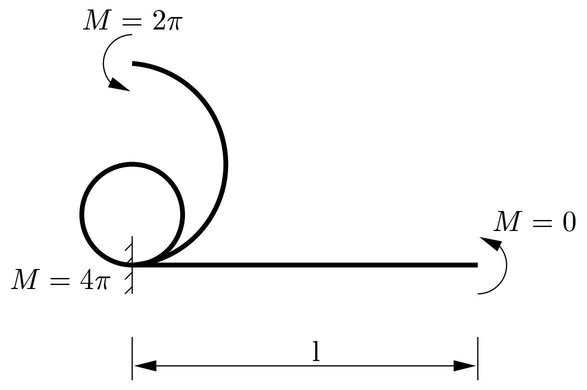


Figure 6.6: Cantilever beam subjected to an extreme tip moment

expansion series of the first degree. The same case is also calculated with 10 elements of uniform length. Unfortunately it was not possible to roll up the beam completely.

6.1.4 Euler Buckling at Different Slendernesses

Buckling behavior of beams is one of the classical mechanic subjects. A first analytical solution has been found by Euler. In his derivation the beam is assumed to be very slender, i.e. the radius of gyration $r = \sqrt{I/A}$ over the length of the beam l is small, approaches 0. Since the beam developed in this thesis is assumed to be slender as well, it must be can be compared to Euler's theories. In this case, the buckling stiffnesses are compared to the slenderness of the beam in order to check in which regions the results are reliable.

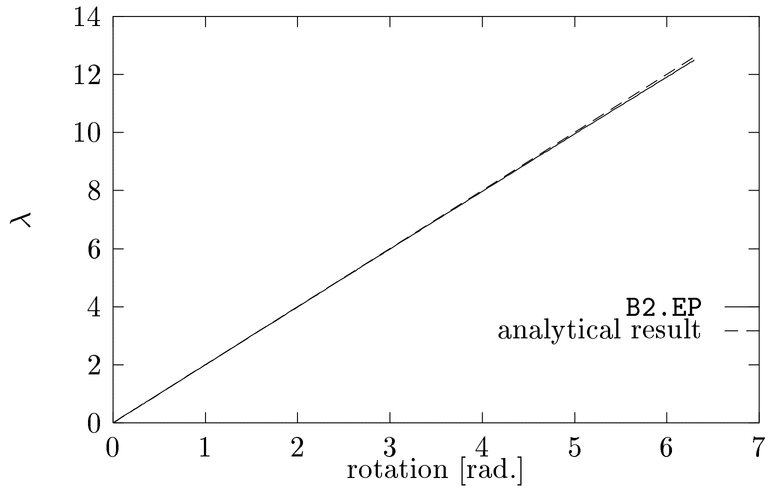


Figure 6.7: Tip rotation of cantilever beam under extreme moment

A Hinged Euler Strut

A cantilever beam shown in figure 6.8 is investigated. The length l is 100, its moment of inertia $I = 0.018$. The cross section A can be varied. The beam is made of an isotropic material with Young's modulus $E = 10^8$. A compressive force N is applied at the end of the beam. Using Euler formula, the critical load can be calculated by

$$N_{cr} = \pi^2 \frac{EI}{l^2}$$

In this case the critical load will be $N_{cr} = 1776.53$.

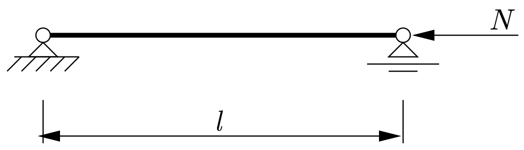


Figure 6.8: A simply supported Euler strut

The finite element model of the beam is made of 10 elements of equal length. The displacement in x- and y-direction of the unloaded end are locked, as well as the displacement in y-direction of the loaded end. In order to avoid rigid body modes, the rotation of the midpoint of the beam is locked too.

The critical loads are calculated using the B2000 buckling analysis macro-processor (B2BUCK). Both 2-dimensional elements are used. Different values for the cross section area are used to obtain different slendernesses.

A Clamped Euler Strut

In this case, both ends of the beam are clamped, figure 6.9. All dimensions and material constants of the strut are equal except for the length, which has

I	A	r/l	`B2.EP`	`B2.EP+`
0.018	0.0018	0.0316	1827.05	1790.69
0.018	0.018	0.01	1783.25	1779.72
0.018	0.18	0.00316	1777.21	1776.86
0.018	1.8	$1.0{\cdot}10^{-4}$	1776.61	1776.57
0.018	18	$3.16{\cdot}10^{-4}$	1776.56	1776.55
0.018	180	$1.0{\cdot}10^{-5}$	1776.55	1776.55

Table 6.4: Euler loads for different slendernesses (simply supported beam)

I	A	r/l	`B2.EP`	`B2.EP+`
0.018	0.0018	0.0316	1843.33	1790.69
0.018	0.018	0.01	1783.04	1780.19
0.018	0.18	0.00316	1777.57	1777.22
0.018	1.8	$1.0{\cdot}10^{-4}$	1776.97	1776.93
0.018	18	$3.16{\cdot}10^{-4}$	1776.91	1776.91
0.018	180	$1.0{\cdot}10^{-5}$	1776.91	1776.91

Table 6.5: Euler loads for different slendernesses (clamped beam)

been set to $2l = 200$. The critical load of a clamped Euler strut is equal to the critical load of a simply supported beam with half the length, so $N_{cr} = 1776.53$. Again, the finite element model consists of 10 elements. Again the critical loads

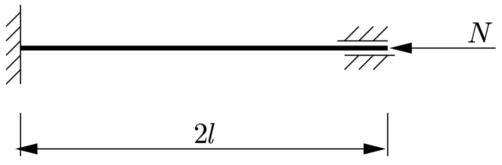


Figure 6.9: A clamped Euler strut

are calculated for varying radii of gyration.

The most spectacular result of these examples is that the performances of the Timoshenko beam are significantly better, even when the beam is less slender. This can be explained as follows. When the beam is slender, the cross sectional area is large compared to the moment of inertia. The corresponding axial strain is then small compared to the curvature. When the beam is not slender, the area is small compared to moment of inertia and the contribution of the axial strain is large. In the simplified beam, the axial strain term in the curvature is neglected. In case of slender beams this is no problem. When this is not the case, the error in the second order term (the curvature) is too large and the results are unreliable. Nevertheless, when the slenderness ratio is smaller than $1 \cdot 10^{-3}$, both beams gives satisfying results.

The analysis is also done with the 3-dimensional element, but without a

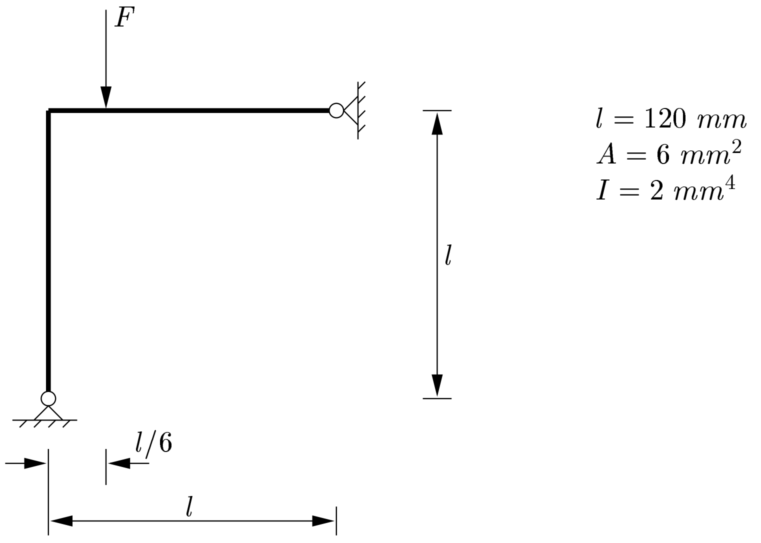


Figure 6.10: Geometry of a hinged right-angled frame

satisfying result.

6.1.5 Buckling of a Hinged Right-angle Frame under a Fixed Load

A more complicated example which combines both Euler buckling and snap-through can be found in Simo-VuQuoc [30]. A hinged frame as drawn in figure 6.10 is subjected to a nodal force at $1/6^{th}$ of the corner. The load is conservative, i.e. the direction of its work-line does not change during the deformation process. The frame is made of isotropic material $E = 7.2 \cdot 10^6\ N/mm^2$. The frame is modeled using 12 `B2.EP+` beam elements of equal length, 6 elements in each leg. The displacements in the hinges are locked; the rotations are free. The slenderness of the legs is $r/l = 2.0 \cdot 10^{-3}$ which is near the upper boundary which has been defined in the previous example.

The load displacement curves of the loaded point is shown in figure 6.11. The results are equal to those found in literature.

6.2 Linear Transient Analysis

The linear part of the transient solver `B2TRANS` can be tested easily. Analytical solutions are available in many cases. Furthermore, the responses can be compared to the results of the eigenvalue processor `B2LIN` or the explicit time integration routine `B2ETA`.

6.2.1 A Flat Plate with In-plane Initial Velocities

The following example which is obtained from a memorandum by P. Volgers [33]. It considers a flat strip that vibrates in the axial direction as a consequence of an increasing initial in-plane velocity. The dimensions and the boundary

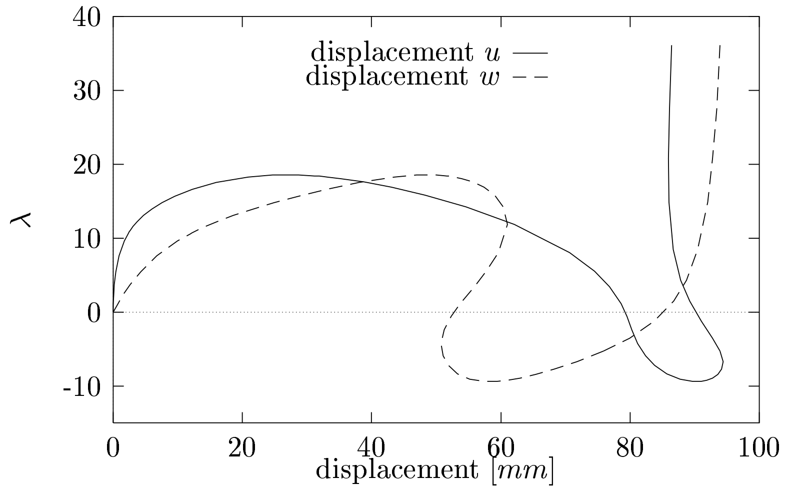


Figure 6.11: Horizontal and vertical displacement of hinged frame under conservative load

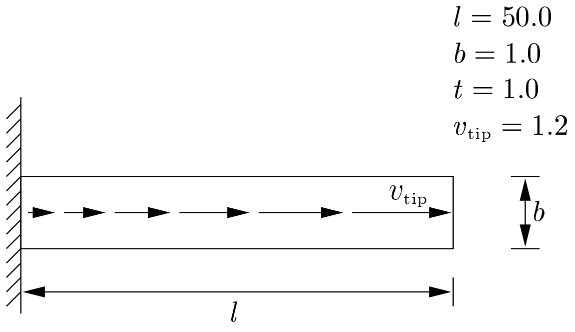


Figure 6.12: Model description

conditions are given in figure 6.12. The strip is made of an isotropic material with $E = 1 \cdot 10^4$ and $\nu = 0.0$. The mass-density is $\rho = 0.01$. There are no external forces applied to the beam. The initial displacement is equal to zero, the axial initial velocity is linearly increasing from zero at the clamped root to v_{tip} at the tip.

Since this case is fully linear, the model is made of 10 Stanley shell elements `Q4.ST`. Furthermore the mass is modeled using a consistent mass representation. The response is calculated using all 4 LMS schemes that have been implemented in the `B2TRANS` macro-processor. The time-step that has been used is $h = 0.01\ s$. The results are compared to a calculation with the explicit time integration macro-processor `B2ETA`. In this analysis, the strip is modeled with 10 `H8.ETA` 8 node volume elements.

It can be seen that results obtained with the trapezoidal rule matches the explicit results best. There is no numerical damping and hardly any phase shift. Park's LMS scheme suffers from a little amount of damping, just as Gear's 2-step method. Gear's 3-step method becomes unstable after a number 50 steps. The response is then completely wrong.

In order to test the consequences of the numerical damping of Park's LMS

Figure 6.13: In plane tip displacement of a flat clamped plate (B2ETA)

scheme, the same exercise is repeated for a number of different time-steps. As can be seen in figure 6.15, the numerical damping and the p[period shift increases with increasing time-step. This was of course expected.

6.2.2 Eigenfrequencies of a Helicopter Rotor

No matter how a structure is loaded, when it is released, it will always oscillate in one (or more) of its eigenfrequencies. By decomposing the response of a structure, it is possible to find these eigenfrequencies. This principle will be used to test the reliability of the transient processor B2TRANS.

The following test-case is obtained from [24]. A scale model of a helicopter rotor is tested in a windtunnel. In order to validate the experimental results, the eigenfrequencies are calculated with the linear processor B2LIN. A simple finite element model as shown in figure 6.16 is used in this analysis. The rotor blades are 1.7 m long and made of an isotropic material. They are connected to the shaft by a hinge. The rigid shaft is fully clamped at the free end. A set of springs with variable stiffness is attached to the blades. By varying the stiffness of these springs, the eigenfrequencies of the rotor can be influenced.

In this case, the blades are modeled with the new B2.NL beam elements, the springs (which have no mass) by R2 rod elements. The blades can only move and deform in the rotor-plane. At a certain moment the Young's moduli of the blades and the springs are $E = 1 \cdot 10^{11}$ N/mm^2 and $E = 1 \cdot 10^{5}$ N/mm^2 respectively. The first two axi-symmetric modes with the corresponding frequencies are shown in figure 6.17.

The results obtained by the linear eigenvalue analysis are simulated by the B2TRANS processor. The tips of all the blades are loaded by a pulsating force in order to generate an axi-symmetric oscillation. The period that is used is $T = 0.02412$ s, which corresponds to the second axi-symmetric eigenfrequency.

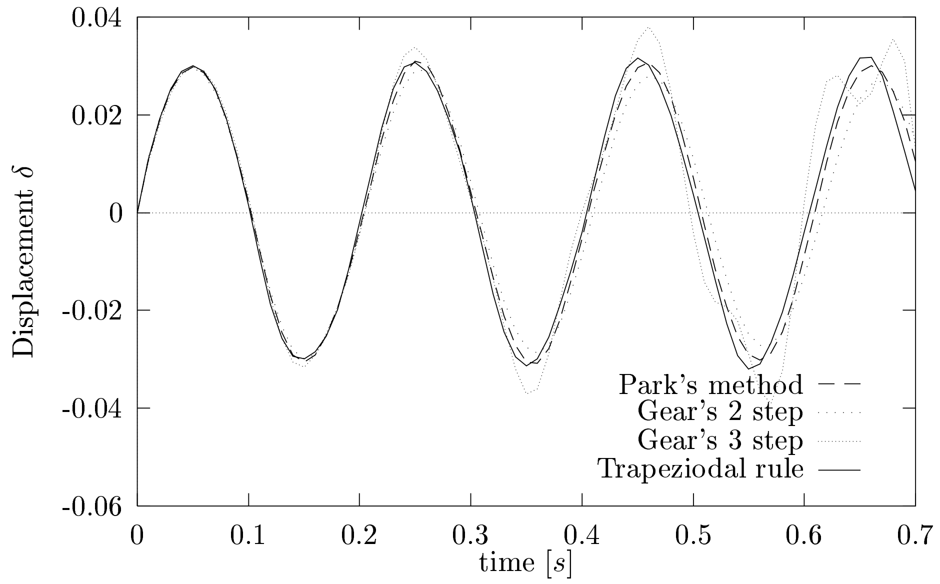


Figure 6.14: In plane tip displacement of a flat clamped plate with initial velocities.

The loading is continued for 3 periods, i.e. at $t = 0.07236\ s$ the forces are released. From this point, the blades are free to vibrate in their eigenmodes.

The lateral displacement of one of the tips as a function of time is shown in figure 6.18. As expected, the response of the blades can be decomposed in the 2 axi-symmetric eigenfrequencies. The large sine wave belongs to the first mode ($T = 0.538\ s$), the small superposed sine wave to the second mode ($T = 0.02412\ s$). It may be concluded that the transient processor is working properly and, more important, produces physical relevant results.

6.3 Nonlinear Transient Analysis

The nonlinear transient macro-processor B2TRANS is capable to perform calculations with all nonlinear structural elements available within the B2000 package. Many of these elements, for example the C2 cable element and Rebel's shell elements do not have a proper description of the mass matrix. In order to be able to use these elements, a simple mass representation for these elements is implemented. This is done by using the description of lumped masses as presented in section 3.6.

6.3.1 Stretched Cable Submitted to Transverse Loading

The following example is designed to test the accuracy of the nonlinear transient solver and is obtained from [8]. It consists of a cable of span L which is stretched with an initial tension σ_0 between 2 supports, with no sag and no initial transverse load, figure 6.19. The dynamic loading consists of a linearly increasing distributed transverse load with the function $f(t) = f_0 t$. The cable

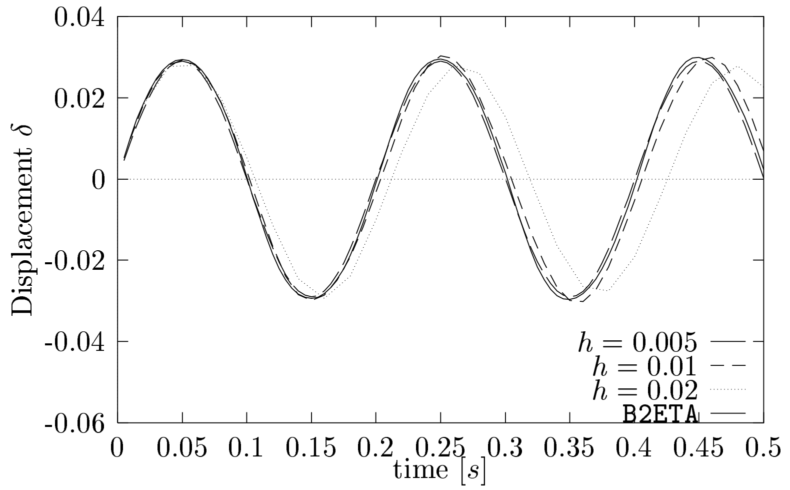


Figure 6.15: In plane tip displacement of the plate calculated with different initial time steps.

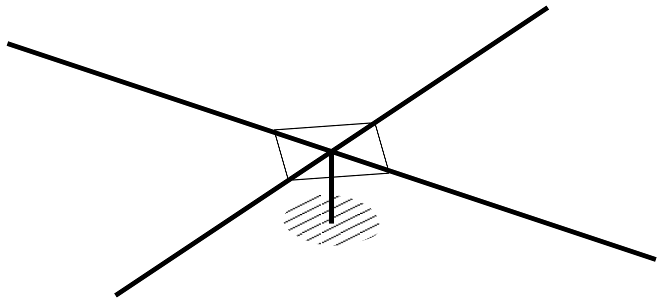


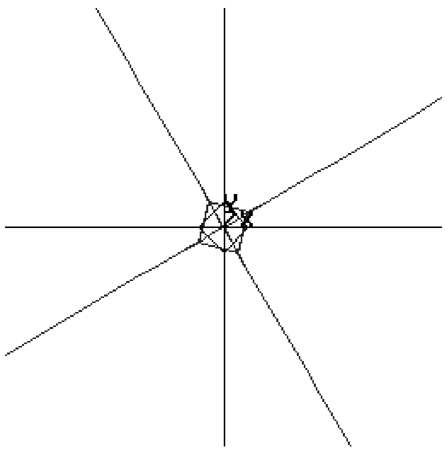
Figure 6.16: Model of a helicopter rotor

is considered to be fully elastic during the calculations and has a rigidity EA, The mass per unit length is ρA.

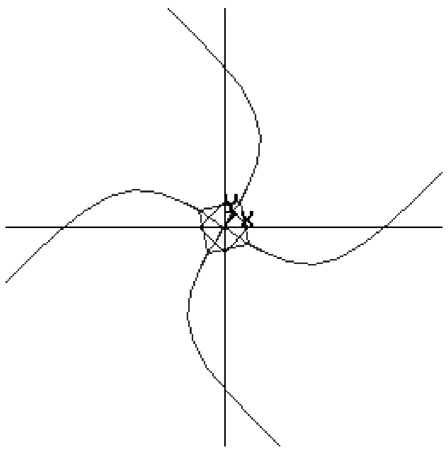
The dynamic motion is checked in the midspan node $u = y(L/2)$. The model is made of 20 `C2` cable elements as implemented in `B2000` and improved by P. Smith. The response is calculated using 3 different initial time steps, i.e. 1 ms, 2 ms and 4 ms respectively. Park's three-step method is used as the LMS scheme.

An analytical, linear solution for this problem can be deduced using the string theory. In the first phase of the deformation (to $t = 0.03$ s) this linear solution describes the nonlinear response quite well. However from $t = 0.032$ s the linear and nonlinear solutions differ rapidly. The nonlinear solution starts to oscillate, with a constant period.

A number of conclusions can be drawn from the response of the cable, figure 6.20. First, all calculations remain numerically stable. Even the analysis with the time-step $h = 4$ ms produces converged solutions. Second, the numerical damping as well as the phase shift is proportional to the time-step. The period



(a) First mode, $f = 1.86\ Hz$



(b) Second mode, $f = 41.46\ Hz$

Figure 6.17: Two axi-symmetric modes of a helicopter rotor

of the vibration is approximately 30 ms as can be seen in the figure. The accuracy of the 1 ms and the 2 ms cases rather high. The damping and phase shift of the 4 ms case is somewhat large, which is not surprising since the ratio T/h is smaller than 10 in this case.

6.3.2 Snap-through of a Cylindrical Shell

In order to test the feasibility of the transient solver in combination with non-linear shell elements, the following test case is presented [14]. Figure 6.21 shows the geometry and boundary conditions of a curved panel. In the exact center a concentrated nodal force is applied with magnitude F. The time history of the load is shown in figure 6.21a. The panel is made of an isotropic material. Young's modulus is $E = 2 \cdot 10^{11}\ N/m^2$; Poisson's ratio $\nu = 0.25$. The mass-density is $\rho = 10^4\ kg/m^3$.

Just a quarter of the panel is modeled in order to save on computation costs. Symmetric boundary conditions are applied to the cutting edges. Note that doing so, only symmetric deformation modes can be found. The nodal force, which is applied on the crossing of the two planes of symmetry, is divided by 4 as well. The quarter of the panel is modeled using 5×5 and 10×10 uniformly placed `Q4N.REBEL` shell elements. The mass is described with the reduced lumped formulation. The response of the panel is also calculated with the `STAGS` package. The 4 node 410-shell elements are used to model the panel. In all cases an initial time-step of $h = 1\ ms$ is used. The response is calculated up to $t = 0.3\ s$.

As can be seen from figure (6.22) the response of the structure depends on the number of elements that has been used. In the first part of the analysis, up to $t = 0.1\ s$ the responses of all three analyses are almost identical. The

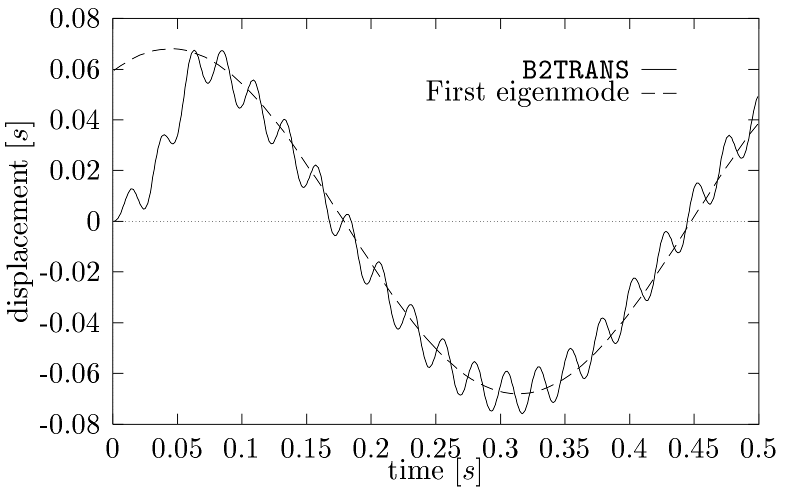


Figure 6.18: Displacement of one of the tips, axi-symmetric response

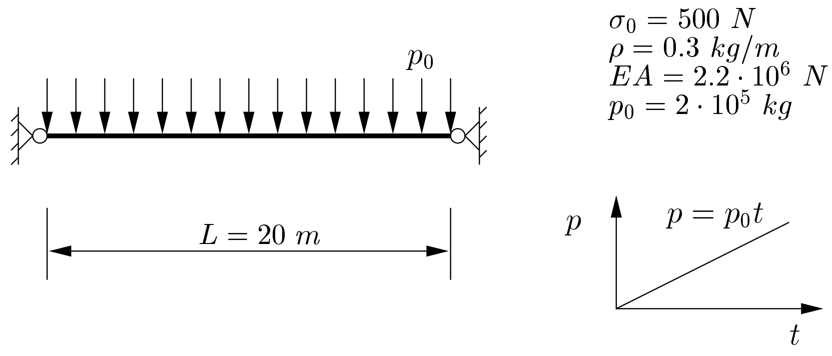


Figure 6.19: Stretched cable submitted to a transverse loading

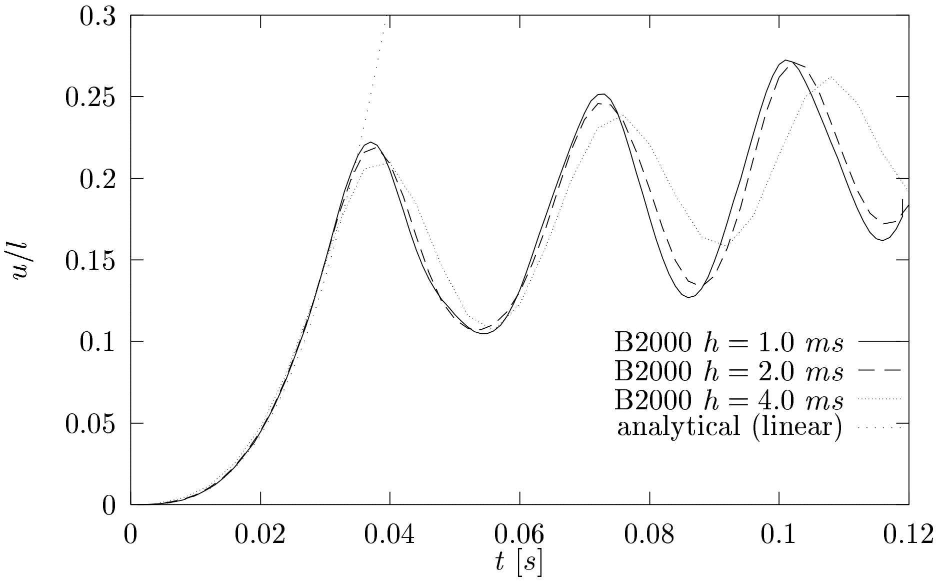


Figure 6.20: Displacement of the middle node divided by the string length l

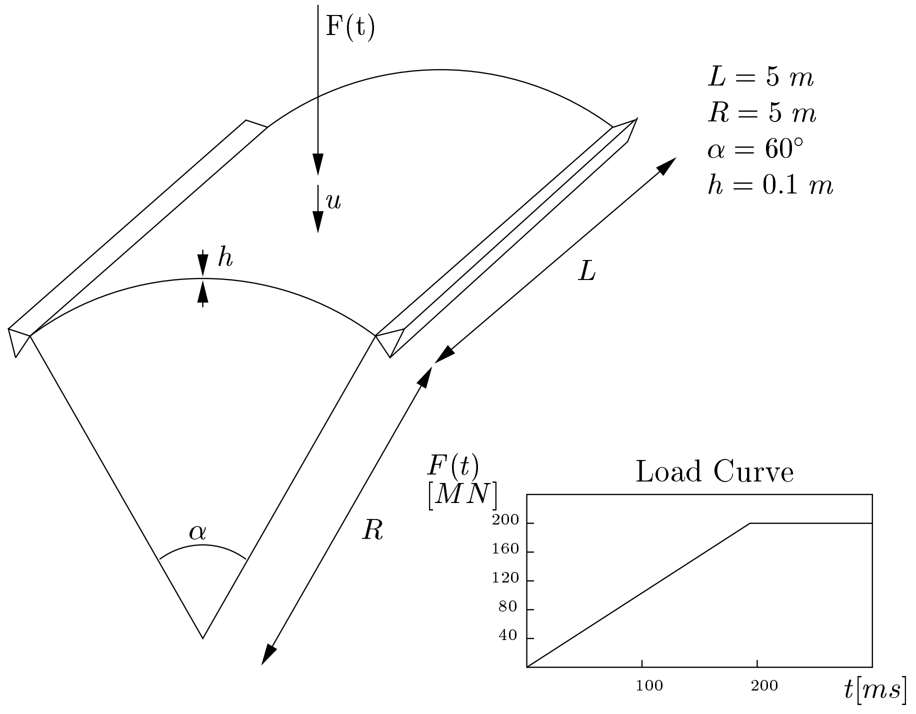


Figure 6.21: Panel geometry and prescribed load function

behavior in the highly nonlinear part of the analysis, $0.14 < t < 0.18$ depends on the number of elements that has been used. Both 5×5 and 10×10 meshes of the STAGS analyses give the same results (only the 5×5 mesh is plotted here). In B2000 the accuracy increases with increasing numbers of elements. The response of a 20×20 mesh (not printed either) approaches the STAGS solutions.

In principle since both the transient solver and the mass matrices of B2000 and STAGS are identical, this difference must be denoted to the quasi-static performances of the 2 elements. The Q4N.REBEL element is designed for thin shell structures. The ratio t/R (thickness over radius of curvature) must be small. In this case however, this ratio is equal to 0.02, which is rather high. The performances of the element decrease at this ratio.

The reduced lumped diagonal mass matrix is working properly. In the last part of the analysis, where the shell is oscillating around its new equilibrium, the response is determined by the mass properties. In this part, the global characteristics of the solutions obtained by the STAGS and the B2000 calculations are identical.

6.4 Mode Jumping

The last series of tests discusses the calculations of mode jumps as described in chapter 5. The presented structures are made of the 2-dimensional beam element as well as Rebel's shell elements Q4N.REBEL.

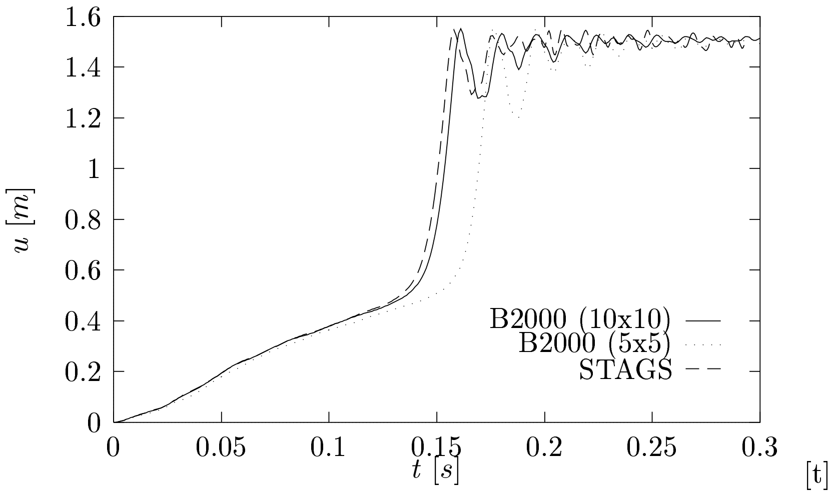


Figure 6.22: Response of the panel as calculated with B2000 and STAGS

6.4.1 A Plane Frame Structure

Consider a beam structure as shown in figure 6.23. The model is made of 32 B2.EP+ beam elements of equal length. The nonlinear post-buckling behavior of the model has been determined using the B2000 continuation package B2CONT. Since there are two different buckling phenomena involved in this example, the displacement in the load displacement plot is a combination of 2 displacements ($u = u_1 + u_2$), figure 6.24. It is possible to jump from the first limit point to the second stable branch. In order to determine the damping coefficient α and β as well as the initial time step h a linear dynamic eigenvalue analysis using B2LIN. The lowest circular eigenfrequency of the system is $\omega = 1.3434$. When a damping ratio $\xi = 0.2$ is used, the corresponding coefficients are

$$\alpha = \xi\omega \approx 0.3 \qquad \beta = \frac{\xi}{\omega} \approx 0.15 \tag{6.1}$$

As an initial time step $h = 0.2$ s is used.

After 8 quasi-static steps the stiffness matrix of the structure becomes singular. This means that the equilibrium is unstable. The estimate limit load is $\lambda_{\text{lim}} = 3.3$. From this point, with the last stable solution of step 7 as initial displacement, the transient analysis is started. The corresponding constant load-factor is $\lambda_d = 3.32$. The transient processor B2TRANS is started. After approximately 150 step, at $t = 30$ s, the kinetic energy has reached a value which is almost 10^{-6} times the maximum kinetic energy, figure 6.25. At this point the structure is at rest and the analysis can be continued using the continuation routine. The last dynamic solution $\mathbf{u}_d$ and the load factor λ_d are used as the initial value.

After a large number of iterations, the kinetic energy was apparently still too high, a new solution on the second, quasi-static stable path is obtained. From this point, the analysis can be continued without any problems.

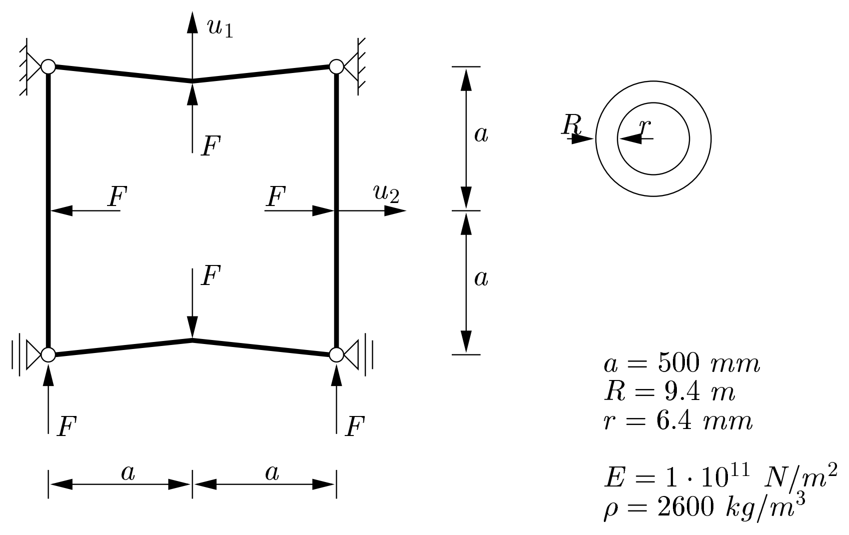


Figure 6.23: Frame geometry and loading

6.4.2 The Verolme Panel

In this example the buckling behavior of an initially undeformed panel is considered. This particular panel was first used by Verolme [32] in his PhD. thesis.

The shells is shown in figure 6.27. It is made of $2024-T3$ aluminum with the following material constants. Young's modulus is $E = 70\ MPa$ and the Poisson ratio is 0.272. The mass density of material is $\rho = 2700\ kg/m^3$. The shell is simply supported at the edges II and IV (6.27a) and clamped at the other two edges (6.27b). The panel is compressed with a prescribed end shortening at edge I, the unit displacement is $\lambda = 2.58\ mm$

The FEM model is built with Rebel-type 4 node elements (`Q4N.REBEL`) in a 20×20 mesh. The mass properties of the panel are modeled with the mass matrix of the 4 node Stanley element. In order to estimate the damping coefficients α and β the fundamental eigenfrequencies are calculated. The fundamental circular frequency is $\omega = 1469.21$. The guessed values for the Rayleigh coefficients are

$$\alpha \approx 150 \qquad \beta \approx 6 \cdot 10^{-5}$$

The period of this vibration is equal to $1/\omega = 6 \cdot 10^{-4}\ s$. As a rule of thumb the numerical damping remains small when the time-step is $1/10^{th}$ of the vibration period. In this particular case an initial time-step $h = 5 \cdot 10^{-5}\ s$ is chosen.

The pre-buckling part is calculated with the path following technique (`B2CONT`). The initial load parameter λ is set to 0.01. The unstable path is yet reached at the seventh step when $\lambda = 0.3593$. The last stable equilibrium was found in step 6 at $\lambda = 0.315$. The load of the bifurcation point of this shell is somewhere between these two load factors. The jump is started from the stable equilibrium with a constant load factor which is higher than the limit-load factor. In this case, to speed up the analysis, a rather high load factor is chosen, $\lambda_d = 0.375$.

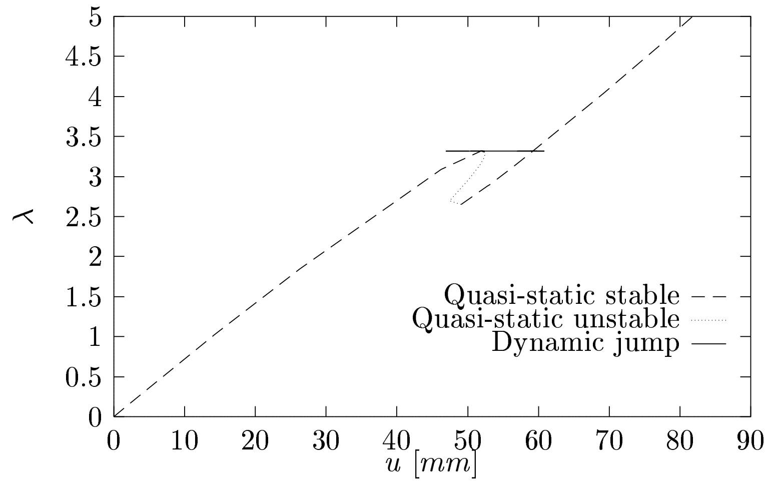


Figure 6.24: Quasi-static behavior of plane frame

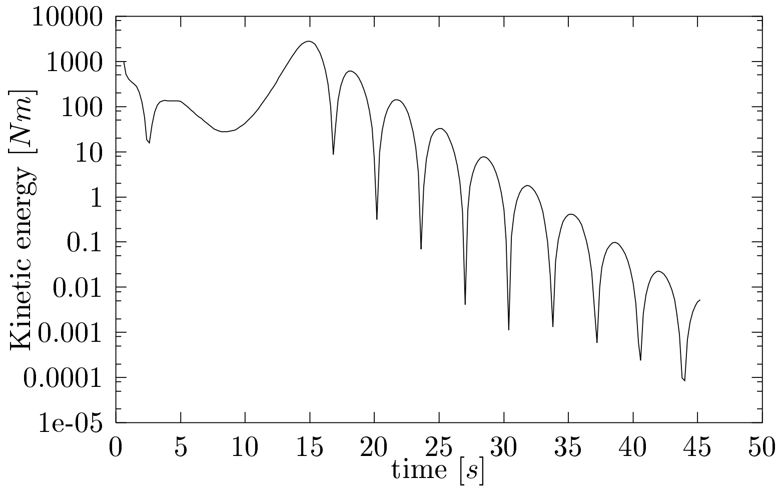


Figure 6.25: Kinetic energy of frame during the jump of the plane frame

During the jump a number of different patterns can be seen on the panel. After 2.45 ms, figure 6.31b, the number of half waves in both longitudinal and circumferential direction are rather high (4×5). Apparently, this equilibrium is not stable either, so that this mode rapidly changes into a 2×3 half-wave pattern 6.31c at $t = 7.45$ ms. It is remarkable that despite the symmetry of the model, boundary conditions and loading, this mode is not quite symmetric. The pattern is slightly moved in longitudinal direction. However, when the jump is continued, this is straightened out. The 2×3 pattern moves towards the center of the panel, 6.31d.

The transient analysis is continued up to $t = 15$ ms. After inspection of the kinetic energy during the process the decision is made whether to continue the transient process or to restart the continuation procedure. The kinetic energy of the transient process has already reached a minimum at $t = 12.3$ ms, fig. 6.30 and although it increases a little after this point, it can be concluded that the

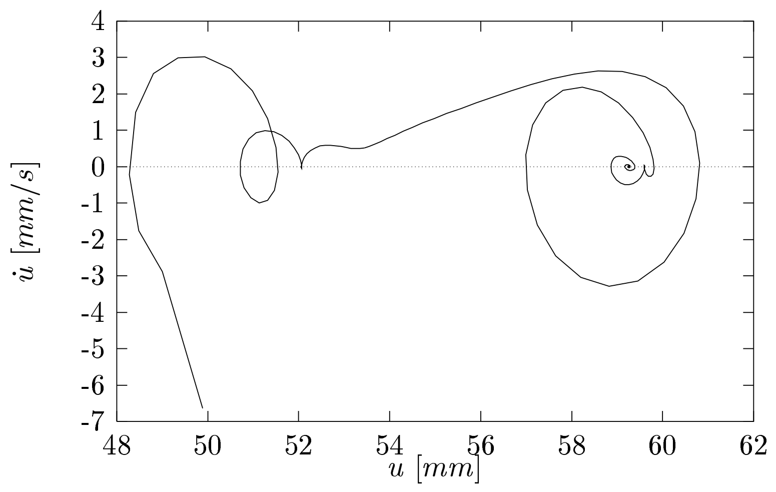


Figure 6.26: Bode plot of mode jump of the plane frame

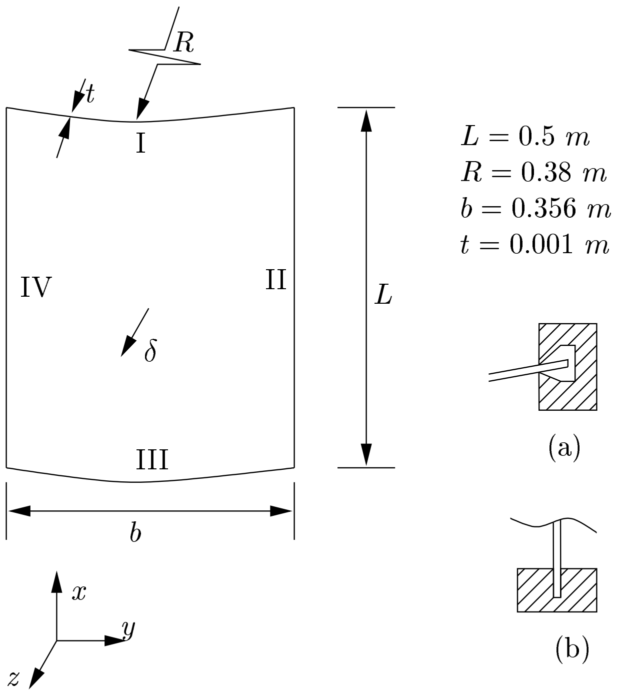


Figure 6.27: The Verolme panel and boundary conditions

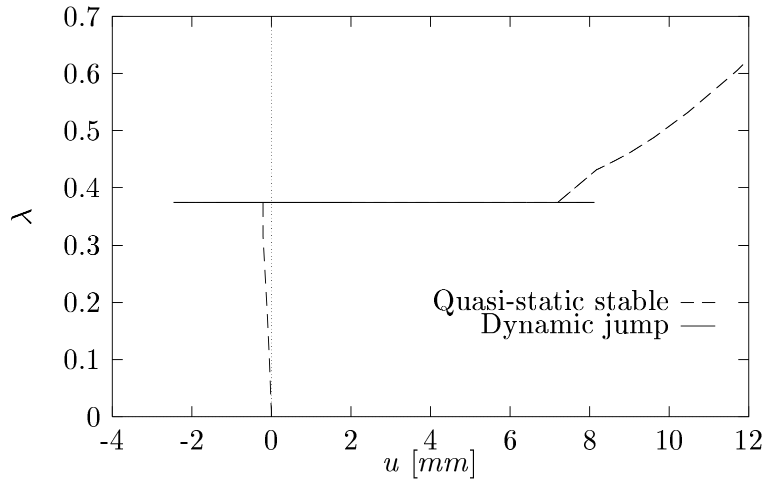


Figure 6.28: Load displacement curve of Verolme panel

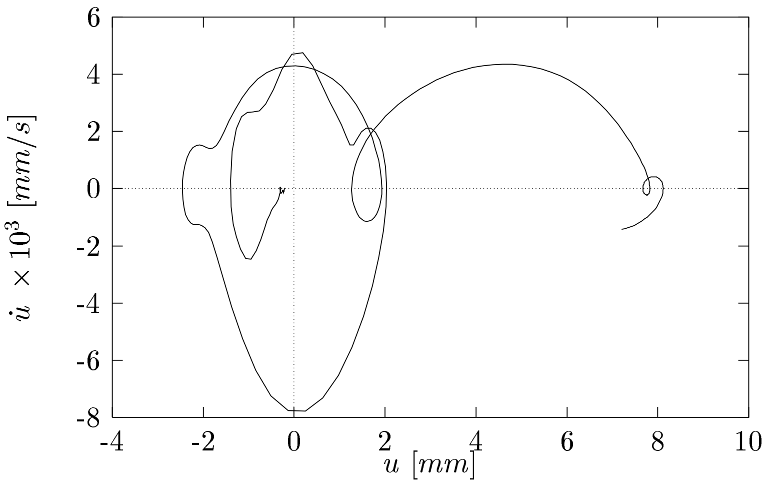


Figure 6.29: Bode plot of transient jump (Verolme panel)

jump as reached a new stable path. The bode-plot 6.29 also shows that the velocity of the node at this stage is almost equal to zero. The continuation routine is therefore restarted from the point where the kinetic energy level is a minimum, at $t = 12.3\ ms$.

Although the structure is apparently at rest, there are quite some iterations necessary to obtain a new equilibrium on the second branch. The stiffness matrices of the equilibrium states on this static path are positive definite. The solution can said to be stable.

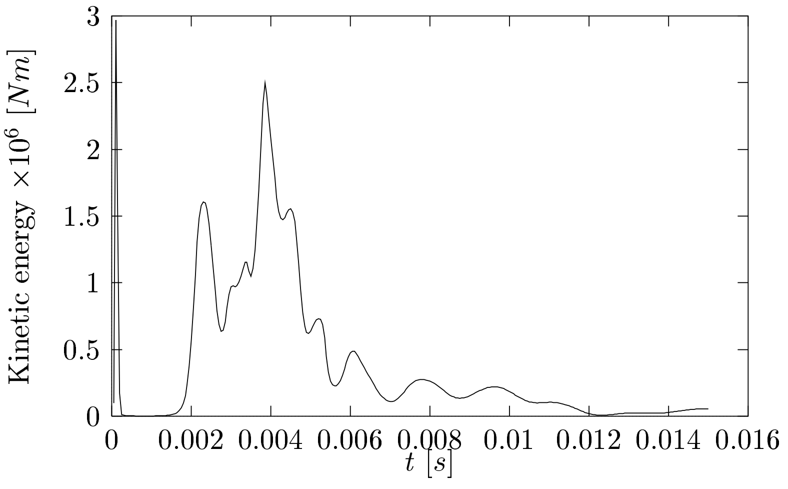
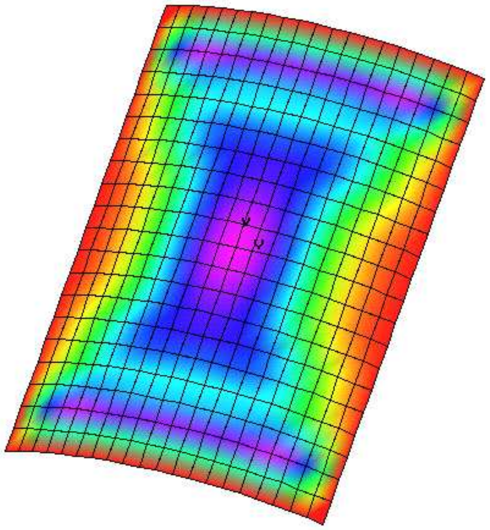
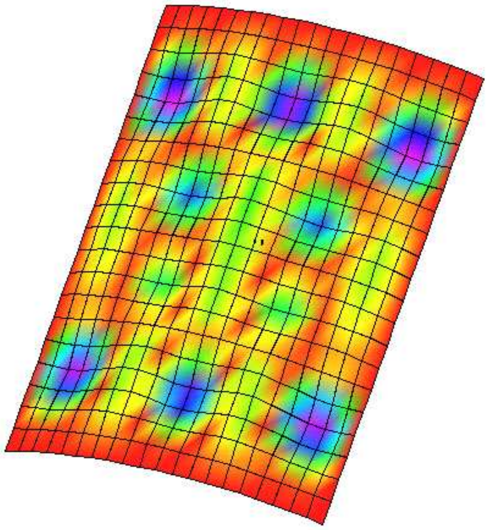
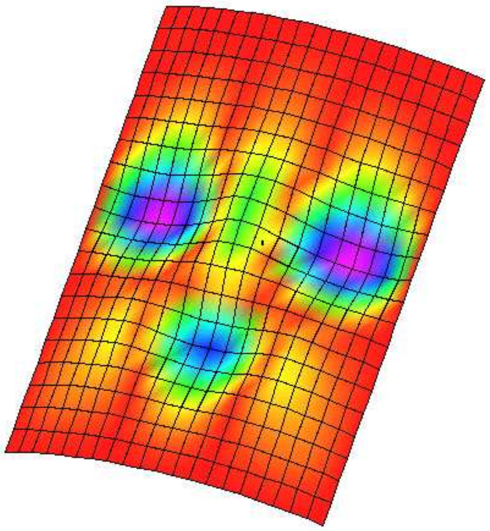


Figure 6.30: Kinetic energy during the jump (Verolme panel)


(a) Pre-buckling path, $\lambda = 0.315$


(b) Mode jump, $t = 2.45\ ms$


(c) Mode jump, $t = 7.45\ ms$

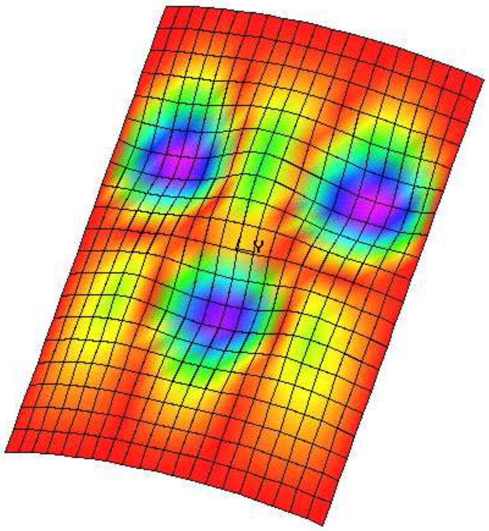

(d) Second stable branch, $\lambda = 0.615$

Figure 6.31: Deformations in z-direction of the Verolme panel during pre-buckling and mode-jump analysis (amplification 4x)

7 Conclusions and Recommendations

The main goal of this research has been the implementation of a set of nonlinear beam elements and a transient solution algorithm in order to be able to perform mode-jumping simulations in B2000. Gradually, a second goal has been defined. The beam elements and the time integration method must be developed in such way, that they fit in the B2000 environment perfectly and that they can be used in other fields of computational mechanics. In practice, this implies that the beam elements must be available for other analyses (e.g. B2LIN, B2BUCK and B2CONT), the transient processor B2TRANS must be able to handle other elements rather than the new beam elements. Furthermore, the source code must be written in such way that other users can immediately survey the code and add new developments.

Theoretical formulation

Above all things, the mode-jumping problem is a post-buckling phenomenon. The beam elements are therefore based on an engineering beam model. The performances of this theory are claimed to be excellent in the post-buckling regions, as opposed to traditional models like for example the Lagrange-Green description.

In order to get insight in the development of finite elements, first a set of 2-dimensional beams has been developed and implemented. The beams are based on papers by Eriksson and Pacoste [5, 6]. The strain description is based on Reissner's theories. Two different models for the curvature are implemented. In the most complex element a Timoshenko description is used, the other element has got a simplified description for the curvature. Furthermore, the Bernoulli hypothesis is applied in order to prevent shear locking.

The experiences with the 2-dimensional beam element are used in the development of a 3-dimensional beam element. This element is based on papers by Simo *et al.* [29, 30] and is also an engineering element. The curvature is described by the simplified expression. Apart from the addition of an out-of-plane direction and the corresponding shear, bending and torsion modes, a formulation for the finite 3-dimensional rotations is derived. In this case, due to the complex formulation of this rotation tensor, the Bernoulli hypothesis is not applied.

All three elements have got a simplified formulation for the mass, a so-called reduced lumped mass description. The total mass of the element is divided over its nodes; the rotational masses (inertia) are equal to zero.

The theoretical formulations are restricted in the following ways.

- The 2-dimensional beam elements are restricted in the pure bending mode. Due to the Taylor expansion of the elastic rotation terms, accuracy is guaranteed for pure elastic bending where $\theta < 0.3$ rad $\approx 17^\circ$. The estimated error of the rotation is of the order $\mathcal{O}(\theta^3)$.

- The Bernoulli hypothesis is applied to the 2-dimensional beam elements. As a result of this, the bending stiffness of these elements is somewhat smaller. In linear static and buckling analyses, snap buckling in particular, the result will be somewhat conservative, which is of course not a severe problem.

- In the description of the rotations in the 3 dimensional beam, a Rodrigues type formulation is used instead of the original rotation formulae. Unfortunately, this description becomes singular when the *total* rotation of the beam is larger than π rad. Quaternions can be used to solve this problem. However, another description for the finite rotation tensor, such as the one used by G. Rebel in his shell elements [23] is a better option, but slightly more complicated to implement.

The transient solver `B2TRANS` is based on an implicit time integration method. In general, the best choice of an integration method depends on the characteristics of the differential equation. In this case, the ODE is a stiff equation: there is a enormous difference between the lowest (base) eigenfrequencies and the higher frequencies. The higher frequencies are less important and need to be filtered out. Linear multi-step methods (LMS) as proposed by Gear [7] have the ability to ignore these higher frequencies, without loosing numerical stability. Unfortunately, these methods suffer from numerical distortion or unstable bahavior. Park's integration scheme [20, 21] is based on Gear's methods, but is said to be unconditionally stable in combination with small numerical distortion.

A good alternative for the LMS methods is the trapezoidal rule. This method produces no numerical damping at all. Unfortunately, it cannot be used to integrate nonlinear equations. Due to the altering eigenfrequencies, it can become unstable. Nevertheless, the method is extremely suitable for linear equations. Park's method is preferred when the analysis is nonlinear.

Most elements that are available does not have a description for the damping. An empirical formulation, the Rayleigh damping, is used instead. The damping is supposed to be a combination of the mass and the stiffness matrix. Physically speaking, this model is not correct. However, in the simulation of mode-jumps it is a useful alternative.

Due to a number of assumptions, there are some restrictions to the analytical derivation of the transient algorithm.

- With the construction of the history vectors $\mathbf{h}_n$ no attention is paid to the fact that the displacement vector $\mathbf{u}$ and the auxiliary vector $\mathbf{v}$ contains rotational terms. In case of large, finite rotations, these terms cannot be added as 'ordinary' vector components: the rotation tensor for compound rotations must be used instead. As a result of this, the correctness of the method cannot be guaranteed for large rotations, $\theta > 1$ *rad*.

- The mass matrix is assumed to be linear. However, when large rotations occurs, the inertia terms of the matrix should change. The results can than be distorted.

Numerical Implementation

The beam elements are implemented in the `B2000` package as two different elements, `B2.EP` and `B2.NL`. The first one contains both 2-dimensional beam elements. With an additional flag `NG`, the desired curvature model can be chosen. When `NG` is equal to 1, the simplified curvature model is used, when `NG`=2 (default) the full Timoshenko curvature is assumed. This last element is denoted as `B2.EP+`. Since the element is of academic interest and not meant for 'commercial' purposes, there is no attention paid to the development of a 2-dimensional space in which the elements can be used correctly. The beam must be used in an ordinary 3-dimensional environment instead, but can only be situated in the xy plane. All out-of-plane displacements and rotations (z,θ_x,θ_y) must be locked.

The 3-dimensional beam element is also implemented as a two node element called `B2.NL`. The three node alternative (`B3.NL`) is in preparation. Unfortunately, the element does not work properly in nonlinear analysis. All axial deformations (strain and torsion) are described correctly. Furthermore, when beam is rotated without deformation (a so-called rigid body mode) the internal forces remain equal to 0. This implies that the description of the rotation tensor is also good. The problems arise when the beam is bent. When this is the case, the material frame, fixed to the beam and the spatial (reference) frame are no longer identical. It might be that some variables are expressed in terms of the wrong frame, which is fatal when the frames are no longer identical. Also, When the beam is curved, membrane locking plays an important role. This problem is tackled by applying the reduced integration method, but perhaps this is not sufficient.

The transient solver `B2TRANS` is in its present form a fully fledged macro-processor. It can be used in combination with the linear processor `B2LIN` and the nonlinear continuation routine `B2CONT`. The following nonlinear elements are tested and can be used in the processor: the `C2` cable, `B2.EP`, `B2.EP+` and

B2.NL beam, Q4.ST and Q4N.REBEL shell elements. The performances of the LMS schemes in combination with Jensen's algorithm are good. The method seems to be quite fast. Although correct benchmarks test have not been carried out yet, it can compete with the original explicit macro processor B2ETA.

Out of the four LMS schemes that have been used (Park's method the trapezoidal rule and Gear's 2 and 3 step method) the first two can be recommended. The trapezoidal rule is the perfect choice in linear transient analysis, in nonlinear cases, Park's method is the best alternative. This scheme is almost unconditionally stable for nonlinear calculations. When the time-step is chosen too big, it can be difficult to obtain a converged solution. A good prediction of the main eigenfrequency of the structure is necessary for a proper estimation of the initial time-step. A time-step that is smaller than 0.1 times the period of the main vibration mode is a good starting point. When the solutions diverge, the time-step is cut.

It is possible to simulate mode-jumps by using the transient processor. All kinds of initial values $(\mathbf{u}_0, \dot{\mathbf{u}}_0)$ and load functions can be used; in this thesis the most fundamental initial values have been considered: the initial displacement is equal to the displacement vector of the last stable solution of the pre-buckling path, the initial velocity is equal to 0, the external load (or prescribed displacement) is constant and larger than the limit load. When a new stable solution is reached, the simulation can be continued using the continuation routine B2CONT. A new restart procedure has been implemented in this macro-processor by G. Rebel.

Recommendations for Further Research

The work presented in this report is not finished yet. Due to assumptions in the theory a number of things are capable for improvement. Furthermore, new features can be added to beam elements and the transient processor. A list of recommendations is given below.

- The 3-dimensional finite rotations beam element must be finished first. When this is done, the beam B2.NL element can be developed further. The addition of a gradient vector in order to calculate stresses in the beam, the implementation of pre-stress, thermal effects and initial imperfections makes the beam a fully fledged member of the element library of the finite element package B2000.

- At the moment, material behavior of the beam element is fully elastic. A plasticity model as well as creep model can be added in order to perform more realistic post-buckling simulations.

- The transient solver B2TRANS can be extended in order to solve first order differential equations such as heat transfer problems. This will take just little effort, since in the Jensen equation, the mass matrix $\mathbf{M}$ can be omitted without any consequences. Thermal conductivity elements and Neumann elements are currently under development at the German Aerospace Laboratory DLR and can be used in order to perform dynamic thermal analyses.

- The current formulation for the history vector allows rotations that are moderately small. This means that when rotations are larger than 1.0 rad, the results can become inaccurate, when the rotations are larger than π rad, the solution will diverge. The formulation of LMS methods can be improved by taking the nonlinearity of large rotations into consideration. The history vectors $\mathbf{h}$ must than be calculated using compound rotations.

- The mode-jumping procedure can be refined by using the bifurcation mode in the determination of the initial displacement vector for the transient analysis. `B2000` offers the possibility to calculate these modes with the undeformed structure as a reference. This linear buckling mode analysis is implemented as the `B2BUCK` macroprocessor. The results are much better when these calculations are executed using the last stable solution of the continuation analysis as a starting point. This feature need to be implemented in either `B2BUCK` or `B2CONT`.

- The procedures to simulate mode-jumps can be developed further in order to obtain a fully reliable simulation. The influence of a number of parameters in the process must be examined much closer. For example the influences of the damping coefficients, initial variables and time-step.

Bibliography

[1] Bathe, K. *Finite Element Procedures.* Prentice Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[2] Crisfield, M. A consistent co-rotational formulation for non-linear, three-dimensional, beam-elements. *Comp. Meth. Appl. Mech. Engrg. 81* (1990), 131–150.

[3] Crisfield, M. *Non-linear Finite Element Analysis of Solids and Structures*, vol. 2: Advanced topics. John Wiley & Sons, 1997.

[4] Dahlquist, G. A special stabiltyproblemfor linear multistep methods. *BIT 3* (1963), 27–43.

[5] Eriksson, A., and Pacoste, C. Element behavior in post-critical plane frame analysis. *Comp. Meth. Appl. Mech. Engrg. 125* (1995), 319–343.

[6] Eriksson, A., and Pacoste, C. Beam elements in instability problems. *Comp. Meth. Appl. Mech. Engrg. 144* (1997), 163–197.

[7] Gear, C. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice Hall, Inc., 1971.

[8] Geradin, M., Hogge, M., and Idelsohn, S. Implicit finite element methods. In *Computational Methods for Transient Analysis*, T. Belytschko and T. Hughes, Eds., vol. 1 of *Computational Methods in Mechanics.* North-Holland, Amsterdam, The Netherlands, 1983, ch. 9, pp. 417–471.

[9] Gere, J., and Timoshenko, S. *Mechanics of Materials*, 3rd si ed. Chapman & Hall, 1987.

[10] Hughes, T. Transient analysis and stability. In *Computational Methods for Transient Analysis*, T. Belytschko and T. Hughes, Eds., vol. 1 of *Computational Methods in Mechanics.* North-Holland, Amsterdam, The Netherlands, 1983, ch. 2, pp. 67–155.

[11] Hughes, T. Analysis of transient algorithms with particular reference to stability behavior. *Comp. Meth. Appl. Mech. Engrg. 136* (1996), 293–315.

[12] Ibrahimbegović, A., and Frey, F. Finite element analysis of linear and nonlinear planar deformations of elastic initially curved beams. Tech. rep., Department of Civil Engineering, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1992.

[13] Jensen, P. Transient analysis of structures by stiffly stable methods. *Computers and Structures 4* (1974), 615–626.

[14] Kuhl, D., and Ramm, E. Constraint energy momentum algorithm and its application to non-linear dynasmics of shells. *Comp. Meth. Appl. Mech. Engrg. 136* (1996), 293–315.

[15] Marsden, J., and Hughes, T. *Mathematical Foundations of Elasticity.* Prentice Hall, Inc., Upper Saddle River, NJ, USA, 1983.

[16] Merazzi, S., and de Boer, A. *B2000 Processors Reference Manual.* S.M.R. Engineering & Development, 1994.

[17] Merazzi, S., and Stehlin, P. *B2000 Data Structure and Programming Handbook.* S.M.R. Engineering & Development, Bienne, Switzerland, 1994.

[18] Merazzi, S., and Stehlin, P. *MEM-COM Reference Manual.* S.M.R. Engineering & Development, Bienne, Switzerland, 1994.

[19] Ogden, R. *Non-Linear Elastic Deformations.* Ellis Horwood Limited, 1984.

[20] Park, K. Evaluating time integration methods for nonlinear dynamic analysis. In *Finite Element Analysis of Transient Non-linear Behavior*, T. Belytschko, J. Osias, and P. Marcal, Eds., Applied Mechanics Symposia Series. ASME, New York, USA, 1975, pp. 35–58.

[21] Park, K. An improved stiffly stable method for direct integration of nonlinear structural dynamic equations. *Journal of Applied Mechanics 42* (1975), 464–470.

[22] Rankin, C., Brogan, F., Loden, W., and Cabiness, H. *Stags Manual, Version 3.0.* Lockheed Martin Missiles & Space Co., Inc., Palo Alto, CA, USA, 1997.

[23] Rebel, G. *Finite Rotation Shell Theory Including Drill Rotations and its Finite Element Implementation.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1998.

[24] Remmers, J. *Development of a Database with B2000 Test Cases.* National Aerospace Laboratory, NLR, Noordoostpolder, The Netherlands, 1995.

[25] Riks, E. An incremental approach to the solution of snapping and buckling problems. *Int. J. Solids Structures 15* (1979), 529–551.

[26] Riks, E. Buckling analysis of elastic structures: A computational approach. *Advances in Applied Mechanics 34* (1998), 27–43.

[27] Riks, E., and Rankin, C. Computer simulation of the buckling behavior of thin shells under quasi static loads. *Archive of Computational Methods in Engeneering 4* (1997), 325–351.

[28] Riks, E., Rankin, C., and Brogan, F. On the solution of mode jumping phenomena in thin-walled shell structures. *Comp. Meth. Appl. Mech. Engrg. 136* (1996), 59–92.

[29] Simo, J. A finite strain beam formulation. the three dimensional dynamic probnlem. part i. *Comp. Meth. Appl. Mech. Engrg. 49* (1985), 55–70.

[30] Simo, J., and Vu-Quoc, L. A three dimensional finite-strain rod model, part ii: Computational aspects. *Comp. Meth. Appl. Mech. Engrg. 58* (1986), 79–116.

[31] Stein, M. Loads and deformation of buckled rectengular plates. Tech. rep., National Aeronautics and Space Administration, 1959.

[32] Verolme, K. *The Development of a Design Tool for Fiber Metal Laminate Compression Panels.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1995.

[33] Volgers, P. Theory and implementation of the 8 node volume element in eta. Tech. rep., S.M.R. Engineering & Development, 1997.

[34] Yildirim, K. The implementation of an implicit direct time integration processor in the finite element code b2000. Master's thesis, Delft University of Technology, Delft, The Netherlands, 1995.

[35] Young, W. C. *Roark's Formulas for Stress & Strain*, 6th ed. McGraw-Hill, 1989.

A
Users manuals

The beam elements and the transient macro-processor are implemented in the B2000 platform. The user's manuals for these elements and processor and the related input decks are given in this appendix. In the near future these pages will be added to the B2000 processors reference manual [16]. At the end of this appendix the usage of the elements and the macroprocessor will be illustrated by some small examples.

A.1 Nonlinear Beam Elements

The `B2.EP` and `B2.NL` nonlinear beam elements can be used for all linear and nonlinear analyses available within the B2000 platform.

The `B2.EP` element is a 2-dimensional element and based on the papers by Eriksson and Pacoste [5, 6]. The strains are based on a model proposed by Reissner. Furthermore, the Bernoulli hypothesis is applied. As a consequence of this, the beam cannot be deformed in a pure shear mode.

The `B2.NL` is a finite rotation 3-dimensional nonlinear beam element. The exact description of the model can be found in papers by Simo *et al.* [29, 30]. Again, Reissner's strain model is used. On the other hand, the Bernoulli hypothesis is not applied.

Since both elements are 2 node elements its position is defined by the 2 node points i and j. The `B2.NL` element requires an additional third node k in order to determine the direction of the element local x axis.

`Element Name`

`B2.EP` or `B2.NL`. Using the `NG` parameter (originally meant to determine the number of Gauss integration points) the curvature model of the `B2.EP` element can be chosen. When `NG` is equal to 1 a simplified model is used; when `NG` is equal to 2 (default), Timoshenko curvature is used.

`Element Variables`

The displacements u_x, u_y and u_z and rotation r_x, r_y and r_z at the element nodes i and j. In case of the two dimensional element `B2.EP` the displacement u_z and rotations r_x and r_y are superfluous. In the output these parameters will always be equal to zero.

`Integration`

The `B2.EP` element is integrated analytically, the `B2.NL` is integrated numerically, according to a two point reduced Gauss integration. The optional `NG` parameter is obsolete.

Required Input Processor element attributes

`mid` **m**

Specifies the element material number **m**.

`section` **parameters**

Specifies the section parameters with respect to the beam local axes. The parameters for the two dimensional are specified according to

`stiff` **area inert dummy dummy dummy dummy** [1]

area is the beams cross section, **inert** the moment of inertia.

The section parameters of the 3 dimensional beam element are specified the following way

`stiff` **area jtors dummy dummy inertx inerty**

area is the beams cross section, **jtors** the torsional stiffness, **inertx** and **inerty** the moments of inertia of the beam in the element local x and y axes.

`type` `B2.EP` **or** `B2.NL`

Specifies the current element type.

Optional Input Processor element attributes

None.

Required Input Processor material attributes

For material type *ISO* specify E and ρ.

Optional Input Processor material attributes

None.

[1] The **stiff** input card expects 6 entries. Since there are only two entries required for the 2 dimensional beam, the other 4 remain empty. By inserting a dummy value, i.e. 0.0 the input processor will accept this input card.

A.2 B2TRANS, The Transient Analysis Solver

`B2TRANS` is an independent macro-processor that performs a (non)linear dynamic implicit time integration analysis by calling the `B2000` processors `B2EP` (`B2EPN`), `B2MP` and `B2AEM`. The system of equation is solved using the `B2000` LDL solver `b2es`. The second order system of differential equations is transformed into a first order one by using Jensen's algorithm [13]. This first order ODE is solved with a linear multi-step algorithm derived by Park [20, 21] (alternative LMS schemes like the trapezoidal rule and Gear's methods [7] are available as well).

The macro-processor is able to operate alone, but can also be used in combination with the linear macroprocessor `B2LIN` or the continuation routine `B2CONT` to perform mode-jump analyses. The explanation of the strategy parameters concerning the nonlinear solution procedure can be found in the `B2000` processors reference manual [16], paragraph *ADIR Analysis directives*

Synopsis

`b2trans`(parameters)

Parameters

`-i` **ifile**

Specifies the input file name (equivalent to the processor command language command *input*). The default input is the terminal. If the processor command language input file is specified by the *-i* parameter, the processor is automatically set to *batch* mode since interactive control is no more available.

`-o` **ofile**

Specifies the output file name (equivalent to the processor command language command *output*. The default output is the terminal.

`-p` **pfile**

Specifies the name of an input file to be executed at start-up. The input file must contain valid processor command language commands. No default.

`-e` **efile**

Specifies the B2000 echo file name (see Paragraph "Common Commands". If ***efile*** is set to ***NIL*** no echo will be produced (default).

`-j` **jobname**

Specifies a job name for job identification by the B2000 job monitor (optional). No default value.

Required PCL commands

`adb` **fname**

Opens the archival data base file ***fname***. No default value assumed.

`cdb` **fname**

Opens the computational data base file ***fname***. No default value assumed.

`go` **or** `run`

Terminates the input sequence of the processor commands and starts execution.

Optional PCL commands

`acceleration`

Calculates and saves the acceleration on the database in the dataset `ACCE.GLOB.cycle`.

`adir` **parameters** `end`

Specifies the strategy parameters for the current run. The *adir* parameters are to be found in the chapter "Input Description Language", paragraph "*adir* Analysis Directives"[16].

`dyna` **parameters** `end`

Specifies the time parameters for the current run as well as the starting conditions for the transient analysis. The *dyna* parameters can be found in section (A.3).

`fullnewton`

Use full newton procedure (instead of modified newton) to solve the nonlinear equation.

`load` ***n type*** $\mathbf{t}_0$ ***per amp***

Defines the load functions. **n** is the loadfunction number. There are no limitations to the number of load-functions. All load-functions are related to the loads specified in load case 1. The forces (or prescribed displacements) in case 2, if any, will be kept constant during the analysis. Case numbers 3 and higher will be neglected in the analysis. **type** defines the type of load function. for example ***sin*** or ***cos***. The function sets in at $\mathbf{t} = \mathbf{t}_0$, ***per*** is the function's period (in case of repeating functions) or the total length. The amplitude is denoted by ***amp***. With the development a new function is introduced the slope function, which is a linear increasing (decreasing) function starting at t_0 with a magnitude 0.0 and ending at $t = t_0 + per$ with a magnitude *amp*.

`lutol` **parameter**

Sets error tolerance for LU decomposition solver.

`predmethod` **parameter**

An explicit Euler prediction is used if the parameter is set to `1`. If set to `0` a Newton Raphson prediction is used. Default is `0`.

A.3 Dynamic Analysis Parameters DYNA

`DYNA` defines the parameters which are needed for controlling the time dependent dynamic analysis in `B2TRANS` and `B2ETA` as well. Most of the parameters are required for both linear and nonlinear analysis. Whenever a parameter is optional, a default value is defined.

Synopsis

```
adir
  parameters
  end
```

Required parameters

`timestart` ***val***
: Starting time of iteration. ***val*** should be equal or larger than 0.0

`timeend` ***val***
: End time of iteration. ***val*** should be larger than `time-start`.

`dt` ***val***
: Initial time step. ***val*** should be larger than 0.0

Optional parameters

`alfa` ***val***
: Rayleigh's α damping coefficient. Default value: `0.0`

`beta` ***val***
: Rayleigh's β damping coefficient. Default value: `0.0`

`initdisp`
: Use displacements in the database (calculated by `B2LIN` or `B2CONT`) as initial displacements $\mathbf{u}_0$.

`initvelo`
: Use velocities as defined in the inputfile as initial velocities $\dot{\mathbf{u}}_0$.

`load` ***n type*** $\mathbf{t}_0$ ***per amp***
: Defines the load functions. **n** is the loadfunction number. There are no limitations to the number of load-functions as long as they are inserted consecutive. **type** defines the type of load function. for example ***sin,cos***. $\mathbf{t}_0$ is the time when the function sets in, ***per*** the function's period (in case of repeating functions) or the total length. ***amp*** is the amplitude of the function. For more information see the `B2000` Processors reference manual [16],

`method` ***name***
: Defines the linear multi-step method used to solve the dynamic equation.

Up till now, 4 methods are available, i.e. Parks, Gear 2 step, Gear 3 step and the trapezoidal method, `park`, `gear2`, `gear3`, `trapezoidal` respectively. Default value: `park`.

`noload`
Ignore load functions as defined previously.

A.4 Known Bugs

Despite all the hard work, some bugs are left in the code. Since they are all part of the user interface, they do not affect the results. In other words, so far, they are just annoying.

Initial velocities:
It is possible to assign arbitrary velocities (or spins) to specific nodes in the input processor. Most likely, this should be done in the input file using the data set called `VELO`. Right now, this data set cannot be used. The velocities can be assigned by using a force data set and calling it case 999. For instance, when node 4 must move with a speed of 3 in the x-direction, the input file must contain the following lines

```
FORCE
  CASE 999  DOF 1
  F=3.0     4
END
```

Load history function:
Due to an error, in the dyna data deck in the input file, all input after a load history function will be neglect. So, be sure that the dyna deck is always closed with the set of load history functions, if any.

```
DYNA                                DYNA
  TIME_S 0.0                          TIME_S 0.0
  TIME_E 1.0                          TIME_E 1.0
  DT     0.01                         LOAD 1 SIN 0. 1. 0.3
  ALFA   0.04                         DT     0.01
  LOAD 1 SIN 0. 1. 0.3                ALFA   0.04
END                                 END
```

The left hand side example is correct. In the other one the data assigned to `DT` and `ALFA` will be neglected.

A.5 Examples

To illustrate the usage of the beam elements the complete input file of the example as presented in section 6.1.2 is given. The mode-jumping simulation is illustrated with the input-file and `PCL` commands of the Verolme test case, section 6.4.2.

A.5.1 Large Deflection of a Cantilever Beam

The archival database `largedefl.adb` of this test case is made with the input processor `B2IP`. The `ascii` text input file is presented below.

```
Title "Large Deflection Analysis of a Cantilever Beam"
#
#  Obtained from
#
#  Author  :  C. Pacoste, A. Eriksson
#  Title   :  Element behavior in post-critical
#             plane frame analysis
#  Journal :  Comp. Meth. Appl. Mech. Engrg
#  Vol.    :  125
#  Year    :  1995
#  Pages   :  319-343
#
#  Declaration of options
#
#  eltype  :  1 Eriksson Pacoste type (88)
#               Simplified bending
#             2 Eriksson Pacoste type (88)
#               Timoshenko bending
#             3 Simo-VuQuoc type (89)
#
#  lotype  :  1 Tip force
#             2 Tip moment
#
   (eltype=2)
   (lotype=2)
#
ADIR
   ANALYSIS NONLINEAR
   LCA 1 PAS 0.0 DPAS 0.001 PAMAX 3.5              Loadcase A
   LCB 0                                           Loadcase B
   NCUT 15 NFACT 80 NSTRAT 0 MAXIT 15 MAXSTEP 100  Strategy parameters
   EPSDIS 0.001                                    error tolerance incr. displacement
   EPSR   0.001                                    error tolerance residual force
END
#
BRANCH 1
#
#  NODE DESCRIPTION
#
NODES
  1     0.0  0.0   0.0
  2     25.  0.0   0.0
  3     50.  0.0   0.0
  4     75.  0.0   0.0
  5     100. 0.0   0.0
#
  999   0.0  1.0   0.0
END
#
#  ELEMENTS
#
ELEM
  if (eltype=1) (
    TYPE B2.EP NG 1                                1: simplified 2 dimensional
    )
  if (eltype=2) (
    TYPE B2.EP NG 2                                2: Timoshenko 2 dimensional
    )
  if (eltype=3) (
    TYPE B2.NL NG 2                                3: 3 dimensional
    )
  MID 1                                            Material id. 1
  STIFF 0.6  0.0  0.0  0.0  0.0  0.018  0.018      Stiffness parameters, resp.
  101  1    2    999                               Area, Torsional stiffness
  102  2    3    999                               Shear stiffness in x and y direction
  103  3    4    999                               Moment of inertia in x and y direction
  104  4    5    999
END
#
#  BOUNDARY CONDITIONS
#
BOUND
  LOCK LLLLLL 1                                    Node 1 fully locked
  ALLLOCK FFLLLF                                   All out-of-plane deformations locked
END
#
#  FORCES
#
FORCE
  if (lotype=1)(
    CASE 1 DOF 2                                   Nodal displcement in y direction
    P=1000.   5                                    Unit load, load applied at tip
    )
  if (lotype=2)(
    CASE 1 DOF 6                                   External moment around z axis
    P=10000.  5                                    Unit moment, moment applied at tip
    )
END
#
#  MATERIAL CONDITIONS
#
EMAT
  MID 1                                            Material id. 1
    TYPE BEAM E 1.E8  P 0.3                        Beam type material
    ENDMID                                         Young's modulus, Poisson's ratio
END
#
ENDBRANCH
#
RUN
```

When the archival database is copied to obtain a computational database `largedefl.cdb` the continuation procedure can be started. After typing

```
[001]dutlcc1 % b2cont
```

the following PCL commands must be given.

```
Continuation procedure Macro Processor (b2cont), Version 1.9
```

```
B2CONT -> adb largedefl.adb                   Archival database
B2CONT -> cdb largedefl.cdb                   Computational database
B2CONT -> fullnewton                          Full Newton method used
B2CONT -> go                                  Start analysis
```

A.5.2 The Verolme Panel

The archival database `verolme.adb` is constructed in an ordinary way using the `B2IP` input processor. The input file with the specific strategy parameters is shown below.

```
TITLE='Modejumping analysis of the Verolme Panel'
#
#  NONLINEAR ANALYSIS STRATEGY PARAMETERS
#
ADIR
  ANALYSIS NONLINEAR                            Nonlinear analysis
  LCA 1 PAS 0.0 DPAS 0.01 PAMAX 100.0           Loadcase A
  LCB 0                                         Loadcase B
  NCUT 10 NFACT 10 NSTRAT 0 MAXIT 15 MAXSTEP 300   Strategy parameters
  EPSDIS 0.0001                                 Error tolerance incr. displacement
  EPSR   0.0001                                 Error tolerance residual force
END
#
#  DYNAMIC ANALYSIS
#
DYNA
  TIMESTART 0.0                                 Starting time
  TIMEEND   0.3                                 Ending time
  DT        0.0001                              Initial timestep
END
#
#  BRANCH DIRECTIVES
#
BRANCH=1
#
BDIR
   MATERIAL=ELASTIC
   DEFORM=NONLINEAR
END
#
# NODES                                         Node generation
#
NODES
   1  -2.50000E+02  -1.71570E+02   4.09371E+01
   2  -2.25000E+02  -1.71570E+02   4.09371E+01
   3  -2.00000E+02  -1.71570E+02   4.09371E+01
   4  -1.75000E+02  -1.71570E+02   4.09371E+01
   .    ...           ...            ...

   .    ...           ...            ...
 438    1.75000E+02   1.71570E+02   4.09371E+01
 439    2.00000E+02   1.71570E+02   4.09371E+01
 440    2.25000E+02   1.71570E+02   4.09371E+01
 441    2.50000E+02   1.71570E+02   4.09371E+01
END
#
```

```
#  ELEMENTS                                           Element definition
#
ELEM
  TYPE Q4N.REBEL MID 1 THICK 1.0 NGAUSS 4 NIM 2       Rebel 4 node shell element
   1   1   2  23  22                                  material id. 1
  22  22  23  44  43                                  thickness
  43  43  44  65  64                                  4 points Gauss interpolation
  64  64  65  86  85
  ..  ..  ..  ..  ..

  ..  ..  ..  ..  ..
 356 356 357 378 377
 377 377 378 399 398
 398 398 399 420 419
 419 419 420 441 440
END
#
#  BOUNDARY CONDITIONS
#
BOUND
  LOCK FLLLLL 21/441/21                               Edge I
  LOCK LLLLLL 1/421/21                                Edge III
  LOCK FFLFLL 1/21 421/441                            Edge II and IV
END
#
#  MASS MATRIX
#
MASS
  TYPE CO                                             Constistent mass matrix
  ELEM 1/419                                          All elements
END
#
#  PRESCRIBED DISPLACEMENT VECTOR
#
FORCE
  CASE=1                                              Loadcase 1
  TYPE=D DOF=1                                        Prescribed displacement
  P=-0.2      21/441/21                               Unit displacement on edge 1
END
#
ENDBRANCH
#
#  MATERIAL PROPERTIES
#
EMAT
  MID 1                                               Material id. 1
    TYPE ISO E 7.0E6 P 0.272                          Isotropic material, Young's
    DENS 2.7E-6                                       modulus, Poisson's ratio
  ENDMID                                              Density.
END
#
RUN
```

The computational database `verolme.cdb` can be created by copying the contents of the archival database to the computational database. The continuation processor can now be started. In order to obtain accurate results, a full Newton iteration procedure has been used.

After typing

```
[002]dutlcc1 % b2cont
```

the following PCL commands must be given.

```
Continuation procedure Macro Processor (b2cont), Version 1.9

B2CONT -> adb verolme.adb                         Archival database
B2CONT -> cdb verolme.cdb                         Computational database
B2CONT -> fullnewton                              Full Newton method used
B2CONT -> go                                      Start analysis
```

The path becomes unstable first at step number 7. The last stable result, step 6, will be used to initiate the transient analysis. The corresponding load-factor at step 7 is $\lambda = 0.316$. To give the structure a large enough 'boost' the constant load-factor during the jump is set to $\lambda = 0.375$. De transient processor is started by typing

```
[003]dutlcc1 % b2trans
```

The corresponding PCL commands are:

```
Transient Analysis Macro Processor (b2trans), Version 1.9

B2TRANS -> adb verolme.adb                       Archival database
B2TRANS -> cdb verolme.cdb                       Computational database
B2TRANS -> fullnewton                            Full Newton method used
B2TRANS -> predmethod 1                          Explicit euler prediction used
B2TRANS -> adir                                  Change ADIR directives
B2TRANS ->   step 6                              Start at step 6
B2TRANS -> end
B2TRANS -> dyna
B2TRANS ->   timestart 0.0                       Start time
B2TRANS ->   timeend   0.3                       End time
B2TRANS ->   dt        5.0e-5                    Initial timestep
B2TRANS ->   alfa      150.0                     Rayleigh's damping coefficients
B2TRANS ->   beta      6.0e-5
B2TRANS -> end
B2TRANS -> load 1 step 0. 0.3 3.32               Load function
B2TRANS -> go                                    Start analysis
```

B Numerical Implementation

The implementation of the beam elements and the transient processor has resulted in a large amount of new source code. There is no need to print this code here, since it can always be looked up in the `B2000` master version. Just a global outline is given in this appendix. For more information about the code, one can refer to the list of all created and modified subroutines which is given in appendix C.

B.1 Beam Elements

As said before, the `B2000` platform is designed to be used in a research environment. This means that new developments in FEM analysis can be implemented in the code very easily, using standard formats. In principle, the implementation of a new element, whether it is a structural, a viscous or a heat transfer element, can be done by creating a single source file. In this file, the construction of the *element local* property matrices (e.g. stiffness or heat conductivity matrices) of a single element are calculated. All standard operations, such as the construction of the total *global* matrices of the model and the sky-lining of these matrices are performed by other routines.

Each element in the `B2000` package has its own *id* number. The 2-dimensional beam element that has been developed in this thesis has got number 88, the two node, 3-dimensional beam has got number 89. Element number 90 is reserved for the three node 3-dimensional beam element. The mechanical properties are given by the internal forces vector and the stiffness matrix, the dynamic properties are given by the element mass matrix. For historical reasons, the creation of the mass matrix is done by an external source file. The total number of new source files is therefore equal to 2 files per element, one for the mechanical properties (internal forces vector and the stiffness matrix) and one for the dynamic properties (mass matrix).

B.1.1 Internal Forces and Stiffness Matrix

The assemblage of the element mechanical properties is done by the subroutine `b2ep`n`.F`, where n is the element id number, in this case `b2ep88.F` and `b2ep89.F`. Both files have a standard argument list in which the input as well as the output parameters (first- and second variation matrices) are included. The most important (and in this case relevant) input variables are the nodal displacements and rotations of the element (stored in the array `disp(*)`) and the incremental displacements and rotations `ddis(*)`. Material data, the element prevariational package `elprev(*)` and the element updated reference frame `elurf(*)` are also important input variables. The displacement arrays are expressed in terms of the branch global coordinate system. Since the element properties are calculated in the element local coordinate system, the displacements must be transformed into this coordinate system first. The transformation matrix which is used for this operation can be obtained from the element prevariational package.

The output of this routine, i.e. the element first variation vector `elfvar(*)` and the element stiffness matrix `elsvar(*)`, must be transformed the other way around. These quantities are calculated in the element local frame and must be transformed to the branch global system. Since the stiffness matrix is always symmetrical, just the upper triangle is stored in the array `elsvar(*)`. When the stiffness matrix is not symmetrical, which is the case with the 3-dimensional beam element `B2.NL`, the matrix must be split into a symmetric and a skew-symmetric part first. The symmetrical part is then copied to the `elsvar(*)` array.

All the element first and second variation matrices (in branch-global format) are collected by the element kernel interface module `b2ep0`. The macro-processor `B2EPN` places all element matrices in the right position and creates the global first- and second variation vector. The second variation vector (which is still the upper triangle of the stiffness matrix) is sky-lined by an external processor `B2AEM`. The results of this operation, the band of the matrix and its address vector are stored on the database as the datasets `SVAR.GLOB` (band) and `SVAR.ADR` (address vector).

B.1.2 Mass Matrix

The assemblage of the global mass matrix is done in the same way by the processor `B2MP`. The element mass matrices are constructed in the subroutines `b2ep88.F` and `b2ep89.F` respectively. The description of the mass is linear, which means that it is independent of the displacements and rotations. This implies that there are less input variables. Just general data, such as material density parameters and element dimensions (via the `elprev(*)` array) is input. The element mass matrix, in branch-global coordinate system, `sme(*)` is the only output.

The element mass matrix can be stored in two forms. When a consistent mass description is used, the mass matrix is symmetric and fully filled. The output matrix `sme(*)` is then the upper triangle of this matrix. When a lumped

elprev(1)	empty	elprev(8)	moment of inertia I_2
elprev(2)	empty	elprev(9)	empty
elprev(3)	empty	elprev(10)	empty
elprev(4)	beam length L	elprev(11)	Young's modulus E
elprev(5)	cross section A	elprev(12)	Poisson's ratio μ
elprev(6)	moment of inertia I	elprev(13)	mass density ρ
elprev(7)	cross section A_2	elprev(14)	empty

Table B.1: Element prevariational data package of 2 dimensional beam

diagonal description is used, the mass matrix is just a diagonal matrix and in order to safe expensive memory space, `sme(*)` is smaller and just contains these diagonal terms.

In the transient processor, the global stiffness and the mass matrices are added with the *dynamic stiffness matrix* as a result. It need no discussion that this can only be done when both matrices are of the same size, i.e. when a consistent mass matrix is used. In the derivation of mass matrices for the nonlinear beam elements, a reduced lumped diagonal mass matrix has been proposed. These matrices have been implemented within B2000 as consistent mass matrices, with just zeros in the off-diagonal terms. A pleasant property of diagonal matrices is that they are invariant of the coordinate system. This means that the expression for such matrices in an element-local system is equal to the expression in a branch-global system, which implies that there is no transformation needed.

B.1.3 Element Prevariational Data Packages

The previously mentioned `elprev(*)` array contains a number of constants, which describe the initial state of an element. This data does not change and is available throughout the analysis. All data in the array is stored in a double precision format. The length of the array is defined in the include file `b2ipepar.inc`. The `elprev(*)` arrays of all elements in the model are stored in one single database, called `ELPREV`.

In this case, the beam dimensions, its original position vectors (which are used to build the transformation matrix) and stiffness parameters are stored in this array. The exact location of the variables within the array of both the 2-dimensional beam element as well as the 3-dimensional beam element can be found in the tables B.1 and B.2.

B.1.4 Element Updated Reference Frame

The description of the rotation tensor of the 3-dimensional element is based on a nonlinear formulation. A new rotation tensor is calculated by updating the old one, using incremental rotations. This implies that at every step, the rotation tensor (or the so-called reference frame) must be saved on the database. It can be read at the next step and used to calculate the new rotation tensor. This data is stored in the `elurf(*)` array. This array is constantly updated (as

elprev(1)	length in x-dir.	elprev(11)	Young's modulus E
elprev(2)	length in y-dir.	elprev(12)	Poisson's ratio ρ
elprev(3)	length in z-dir.	elprev(13)	shear modulus G
elprev(4)	beam length L	elprev(14)	n_x direction vector
elprev(5)	strain stiffness EA	elprev(15)	
elprev(6)	torsional stiffness GJ	elprev(16)	
elprev(7)	shear stiffness GA_x	elprev(17)	
elprev(8)	shear stiffness GA_y	elprev(18)	
elprev(9)	bending stiffness EI_x	elprev(19)	
elprev(10)	bending stiffness EI_y	elprev(20)	

Table B.2: Element prevariational data package of 3 dimensional beam

opposed to the `elprev(*)` array which is kept constant). It can contain any information of the type double precision real. The length of this array is also defined in the `b2ipepar.inc` include file. The contents the `elurf(*)` array is listed in table B.3.

B.2 B2TRANS macro-processors

As opposed to the rather straightforward implementation of new elements, the development of a completely new macro-processor requires are more fundamental approach. There is no specific format in which the macro-processor must be written. As a result of this, many decisions regarding the internal structure of the processor are left to the programmer.

Although the `B2TRANS` macro-processor is based on the implicit time integration processor `B2IDTI` by K. Yildirim [34], it has been rewritten completely for a number of reasons. First of all Yildirim's version was not stand alone. It used the `B2LIN` macro-processor as a front end. The stiffness and mass matrices were calculated by this macro-processor. The `B2IDTI` program only performed the actual time integration procedure. Strictly speaking, it was therefore not a macro-processor, since it could not be used as an independent processor. Second, there was no interface with the input processor `B2IP`. All analysis parameters must be given by PCL commands. Finally, little attention was paid to the internal structure of the program. For example, there was no unique use of static of dynamic allocated variables.

When rebuilding the program, two macro-processors have been used as an example. The first one is `B2ETA`. This macro-processor performs the same analysis as `B2TRANS`, namely a time integration analysis. When they are considered as black boxes, they must be identical. For this reason and in order to maintain the unity throughout the `B2000` package, the user interface for `B2TRANS` is copied from `B2ETA`.

The internal structure of the source code and the data storage is copied from macro-processor `B2CONT`. The path-following technique which is implemented in this processor has many resemblances to the implicit transient analysis. Both methods calculate the response of a structure step wise, using an iterative solution procedure. Also the way the data is stored on the database is copied

Integration Point	$\xi = -0.577$		$\xi = 0.577$		$\xi = 0.0$
elurf(1)	Init. flag				
elurf(2)	$\Lambda_{1,1}$	elurf(20)	$\Lambda_{1,1}$	elurf(38)	$\Lambda_{1,1}$
elurf(3)	$\Lambda_{2,1}$	elurf(21)	$\Lambda_{2,1}$	elurf(39)	$\Lambda_{2,1}$
elurf(4)	$\Lambda_{3,1}$	elurf(22)	$\Lambda_{3,1}$	elurf(40)	$\Lambda_{3,1}$
elurf(5)	$\Lambda_{1,2}$	elurf(23)	$\Lambda_{1,2}$	elurf(41)	$\Lambda_{1,2}$
etc.		etc.		etc.	
elurf(11)	ω_1	elurf(29)	ω_1	elurf(47)	ω_1
elurf(12)	ω_2	elurf(30)	ω_2	elurf(48)	ω_2
elurf(13)	ω_3	elurf(31)	ω_3	elurf(49)	ω_3
elurf(14)	γ_1	elurf(32)	γ_1	elurf(50)	γ_1
elurf(15)	γ_2	elurf(33)	γ_2	elurf(51)	γ_2
elurf(16)	γ_3	elurf(34)	γ_3	elurf(52)	γ_3
elurf(17)	κ_1	elurf(35)	κ_1	elurf(53)	κ_1
elurf(18)	κ_2	elurf(36)	κ_2	elurf(54)	κ_2
elurf(19)	κ_3	elurf(37)	κ_3	elurf(55)	κ_3

Table B.3: Element updated reference frame data package of the 3-dimensional beam element

from this macro-processor, which benefits the interaction between `B2TRANS` and `B2CONT`. This is important regarding the most important goal for the new transient processor: the calculation of mode-jumping phenomena.

The various aspects of the macro-processor `B2TRANS` are regarded in this section. This will be done in a chronological fashion, from the initialization of the program to the final output. In figure B.1 a flow chart is presented which can be used as a guide through the program.

B.2.1 Initialization

In the main file, `b2trans.F` all I/O is initialized. The archival and the computational databases are opened and the `log` file, in which the analysis is evaluated, is created. The initialization of the integration process is done in the command module of the processor. In this file, called `b2transcm.F`, all necessary data is read from the archival data base. Furthermore, the mass and stiffness matrices are constructed.

Strategy Parameters

The initialization of the analysis also includes the reading of all strategy parameters. The strategy parameters can be divided into two classes. The first class contains data dealing with the time integration process. This information is stored on the archival database in the dataset `DYNA`. The contents of this dataset is given in table B.4. The second class contains the nonlinear analysis

Name	Description	Type [1)]	Default
`ALFA`	Rayleigh's constant α	E	0.0
`BETA`	Rayleigh's constant β	E	0.0
`INITDISP`	Initial displacement flag	I	0 (off)
`INITVELO`	Initial velocity flag	I	0 (off)
`METHOD`	LMS method	C	PARK
`STARTTIME`	starting time	E	-
`ENDTIME`	end time	E	-
`DELTAT`	initial time step	E	-
`LOADFUNC`	Number of history functions	I	0

1) I=integer, E=single precision real, C=character

Table B.4: Data entries in the `DYNA` dataset used by `B2TRANS`

Name	Description	Type [1)]	Default
`ANALYSIS`	Linear or nonlinear analysis	C	`LINEAR`
`FULLNEW`	Full or modified newton	i	1
`ICY`	Starting cycle	i	0
`MAXIT`	Maximum number of iterations	i	5
`EPSDIS`	error tolerance incr. displacement	e	0.001
`EPSEQ`	error tolerance res. force	e	0.001
`MAXSTP`	maximum number of steps	i	9999

1) I=integer, E=single precision real, C=character

Table B.5: Data entries in the `ADIR` dataset used by `B2TRANS`

strategy parameters, which control the iteration process. They can be found in the dataset `ADIR`, table B.5.

Both the dynamic and the nonlinear analysis parameters are immediately stored in common blocks, which are defined in the include file `b2trans.inc`. Doing so, the strategy parameters are available in all parts of the program.

Initial Conditions and Unit Force

When the strategy parameters are read and stored in the common blocks, the initial conditions are obtained. The initial displacement $\mathbf{u}_0$, initial velocity $\dot{\mathbf{u}}_0$ and the unit force function $\mathbf{f}_0^{\text{ext}}$ must be read from the database. When the initial displacement is the result of a quasi-static analysis (`B2CONT`) it is written in the dataset `DISP.GLOB...`n, where n is the cycle number. When the initial displacement is obtained from a linear static analysis , it is stored in the data set `DISP.GLOB.1`. Initial velocities can always be found in the dataset `VELO.GLOB...`n, the unit load vector in `FORC.GLOB.1`. It is also possible to deform the structure using prescribed displacements. The unit prescribed displacement vector is stored in the dataset `GDC.GLOB...1`, its address vector in `GDCW.GLOB`.

Construction of the matrices

The construction of the mass and stiffness matrix as well as the internal force vector is already explained in the sections B.1.1 and B.1.2, which is an interpretation from the element point of view. In this section the construction of the matrices will be considered from the transient solver point of view.

The mass matrix is constructed using the B2000 processor B2MP. This processor constructs the complete branch-global mass matrix of each branch of the model. The output is written in a vector called EMSS.*br*, where *br* denotes the branch number. The total mass matrix is constructed out of the branch related mass vectors using the processor B2AEM. The resulting matrix is stored in two data sets EMSS.GLOB and EMSS.ADR. The actual sky-lined band of the matrix is stored in the first dataset. The addresses, which mark the position of the band are stored in the second one. Since the mass matrix is linear and independent in the time domain, this procedure only has to be done once.

The stiffness matrix can be either nonlinear or linear. The nonlinear stiffness matrix is assembled using the processor B2EPN, following the same procedure as when assembling the mass matrix. First the stiffness matrix is calculated per branch. The results are stored in the data sets ELSV.*br*.n, where n denotes the cycle number. The matrix assembler creates a global stiffness matrix for the complete structure. The band is stored in the data set SVAR.GLOB.n, the addresses in SVAR.ADR.n. The consistent mass and the stiffness matrices are sky-lined using the same procedure. As a result of this, both address datasets EMSS.GLOB and SVAR.GLOB are identical. One of them can be deleted immediately. The internal forces vector is calculated also calculated by the B2EPN processor. The result is stored in the dataset FVAR.GLOB.n. When a linear element description is used, the stiffness matrix is calculated by B2EP.

B.2.2 Time Integration Process

When the initial conditions, the internal forces vector and the mass and stiffness matrices are known, the actual time integration process can be prepared. This is done in a new subroutine, called b2trlms.F.

History Vectors

First the auxiliary vector $\mathbf{v}$ and the history vectors $\mathbf{h}_n^{\mathbf{u}}$ and $\mathbf{h}_n^{\mathbf{v}}$ are calculated. The auxiliary vector is composed out of the internal forces vector and the current load. This is done using MEMCOM commands. The big advantage of this method is that the data need not to be read by the program. The manipulations are carried out in the database itself, which saves a lot of I/O time. The auxiliary vector is stored in the dataset V.PREV.1. The history vectors are calculated following the same procedure. They are stored in the datasets UHIS and VHIS.

Load Factor

The load-factor $\lambda(t)$ for the current time-step t_n is also calculated in this subroutine. The load-factor is the sum of a maximum of 20 load-functions. The value of these load-functions is determined by the subroutine `b2mapch.F`. This routine returns the function value which corresponds to ther current time t_n. The function values of all load-functions are summed with the total load-factor $\lambda(t_n)$ as a result.

Dynamic Stiffness and Force Vector

Out of the history vectors, the load-factor and the mass- and stiffness matrices, the complete Jensen equation can be formed. This is done in two different ways. In the linear case, the dynamic stiffness matrix $\mathbf{E}$ is formed and stored in the dataset `EVAR.GLOB`. The dynamic force vector $\mathbf{g}$ and the forces due to the prescribed displacement $\mathbf{g}^p$ are also determined and stored in the datasets `RHS` and `FRHS` respectively. These calculations are performed in the subroutine `b2trlin.F`. In the nonlinear case the Jacobian $\mathbf{H}_n$, the residual vector $\mathbf{r}$ and the prescribed force vector $\mathbf{g}^p$ are calculated in the subroutine `b2trnonl.F` and stored in the datasets `HVAR.GLOB.`n, `RHS` and `FRHS`.

B.2.3 Solution Techniques

When the various parts of the Jensen equation are calculated and stored on the dataset, the equations can be solved for $\mathbf{u}_n$ using the equation solver `b2es`, which solves the general linear (or linearized) equation $\mathbf{Ax} = \mathbf{b}$ for the unknown $\mathbf{x}$. The matrix $\mathbf{A}$ is first decomposed. The result is used to solve the equation using a LDL technique. When the matrix is already decomposed, the first step can be skipped. In this section the solution technique for both the linear and the nonlinear equation are regarded.

Linear Equation

The solution of the linear equation is rather straightforward. The dynamic stiffness matrix $\mathbf{E}$ is constant for all time-steps. As a result of this it must only be decomposed once. The factorized matrix can be used throughout the calculations, for every time step. The right hand side vector $\mathbf{g} - \mathbf{g}^p$ contains also the internal forces due to the prescribed displacement. These prescribed displacements are still part of the system of equations $\mathbf{E}\mathbf{u}_n = \mathbf{g} - \mathbf{g}^p$ which is therefore undetermined: the number of equations is larger than the number of unknowns. This problem is also tackled by the `b2es` processor. Instead of reducing the number of equations to the number of unknowns, which is a rather time- and memory consuming operation, penalty values are used. First the solver checks which rows and columns of the dynamic stiffness matrix belong to the described degrees of freedom. The values of these positions are replaced by an infinite large number, say $1 \cdot 10^6$ times the highest value in the original matrix. The corresponding terms in the right-hand-side vector are replace by 0. As a result of this, the prescribed d.o.f. in the displacement vector $\mathbf{u}$ will become zero (or almost zero). Later these values are replaced by the original prescribed values.

Nonlinear Equation

The solution procedure for the nonlinear equation is equal to the described above, but with some exceptions. In this case the solution must be found iteratively and the matrix of the equation $\mathbf{H}$ is not constant for all time steps. The solution can be found using a full- or a modified Newton iteration method. The only significant difference between these two methods is that for the first method the Jacobian $\mathbf{H}$ must be calculated and decomposed at every iteration step. When using the modified Newton procedure the Jacobian is constant for all iteration steps. It just needs to be calculated and decomposed at the beginning of the iteration process. The solution of this equation, $\Delta\mathbf{u}$ must be added to the previous solution $\mathbf{u}_n^k$. The rotational d.o.f. in this incremental displacement vector are added according to the theorem as described in section 4.6.3. This is done in a general subroutine `b2cirot.F`, previously used in the continuation routine.

After every iteration step the convergence is checked by evaluation the total length of the residual force vector, $||\mathbf{r}||$ and the length of the incremental displacement $||\Delta\mathbf{u}||$. When both values are approaching 0, the solution is converging. When the values are smaller than a certain value, section 4.6.4, the solution $\mathbf{u}_n$ is accurate enough and can be saved on the database as the displacement of this step, `DISP.GLOB.`n. When the solution does not converge within a maximum number of iterations, the iteration procedure is aborted. The time step h is cut and the iterations start from the beginning with the previous solution $\mathbf{u}_{n-1}$ as a starting point. This implies that the complete procedure must be repeated, starting with the assemblage of the history vectors $\mathbf{h}_n^{\mathbf{u}}$ and $\mathbf{h}_n^{\mathbf{v}}$. When the maximum number of time step cuts is exceeded the complete analysis is aborted.

B.2.4 Additional Calculations

When a converged solution $\mathbf{u}_n$ is found, some additional calculations must be done. These calculations are the same for both the linear and the nonlinear equation and are therefor executed in the central part of the LMS module, `b2trlms.F`. The data is stored on the database and the calculations are prepared for the next step.

Velocity, Acceleration and Energy

The velocity of the current time-step $\dot{\mathbf{u}}$ is calculated using the converged solution and the history vector $\mathbf{h}_n^{\mathbf{u}}$. The result is stored in the dataset `VELO.GLOB.`n. The acceleration will be calculated on request, using the current velocity and an additional history vector[1] $\mathbf{h}_n^{\dot{\mathbf{u}}}$. The result is stored in `ACCE.GLOB.`n. Finally,the

[1]this history vector is not relevant for the Jensen procedure, but can be determined at the same time as the other two history vectors.

kinetic and strain energy are also calculated and saved in the description table of the dataset `DISP.GLOB.`n.

Removing Temporary Datasets

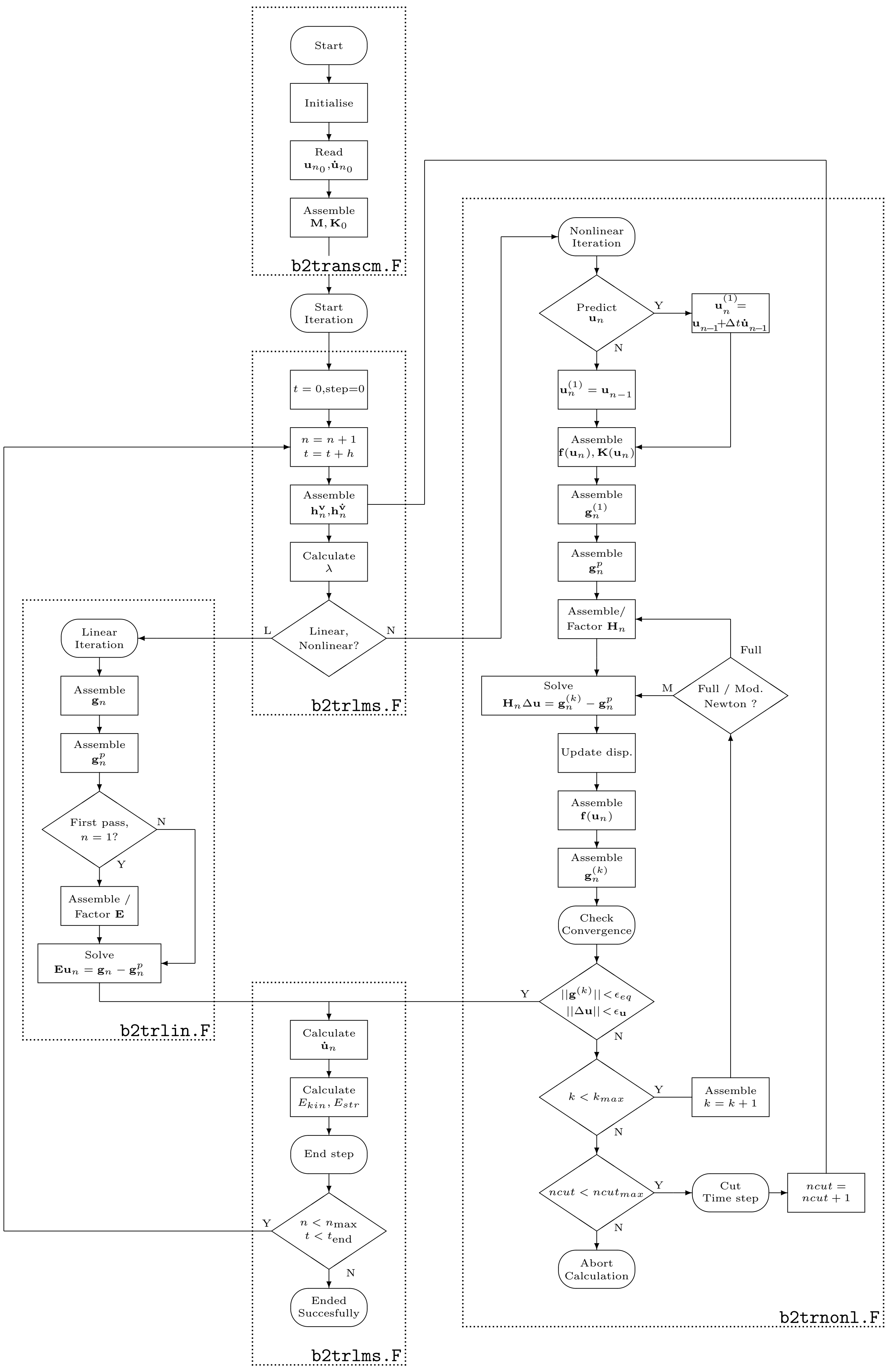
Due to the complicated solution method, a large amount of datasets is created. Most of them are as large as the total number of degrees of freedom, such as the displacement and internal forces datasets. The datasets which contain a sky-lined matrix, such as the mass, the stiffness and the dynamic stiffness matrix (and the Jacobian) are extremely large. In a nonlinear analysis two of these datasets are created at every step, i.e. the stiffness matrix $\mathbf{K}$ and the Jacobian $\mathbf{H}$. In principle, they are not needed in the future and can be deleted after each step. In the older versions of `MEMCOM`, a dataset could not be removed completely. Just the label was deleted. The size of the database did not change. In newer version of `MEMCOM` the database can be reassembled (the empty datasets are removed) using the **`defrag`** command.

B.3 Datasets

Since all numerical operations are executed in the database, every variable in the Jensen algorithm has its corresponding dataset entry. Apart from a number of temporary datasets, named `A`, `B`, `C` and `D` the most important ones are listed in the table below.

Dataset name	Symbol	Description
`ACCE.GLOB.`n	$\ddot{\mathbf{u}}_n$	acceleration at step n
`DDIS.GLOB`	$\Delta\mathbf{u}$	incremental displacement
`DISP.GLOB.`n	$\mathbf{u}_n$	displacement at step n
`EMSS.ADR`		address vector of mass matrix
`EMSS.GLOB.0`	$\mathbf{M}$	band of the mass matrix
`EVAR.ADR`		address vector of dynamic stiffness matrix
`EVAR.GLOB.`n	$\mathbf{E}$	band of dynamic stiffness matrix
`FINT`	$\mathbf{Ku}$	internal forces vector (linear)
`FORC.GLOB..1`	$\mathbf{f}_0^{\text{ext}}$	Unit force vector, $\mathbf{f}^{\text{ext}}(t) = \lambda(t)\mathbf{f}_0^{\text{ext}}$
`FRHS`	$\mathbf{g}^p$	prescribed dynamic force vector
`FVAR.GLOB.`n	$\mathbf{f}_n^{\text{int}}$	first variation vector at step n

GDC.GLOB	$\mathbf{u}_0^p$	unit prescribed displacement vector
GDCW.GLOB		prescribed displacement address vector
HVAR.ADR.n		address vector of Jacobian at step n
HVAR.GLOB.n	$\mathbf{H}_n$	band of Jacobian at step n
RHS	$\mathbf{g}$	right hand side of linear equation (dyn. force)
	$\mathbf{r}$	residue vector
SVAR.ADR.n		address vector of siffness matrix
SVAR.GLOB.n	$\mathbf{K}_n$	band of the stiffness matrix
UHIS	$\mathbf{h}_n^{\mathbf{u}}$	History vector of displacement $\mathbf{u}$
VELO.GLOB.n	$\dot{\mathbf{u}}_n$	Velocity vector
VHIS	$\mathbf{h}_n^{\mathbf{v}}$	History vector of auxiliary vector $\mathbf{v}$ at step n
VHIS.PREV	$\mathbf{h}_{n-1}^{\mathbf{v}}$	History vector of auxiliary vector $\mathbf{v}$ at step $n-1$


Start
Initialise
Read $\mathbf{u}_{n0}$,$\dot{\mathbf{u}}_{n0}$
Assemble $\mathbf{M}, \mathbf{K}_0$
b2transcm.F
Start Iteration
$t = 0$,step=0
$n = n + 1$
$t = t + h$
Assemble $\mathbf{h}_n^{\mathbf{v}}$,$\mathbf{h}_n^{\dot{\mathbf{v}}}$
Calculate λ
Linear, Nonlinear?
L
N
b2trlms.F
Linear Iteration
Assemble $\mathbf{g}_n$
Assemble $\mathbf{g}_n^p$
First pass, $n = 1$?
N
Y
Assemble / Factor $\mathbf{E}$
Solve $\mathbf{E}\mathbf{u}_n = \mathbf{g}_n - \mathbf{g}_n^p$
b2trlin.F
Nonlinear Iteration
Predict $\mathbf{u}_n$
Y
N
$\mathbf{u}_n^{(1)} = \mathbf{u}_{n-1} + \Delta t \dot{\mathbf{u}}_{n-1}$
$\mathbf{u}_n^{(1)} = \mathbf{u}_{n-1}$
Assemble $\mathbf{f}(\mathbf{u}_n), \mathbf{K}(\mathbf{u}_n)$
Assemble $\mathbf{g}_n^{(1)}$
Assemble $\mathbf{g}_n^p$
Assemble/ Factor $\mathbf{H}_n$
Full
Full / Mod. Newton ?
M
Solve $\mathbf{H}_n \Delta\mathbf{u} = \mathbf{g}_n^{(k)} - \mathbf{g}_n^p$
Update disp.
Assemble $\mathbf{f}(\mathbf{u}_n)$
Assemble $\mathbf{g}_n^{(k)}$
Check Convergence
$||\mathbf{g}^{(k)}|| < \epsilon_{eq}$
$||\Delta\mathbf{u}|| < \epsilon_{\mathbf{u}}$
Y
N
$k < k_{max}$
Y
Assemble $k = k + 1$
N
$ncut < ncut_{max}$
Y
Cut Time step
$ncut = ncut + 1$
N
Abort Calculation
b2trnonl.F
Calculate $\dot{\mathbf{u}}_n$
Calculate E_{kin}, E_{str}
End step
$n < n_{\max}$
$t < t_{\text{end}}$
Y
N
Ended Succesfully
b2trlms.F

B2TRANS

This flow chart of the B2TRANS macroprocessor contains all important actions. In order to save space, additional computation, such as the algorithm which updates the rotational increments or the routines to solve the (linearized) system of equations are not included.

Figure B.1: B2TRANS flow chart

C Source Code

The source code of the B2000 platform is available for every user. It is therefor important to keep up a good description of every piece of the syntaxis of the program. In principle, this is done in two ways. The most important is the B2000 Programmers handbook in which the syntaxis of every subroutine is explained with all input flags. Secondly the code is described in the source code itself by using comment statements. This last method is not very official and it is the responsibility of the programmer to maintain the comment-lines in his source code.

All new pieces of source code that have been written during this research are reported in this paper. A short overview of their functionality is given as well. Although the finite element method B2000 is claimed to be fully modular, some new implementations also required changes in existing parts of the source code. In order to prevent miscommunications, all changes to these source files are reported in this chapter.

C.1 2 dimensional beam element, B2.EP

The 2 dimensional, nonlinear beam element B2.EP could be implemented in B2000 very easily. This straightforward element did not require any additional changes in existing source code: it could be inserted as a standard element. The element has got number 88. All source files that are related to this element start with the code b2ep88.

Filename	Description
`b2ipepar.inc`	The new and sizes of the elements are defined in this include file. `enum=88` element number `enam='B2.EP'` element name `nnel=2` number of nodes

`nncl=3`	number of reference nodes
`lefo=12`	number of d.o.f.'s
`ndfn=20`	length of `elprev` array
`flgn=1`	nonlinear element flag

`b2ep88.F`	Main element file. In this routine all the the other routines are invoked. Furthermore, the transformation from Antman's parameters to nodal displacements, as well as the transformation from the element local to the branch global coordinate system is executed here.
`b2epv88.F`	In this file, the prevariational data is calculated and stored in the `elprev(*)` array. The structure of this array can be found in table B.1
`b2ep88elbg.F`	The transformation matrix for the coordinate system transformation is created in this routine. The x and y coordinates of both nodes are used to do so. The actual transformations are done in the main file.
`b2ep88fvar.F`	In this file the first variation vector (for both the Timoshenko and the simplified beam) in terms of Antman's coefficients are calculated.
`b2ep88svar.F`	The second variation matrices for both beams in terms of Antman's coefficients are calculated in this routine.
`b2ep88trans.F`	In this routine the matrices for the transformation of the first and second variation to nodal displacements and rotations are calculated.

C.2 3 dimensional beam element, B2.NL

The implementation of the 3 dimensional nonlinear element has been done using the same procedures as described above. Again the parameters of the element are set in the `b2ipepar.inc` file. The 2 node 3 dimensional element has got number 89. Number 90 is reserved for the 3 node nonlinear beam element, which is due to be implemented in the near feature.

Filename	Description
`b2ipepar.inc`	The new and sizes of the elements are defined in this include file.

`enum=89`	element number
`enam='B2.NL'`	element name

`nnel=2`	number of nodes
`nncl=3`	number of reference nodes
`lefo=12`	number of d.o.f.'s
`ndfn=6`	
`lepr=20`	length of `elprev` array
`leuf=55`	length of `elurf` array
`flgn=1`	nonlinear element flag

`b2ep0.F` The element routines are invoked in this kernel routine. There are 4 additional arguments (among which the incremental displacement `ddis(*)` added the list for the `b2ep89.F` element. The syntaxis as used by G.Rebel for the implementation of his finite rotation shell elements is used.

```
 1089 call b2ep89(coor, disp, etrans,
     *       eprop, epropall, elaminates,
     *       elprev, elurf, elfvar, elsvar, elstab,
     *       eltfor, ellfor, plasold, plasnew,
c-JR
     *       ddis, d, delfvar, delsvar,
c-JR
     *       work, irad, istat)
```

`b2ep89.F` Main file for the 3 dimensional beam element. In this file the numerical integration as well as the complete construction of the first and second variation matrices are tackled. Also the transformation from element local to branch global and vice versa.

`b2epv89.F` The prevariational package of the element is created in this routine. The contents of the `elprev(*)` array is given in table B.2.

`b2arr2tens.F` This routine transforms the aixal vector into its skew symmetric tensor form.

`b2tens2arr.F` This routine performs the inverse operation of the previous routine. A skew symmetric matrix is transformed into a axial vector.

`b2ep89cspat.F` The constitutive matrix $\hat{\mathbf{C}}$ is created in this subroute. The matrix is first written in the moving frame. It is transformed into the spatial base with use of the rotation tensor $\mathbf{\Lambda}$.

`b2ep89gstif.F`	The geometric stiffnes matrix $\mathbf{B}$ is created in this routine.
`b2ep89xi.F`	Construction of the first differential operator $\boldsymbol{\Xi}$ according to equation (3.102).
`b2ep89psi.F`	Construction of the second differential operator $\boldsymbol{\Psi}$ according to equation (3.97).
`b2epbeamrot.F`	The calculations of the updated rotation tensor, as well as the spatial strain and curvature vector are done in this subroutine. This routine can also be used by the feature 3 node element
`b2multmtv.F`	This subroutine is a variant of the `b2multmv.F` and is able to multiply a transposed matrix by a vector.

C.3 Transient Solver B2TRANS

In this section the changes to the input processor and the new source file for `B2TRANS` are listed.

Filename	Description
`b2ipdyna.F`	The dynamic strategy parameters are extended with a number of new variables, such as `DT` for the initial timestep, `METHOD` to choose the LMS scheme that must be used, variables to determine the Rayleigh constants, `ALFA` and `BETA` and variable to determine the presence of the initial displacements, initial velocities and force function.
`b2maputil.F`	A new history function is added in this routine. Apart from the trigonemetric functions and step functions, the `SLOPE` function is available. The slope function is a linear increasing (decreasing) function.
`b2pclmap.F`	The PCL input command for the slope function is added.

Created source files

Filename	Description
`b2trans.F`	Main file. In this file all the memcom data base commands are intialized, the input is read and the actual calculations are started.
`b2transcm.F`	Command module. In this subroutine the actual transient calculations are executed. Within this source file, a number of specific subroutines can be distinguished. The calculations are initiallized in the main routine is `b2transcm`. The history vectors are calculated in `b2trlms`. The Jensen equation is also set up in this routine. The linear or nonlinear system of equation is solved in `b2trlin` or `b2trnonl` respectively.
`b2getdyna.F`	Memcom related file. The dynamic analysis parameters are read from the dataset `DYNA` in the archival database and stored in common blocks.
`b2transdyna.F`	In this file the dthe PCL are read.
`b2putdyna.F`	PCL related file. The dynamic analysis parameters obtained in the PCL command line are stored in the dataset `DYNA`.
`b2setdynasol.F`	In this routine all additional data of a single timestep, such as current time, kinetic energy and convergence status is written to the descriptor table of the idsplacement data set `DISP.GLOB.`n.

C.4 Miscellaneous

A number of files have been altered for various purposes. Most of the files contain the element mass matrix description for the cable and the fintie rotation shell elements.

Filename	Description
`b2mp.F`	Previously, in the construction of the mass matrix, the element prevariational package could not be used. After a small adjustment, the dataset `PREV` is read and stored as the array `elprev(*)`.

`b2mp39.F`	Mass matrix of the 2 node nonlinear cable element (`C2`) This reduced lumped diagonal mass matrix is stored as a consistent mass matrix in order obtain a skylined mass matrix of the same size as the stiffness mastrix.
`b2mp91.F`	Mass matrix of the `Q4N.REBEL` 4 node shell element, see `b2mp39.F`
`b2mp92.F`	Mass matrix of the `Q8N.REBEL` 8 node shell element, see `b2mp39.F`
`b2mp93.F`	Mass matrix of the `Q9N.REBEL` 9 node shell element, see `b2mp39.F`