INSTRUCTIONS FOR

COMPUTER LABS

SD2411 Lightweight Structures and FEM

Contents

Overview

Teachers

Stefan Hallström - stefanha@kth.se
Dan Zenkert - danz@kth.se

Lab assistants

Sahar Akbarpour
Karl Bouton
Tomas Ekermann
Ross Harden
Wilhelm Johannisson
Johan Larsson
Anton Shipsha

Schedule

You sign up together with a partner for one of the lab groups. Each group has one scheduled two-hour lab session per week. The computer exercises start with a short introduction and from then you are supposed to start working with the problems in parallel with the classes. You have 24/7 access to the computer hall, except when it is busy with other scheduled classes. The scheduled lab hours for your group are primarily there for your assistance, allowing you to ask for explanations or advice if you get stuck. As you may already know, debugging your code can be quite tiresome and if you fail to find the problem the assistants may have some hints or advise to help you get around the problems you encounter. The assistants could also be reached by email, or through KTH-Social, see above. All details about times and schedules are found in the course programme.

Introduction

During the course theory you are supposed to learn analysis of thin-walled beams. Later in the course, as part of this lab exercise, you are asked to apply the theory to a certain beam problem. In the mean time you are expected to write a finite element (FE) code of your own using MatLab and learn the basics of using a commercial FE code. In the end you are supposed to compare and think about the similarities and differences between the three different approaches to the beam problem. In this document you find instructions for the labs. You will also probably need to study the FEM lecture notes while working with the lab tasks.

In exercise 1 you are to write your own FE program using MatLab. The code should handle beams subjected to bending and torsion (St.Venant). It shall also be able to handle stability in terms of Euler and torsion buckling. Your own code will also be used later, in exercise 2.

In exercise 2 the commercial FE package ANSYS is introduced. First you will work with the example given in Appendix A in this compendium. It contains rather detailed instructions on how to work with ANSYS and after completing the exercise you have more or less finished the input file that is necessary for the work in exercise 2.

In lab 2 a specific beam problem is introduced. It is a thin-walled cantilever beam with a U-profile cross-section, subjected to a point load at its free end. The beam could show combined bending and torsion or various types of buckling, depending on the geometry and the applied load. When the model has been completed (see Appendix A), it shall be used to calculate stresses, displacements and illustrate how the beam deforms due to the applied load. In a second stage you will also do a buckling analysis. In parallel you are supposed to derive corresponding results analytically using the methods presented in class. Some things will agree very well between the ANSYS output and your analytical calculations, while other things will not. Your job is to figure out how and why the results differ and if anything could be done to improve the correlation or to improve the results in general. Which results are reliable and why?

Your MatLab and ANSYS models both have similarities and differences. They handle the same problem, but different methods are used. In your own MatLab FE-code the problem will be solved using engineering beam theory, assuming St.Venant torsion. (Which are the assumptions made in St.Venant theory?) The model is one-dimensional and specific features of the cross section are handled through engineering constants, such as cross section area, A, and moments of inertia, I. With ANSYS the problem is solved through modelling of the entire beam profile with shell elements, i.e. even the shape of the beam cross section is modelled. In this case, the problem is no longer one-dimensional, but rather an idealised three-dimensional problem. The finite elements used are however only 2-dimensional. How does this affect the results?

Background

The exercises deal with bending, torsion and buckling of thin-walled beams. Analyses of this kind are very common in real life. Ships, for example, are designed and analysed using beam theory; the whole ship is treated globally as a stiffened thin-walled beam. In all the lab cases, the studied beam is a cantilever. This may seem overly academic perhaps, but may still have quite useful applications. The wing of an aircraft may be treated as a cantilever, since it has a symmetry plane along the centre of the fuselage. The aerodynamic forces on the wing create lift, or more specifically a pressure distribution over the wing, as shown in Figures 1 and 2. An aerodynamic wing on a race car works in the same way although the boundary conditions are different. For those of you interested in naval applications there are many similarities with a keel blade of a sailing yacht or a rudder.

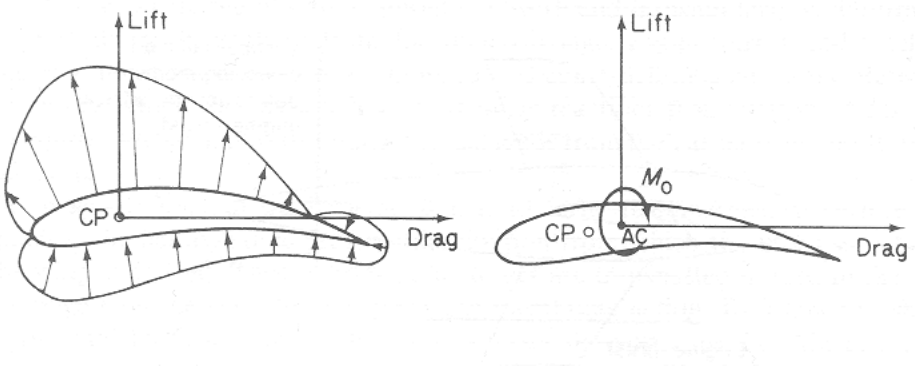


Figure 1. Pressure loads on a wing, looking at a cross section

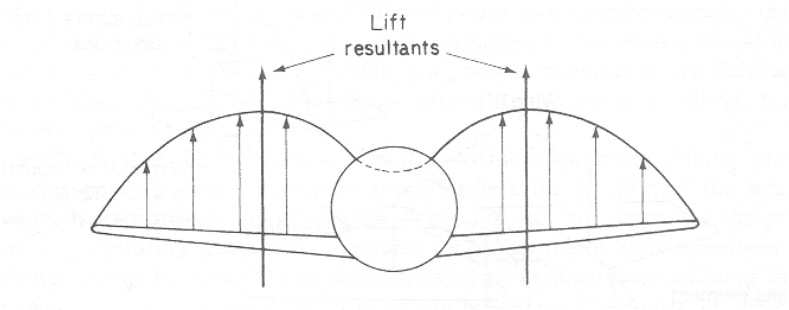


Figure 2. Pressure loads on a wing, looking along the fuselage

The pressure distribution over the wing generates resulting vertical (lift) and horizontal (drag) forces. One can integrate the pressure distribution of the cord of the wing to achieve resulting equivalent lift and drag forces, and a resulting torque (M_0 in Figure 1). In the structural analysis we then have a cantilever beam, since the symmetry plane provides constraints similar to clamped boundary conditions along the centreline. The beam is in principle a stiffened, thin-walled shell, with a bending and torsion stiffness that varies over the length (x) as $EI = EI(x)$ and $GJ = GJ(x)$. A real wing is usually constructed of either a wing beam, which carries the entire load, and an outer shell providing its aerodynamic shape, or from a stiffened thin-walled shell integrating both functions. The beam is subjected to a varying flow that generates a distributed pressure, acting to lift the wing, and a torque, acting to twist it. The FE code you are about to write does not solve this problem entirely, but could do so with only minor modifications – think about how this could be done!

Buckling of a thin-walled beam also has a link to the wing beam problem. If the wing is built as a stiffened, thin-walled shell structure, the upper side of the wing is subjected to large compressive forces that may lead to buckling of the face and the stiffeners, either globally or locally.

Goals

One goal with the labs is to get hands-on experience of FE analysis. ANSYS is one of several commercial multi-purpose FE packages you might come to work with in your future professional life. Another aim is to provide insight into how engineering problems can be approached and solved using FEM. The most important learning outcome from the exercises is that the FEM can be a very powerful tool for someone who knows both FE and structural mechanics, but also that a FE program hardly can be efficiently or accurately used by someone who does not understand the problem at hand. Never forget that using FE in practice is to run a computer code, written by someone else, and the output results only become as accurate as the given input provided by the user. We hope that these exercises help building an understanding for problems in structural mechanics, existing solution methods, how they differ and which results one can expect from different approaches. We believe that this type of knowledge and understanding is essential to convey in the education of professional masters in engineering.

Which problems should be solved?

As already mentioned, you are supposed to complete two lab tasks working in pairs. You are to complete the exercises and solve the problems together, and finally write and hand in brief lab reports using templates provided on the course web site.

To choose the cross-section for the cantilever in the labs, take the last digit in the birthday of the person in your group who is the youngest (for example 3, if he/she was born 1995-07-23). If the last digit is zero, you pick section number 10. The table below provides cross-section data to use.

Choices of beam cross-sections

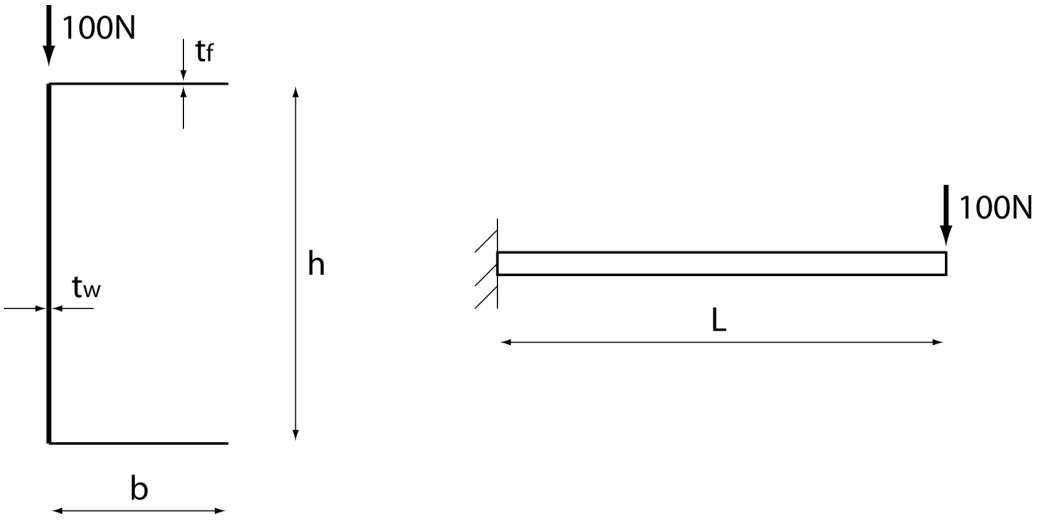


Figure 3. Cross-section and load case

The beam length is 1000 mm, and the applied load is S=100 N, but you can of course apply any load you wish. Remember however that your analyses are all linear, i.e. if you double the load, the displacement, stresses, etc. all double. The beam is made of aluminium with E = 70000 MPa and ν = 0.3. (Consistently defining your input in *mm* and *MPa* will give you stress output in *MPa*. The alternative is to give all input data in SI-units, i.e. *m* and *Pa*.)

#	b [mm]	h [mm]	t_f [mm]	t_w [mm]
1	20	20	3	1
2	20	20	1	3
3	20	40	3	1
4	20	40	1	3
5	20	60	3	1
6	20	60	1	3
7	40	40	3	1
8	40	40	1	3
9	40	60	3	1
10	40	60	1	3

You may assume in all your calculations that t_f and t_w are much smaller than b and h.

Lab 1 - Your own FE-code

You are about to write your own FE-code in MatLab, using beam elements and solving bending/torsion and also axial buckling of a cantilever beam. At the *Canvas* activity under *Files* you find a number of MatLab *m*-file templates. In the templates comments, variable names and function calls are given. Use the program template files and use only the default variable names and function calls given therein. If not, it will be very difficult for anyone else to help you find errors in the code if you make mistakes!

If you have limited experience of MatLab programming there is also a good pdf-introduction to the software from in the *Document* folder, courtesy of the University of Dundee.

You may also use this program again in later courses and extend it for other purposes, and to avoid problems it is recommended to use the given notation. In order to verify the code, analyse a few standard elementary beam cases and compare the results with solutions from engineering formulas, e.g. in the formula handbook in solid mechanics, both for concentrated loads, uniformly distributed load and torque. If the results from your code do not agree with these solutions, something is wrong with your code!

Another load case you should consider is axial compression for which the critical load at buckling shall be determined. For this type of thin-walled cross-sections, different buckling modes can appear and your MatLab code should be able to handle Euler type buckling and torsion buckling. The buckling analysis you will perform is an *eigenvalue buckling analysis* that also could be performed in most commercial FE codes.

Element formulation

The two-node beam element has three degrees of freedom (dof's) per node so that both bending and St.Venant torsion can be handled, see Figure 4. Observe that differences between definitions of positive directions matter and that such differences exist between different codes and literature.

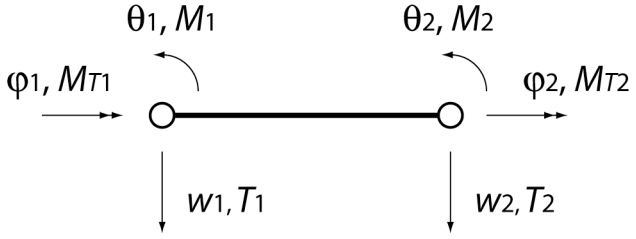


Figure 4. Definition of beam element

Assume displacements and load vectors as

$$\boldsymbol{u} = (w_1,\ \theta_1,\ \varphi_1,\ w_2,\ \theta_2,\ \varphi_2)^t \text{ and } \boldsymbol{F} = (T_1,\ M_1,\ M_{T1},\ T_2,\ M_2,\ M_{T2})^t$$

and the element shape functions as

$$N_1 = 1 - 3\left(\frac{x}{h}\right)^2 + 2\left(\frac{x}{h}\right)^3,\ N_2 = -x + 2\frac{x^2}{h} - \frac{x^3}{h^2},\ N_3 = 1 - \frac{x}{h}$$

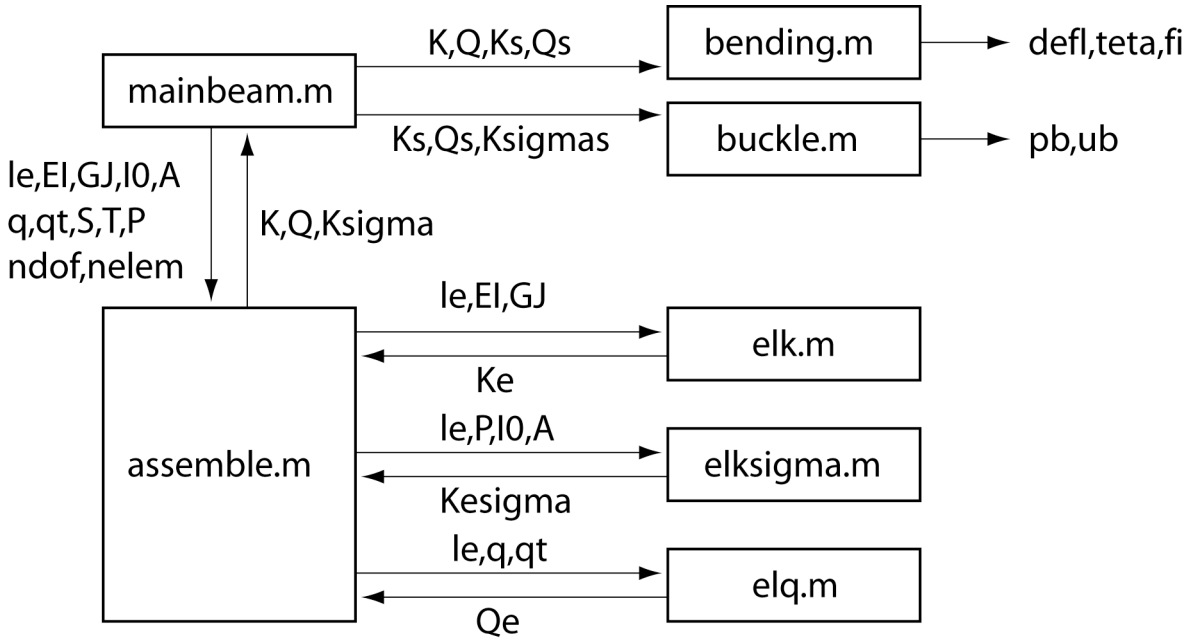
$$N_4 = 3\left(\frac{x}{h}\right)^2 - 2\left(\frac{x}{h}\right)^3,\ N_5 = \frac{x^2}{h} - \frac{x^3}{h^2} \text{ and } N_6 = \frac{x}{h}$$

The stiffness matrices needed for the elements can be obtained by synthesis of matrices in the FE lecture notes. Simply add components related to bending and torsion to the correct degrees of freedom. For static bending and torsion it will appear as

$$\boldsymbol{K}^e = \begin{bmatrix} \frac{12EI}{h^3} & -\frac{6EI}{h^2} & 0 & -\frac{12EI}{h^3} & -\frac{6EI}{h^2} & 0 \\ -\frac{6EI}{h^2} & \frac{4EI}{h} & 0 & \frac{6EI}{h^2} & \frac{2EI}{h} & 0 \\ 0 & 0 & \frac{GJ}{h} & 0 & 0 & -\frac{GJ}{h} \\ -\frac{12EI}{h^3} & \frac{6EI}{h^2} & 0 & \frac{12EI}{h^3} & \frac{6EI}{h^2} & 0 \\ -\frac{6EI}{h^2} & \frac{2EI}{h} & 0 & \frac{6EI}{h^2} & \frac{4EI}{h} & 0 \\ 0 & 0 & -\frac{GJ}{h} & 0 & 0 & \frac{GJ}{h} \end{bmatrix}$$

You will have to do this yourself for the initial stiffness matrix used for buckling calculations, synthesising the matrices for bending and torsion in the same way as done in the original stiffness matrix above. Remember also that you might have to change the value of the moment of inertia. Euler buckling occurs about the axis that has the lowest bending stiffness and thus the lowest moment of inertia (the smallest of the two principal moments of inertia).

Structure of program



Write your program this way:

mainbeam.m main program where input data is given. This module calls for *assemble.m*, *bending.m* and *buckle.m*.

assemble.m	assembles structural stiffness matrices and load vectors. This module in turn calls for *elk.m*, *elksigma.m* and *elq.m*.
elk.m	calculates the element stiffness matrix, K_e (6×6).
elksigma.m	calculates the initial stiffness matrix, $K_{e\sigma}$ (6×6), including torsion buckling components).
elq.m	calculates the element load vector (6×1).
bending.m	calculates the displacements of the beam and prints/plots the results. Also calculates the reactions forces at the boundary.
buckle.m	calculates the buckling load(s) and the corresponding buckling mode(s) for the beam and prints/plots the results.

One benefit with this type of program structure is that you can quite easily add mass matrices (*elm.m*) and calculate eigenfrequencies and eigenmodes for the beam (*vibrate.m*). (This is for instance done in the course SD2810 – Aeroelasticity.) The structure also allows for change of stiffness matrices and element types without making it necessary to rewrite all subroutines.

Input

le – element length
EI – element bending stiffness
GJ – element torsion stiffness (St.Venant)
S – *point load*
T– torque
P – in-plane load (let *P*=–1)
I0 – polar moment of inertia
A – cross-section area
q – pressure load (line load)
qt – distributed torque
nelem – number of elements

Output

ndof – number for degrees-of-freedom
Ke – element stiffness matrix (6×6)
Kesigma – element initial stiffness matrix (6×6)
Qe – element load vector (6×1)
K – structural stiffness matrix (ndof×ndof)
Ksigma – structural initial stiffness matrix (ndof×ndof)
Q – structural load vector (ndof×1)
Ks – reduced structural stiffness matrix (ndof-3×ndof-3)
Ksigmas – reduced structural initial stiffness matrix (ndof-3×ndof-3)
Qs – reduced structural load vector (ndof-3×1)
defl – displacement vector (ndof×1)
teta – rotation vector (ndof×1)
fi – twist vector (ndof×1)
pb – diagonal matrix with load factors (ndof×ndof)
ub – matrix of eigenvectors (ndof×ndof)

The code shall be able to do the following:

1. Use an arbitrary number of elements. These elements do not necessarily have to have the same length, but you may use elements of the same length over the entire beam for simplicity.
2. Handle uniformly distributed pressure *q* over the entire beam, a concentrated load *S* at the end of the beam, a torque *T* at the beam end or a distributed torque *qt* over the entire length of the beam. It is not difficult to apply more general load cases, but it is not required in this exercise.
3. Calculate the deflection due to applied loads.

4. Calculate the twist due to applied torque.
5. Calculate the reaction forces at the support, due to the load cases above.
6. Calculate the critical buckling load and buckling mode for various beam lengths.

Points 1-6 will be discussed in the oral examination. **Make sure to complete the report template form for Lab 1 before examination.**

Further points for thought and discussion

As mentioned your program shall be able to calculate the response of a cantilever beam with a constant cross-section, subjected to for instance a distributed load q or a concentrated load S. For the wing beam, the problem is slightly different, the beam is tapered with varying bending and torsion stiffness and the load varies over the length of the beam. Think about how you could update your program to handle this! Can you do something about the element formulation, the element sub-division or the application of the load? It should require a limited effort to include this, however not necessarily automated! If you wish to solve another beam problem than the cantilever, how would you then alter your program? Are there any limits in what your program then could handle?

For oral examination of this lab, you must have benchmarked your program through comparison with known analytical solutions and verified that agreement is obtained. You should also be prepared to discuss what the program can and cannot do, and how it could be extended to handle more general beam problems.

Lab 2 - Cantilever with thin-walled cross-section

The problem at hand is a thin-walled cantilever subjected to a concentrated load at its outer edge. A clamped boundary condition, as stipulated in this problem, is hardly practical but may be a good approximation for a symmetry plane. (Concentrated forces like this are not practical, but rather used for convenience.)

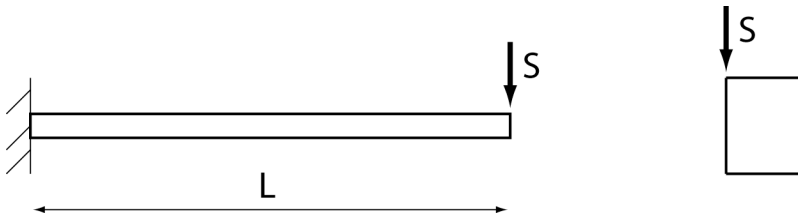


Figure 5. Cantilever beam

It is here intended that you work your way through the ANSYS exercise in Appendix A before you continue with the problem formulation given below.

1. **Analytically** calculate the shear flow *q(s), the location of the shear centre and* the displacements at the edge of the beam (*u(L)*, *v(L)* and *φ(L)*). It may be convenient to use Maple for this. (Observe that the simplified expression for the location of the shear centre in Megson's Example 17.3, does not apply since the thickness varies over the cross section!) For which point of the cross section are the displacements valid?

2. Calculate the corresponding displacements and twist using **your own MatLab FE-code**.

3. Make sure that your ANSYS file works as it should and that all input parameters are correct.

4. Make sure that the load is applied correctly, see Appendix A.

5. Evaluate the displacements u and v, and φ for one point on the web at $z=L$. How should the results be compared with the analytical results and the results from MatLab?

6. Calculate the total strain energy from the FE-solution. Compare the strain energy with the work of the applied loads in the ANSYS and MatLab models.

7. Evaluate the normal and shear stress distributions along the cross-section in the middle of the beam, i.e., for $z = L/2$, and for $z = 0$ at the support. See Appendix A. How do they correlate with analytical results?

8. Calculate the warping along the cross-section at $z = L$. See Appendix A. Can you get the actual warping directly from nodal displacements?

9. Pick the point (x,y) on the cross-section with greatest warping and draw a diagram over how the normal stress along the beam varies from $z = 0$ till $z = L$. This can also be done by

using *Path Operations*. What should the FE-mesh look like in order to best model the obtained stress distribution?

10. Now, compare all your obtained results; the analytical, your own MatLab FE-code and the ANSYS output results. Some properties, displacement for example, will compare rather well, while other will not match at all. One of the main issues in this lab is to discuss why some results compare well and other do not? What differs between the different solution methods? Which results are correct?

At this stage it is a good idea to make a copy of your ANSYS input file and save it as a backup before you proceed with the stability analysis. The buckling analysis you now will perform is a so called *eigenvalue buckling analysis* which could be executed with most commercial FE-codes.

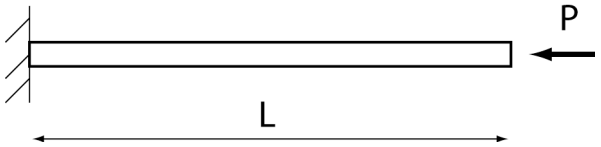


Figure 6. Buckling of column with thin-walled cross-section.

11. Calculate the buckling load for the column analytically. Remember that there are three potential buckling modes to consider; Euler buckling, local buckling and torsion buckling. For some cross sections combined Euler/torsion buckling could also occur.

12. Compare the results from your own FE-code and the analytical results.

13. When you continue with ANSYS, you can use the same geometric model of the problem. Just remember to remove the loads used in the bending case before you run the stability analysis. Apply the new load, ensure that it is placed correctly, has the right magnitude and the correct direction. In this problem the SFL command works, check the syntax in the manuals. To simplify things it is practical to apply a unit load (P = 1N). ANSYS calculates a load factor (just as your MatLab FE-code) that is a multiplier of the applied load. It may be good to check that the load is applied correctly by running the problem statically, without the buckling options, and check the total reaction load, which then should be 1N.
    Note that when applying the load P to the lines at z=L, it should be distributed so that the stress becomes equal in all three walls. Since the load is applied as load per unit length you need to take the different dimensions of the web and the flanges into account. The following code handles that:

```
fracflange=(b*tf)/(h*tw+2*b*tf)      ! Calculates the fraction of the total cross
fracweb= (h*tw)/(h*tw+2*b*tf)        ! section area for the flanges and the web
presline4=(P/b)*fracflange           ! Calculate the line loads for lines 4-6
presline5=(P/h)*fracweb
presline6=presline4                  ! The applied load on lines 4 and 6
SFL,4,PRES,presline4                 ! is equal due to symmetry
SFL,5,PRES,presline5
```

SFL,6,PRES,presline6

In this way the total applied load is equal to *P*. Define *P* (e.g. *P* = 1) at the top of the file where all other parameters are defined. Apply an /EOF command, run the problem and check the value of the reaction force FZ, using *List>Results>Reaction solution*, and make sure that it is equal to 1N. Once you have applied the buckling commands (§15), the reaction force is no longer available in the solution. (Why?)

14. Apply the necessary boundary conditions.

15. To include eigenvalue buckling in the analysis you need to add the following to your input file:

```
After /SOLUTION write
PSTRES, ON              ! Saves initial stresses and creates an initial stress matrix
SOLVE
```

After the calculation part (FINISH) you need to add:

```
/SOLUTION               ! To solve the new problem
ANTYPE, BUCKLE          ! Defines the type of problem to be solved
BUCOPT, LANB, N         ! Defined the type of solution method to be used and how
                        ! many eigenvalues (N) that should be computed
MXPAND, N               ! Necessary command to be able to plot the solutions
SOLVE                   ! Solve the problem
FINISH                  ! Finish the calculation
```

Note that *N* should be a number, e.g. 5. You should preferably use the same value for *N* in both the BUCOPT and the MXPAND commands. A good way is to define *N* as a variable, and then use it as in the example above. Following the /POST1 command, the solution is shown using:

```
SET, FIRST              ! Activate the first mode
PLDISP, 2               ! Plots this mode together with the undeformed beam shape
```

When the input file has been checked and seems to be correct, solve the problem (SOLVE). Every time you wish to restart, click *file>clear & start new* and then *file>read input from* and choose the input file name from the list.
Go to *General Postproc>Plot results>Deformed shape*. This will show the buckling mode shape. The text top left on the screen gives the follow information;

```
STEP=1          (Load step number 1)
SUB=3           (Substep number 3, eigenvalue number 3)
FREQ=xxxx       (Eigenvalue – load factor for this substep)
```

Now press *General Postproc>Read Results>Next step*. Then go to the *Plot* menu on the top line and choose *Replot*. SUB=1 on the screen corresponds to the lowest buckling mode shape with its corresponding load factor.

16. Vary the length of the beam and compare how the buckling loads and buckling modes vary analytically, in your MatLab FE-code and in ANSYS. In ANSYS the length is varied by changing the variable controlling the length in the input file. You may have to change the number of elements along the beam in order to avoid element distortion. The side aspect ratio (small side length/long side length) should preferably be within 1:4 and should not exceed 1:10. You do this best by changing the LESIZE commands. Plot the shape of the model before you run it to check the side aspect ratio.

17. Calculate the buckling loads. In ANSYS they are given as *load factors*, i.e., multipliers of the applied load and if your applied load is 1N, then the load factor output will equal to the calculated buckling load. There are as many buckling loads calculated as given in the *BUCOPT*-command and as many buckling modes as given in the *MXPAND*-command, so try to use the same number *N* in both these commands. In the example above, there are then *N* eigenvalues (buckling loads) and *N* eigenvectors (buckling modes). Each eigenvalue (buckling load) has a corresponding eigenvector (buckling mode) which then is a displacement vector, or vector of nodal displacements (including rotations and twist). This eigenvector can be plotted to examine if the buckled shape corresponds to Euler, local or torsion buckling.

18. Now you can compare all results; analytical, your own MatLab FE-code and ANSYS results. Some load factors (eigenvalues) with their corresponding buckling modes will agree well and some not well at all. What makes the results good or bad? Which results are most reliable? Is any of the models complete and entirely accurate?

Remember to save or move the important files, input file(s) and plots (e.g. .jpg), from the local directory to your own home directory.

Final comment

The aim with this lab is only to give a short introduction to problem solving using ANSYS. Obviously you don't become an expert in FE analysis by running one example. However, the only efficient way to master FEM is to practice and gain experience. A condition for solving a problem with FEM is that you understand it and are able to set it up properly in the FE software. There are many potential sources for errors. One may be numerical errors but those are usually small, provided that your elements look good and are sufficiently many. FEM is by definition approximate and the elements used are relatively primitive. The most serious errors are usually modelling errors due to poor input or misconceptions from the user. Is the material data accurate? Is the mesh sufficiently fine to give a good approximation of the real problem? Are the deformations in the model really small enough to use linear theory? Are there stress concentrations or details in the model that require further mesh refinements? How is the load applied? Does the model represent all aspects of the real problem? Are the boundary conditions realistic? A responsible engineer always makes sure to be on the safe side and checks all results carefully.

It's not about solving anything exactly,

it's about minimizing the error!

Appendix A - Tutorial for lab 2

Introduction

This tutorial is meant to a brief introduction to ANSYS for the second part of the FE lab in AVE's student computer room. It is not intended to explain everything about ANSYS or all the different commands one can use, but simply to help you getting started with the labs. ANSYS is after all, and it may be hard to believe in the beginning, one of the more user friendly commercial FE-codes. The program is supplied with a comprehensive help tool on-line, which may be very useful to address from time to time. If you read this tutorial while working with the lab at the computer it should give you a good start.

Different ways of working with ANSYS

There are in principal two ways to work with ANSYS. Either you work only with graphical user interface (GUI) and construct the model step by step. The other way is to write commands in a text file, a so called *input file*, outside the ANSYS program, which is then read as an input file in ANSYS and executed through the GUI. The method of creating the FE model in the GUI has some drawbacks compared to making it in an input file. Primarily, one is then not in control of what one is doing and changes are more difficult to implement. If you instead are working with an input file, you can control things (names of points, lines, etc.) and it is easy to work with models that use parameter input (variables such a thickness, lengths, etc.). In the beginning it may feel backwards using this approach by typing line commands into a file to build up the geometry, but it will show advantageous and is actually a more "professional" approach.

When modelling very complicated shapes, it may be better to make the geometric model in a CAD program, and then transfer the geometry to an ANSYS input format. Many of the large CAD systems have modules for transferring geometric models to various commercial FE-packages.

The build-up of the input file

The input file is a pure text file in which you write ANSYS commands in sequence. The file contains three main parts;

- The first part is the pre-processor (PREP7) in which you input data for the problem, for example, material parameters and element types. You also define the geometry in the pre-processor.
- The second part is the solution procedure (SOLUTION) where loads and boundary conditions are defined, and finally tells ANSYS to run the calculation.
- The final part is for the post-processing (POST1) in which you can extract results, plot and print result lists.

When working along with the input file, it is convenient to add an /EOF (End Of File) command at suitable positions. This tells ANSYS to stop reading the input at this position. One can thus run parts of the input and check that things have been correctly implemented. Once everything has been cleared, the /EOF command is removed.

Input file for the beam problem

In this section we will go through the build-up of the input file for the beam problem as an example for ANSYS input. The aim is to in a detailed manner show how this input file can be written, explain some useful commands and give examples of how these commands are used. The file is given in a more complete format at the end of this appendix.

Introduction

ANSYS is a commercial FE-package. While running, ANSYS stores information, such as stiffness matrices, in database files. Depending on the size of the problem to be solved (number of degrees-of-freedom) these files can become rather large. Once the calculation is done, the results will however not take up too much disk space. In order not to exceed the personal disk quota it is best to run ANSYS in a directory on the local disk on the computer. This also saves a lot of time since all I/O is done locally and not over the network. Once you start ANSYS, a local directory will be created named /tmp/*yourusername*. If you have already prepared an input file, copy it to this local directory. Open a terminal window and write

>> cd /tmp/*yourusername*
>> ansys

ANSYS will be launched working from this local directory. Make sure that when ANSYS starts, it will be in this local work directory. You can change the current work directory with *File>Change Directory*. **When you are finished, remember to copy or move all necessary files back to your personal directory – the directory /*t*mp/ is erased automatically every 24 hours.**

The actual work with the input file is done in an ordinary text editor, e.g. Notepad++ or *emacs*. You start emacs by typing "emacs" in a terminal window. It may be preferable to type "emacs &" (with a space between *emacs* and &) so that the terminal window may be used in parallel. You should always try to comment things you write in the input file, so you should start by writing what the file contains, a name of the problem, your own name and a date. Comments are given in ANSYS by printing an exclamation mark (!) before the comment is inserted and anything behind the exclamation mark is ignored by ANSYS. Comprehensive comments makes it much easier to find bugs, remember how things have been solved and for other persons to work with the same file.
Create a new file with the command "emacs *filename*.mac &". A new emacs window will appear on the screen. The extension *.mac* makes ANSYS treat the entire file as a macro. Start by writing your initial comments at the header of the file.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Cantilever beam in lab 2
!  by Anna and Bob
!  2006-09-01
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

To be able to run the file again after e.g. changes have been made to the file it can be good to include the following lines after the initial comments.

```
FINISH                    ! Finishes the current job
```

```
/CLEAR                  ! Clears the memory in ANSYS

/PREP7                  ! Starts the modelling part of ANSYS (pre processor)
/TITLE, Lab2            ! Gives the run the name Lab2
```

This starts the pre processor (PREP7) and assigns the name “Lab2” to the run. ANSYS does not differ between lower and upper case letters, but capital letters seems to be easier to read on the screen. Thus, /TITLE and /title are interpreted the same way by ANSYS.

Variables
It is convenient to place a separate section, containing all model parameters, at the beginning of the file. In this way you will not need to search for the variables when you wish to make changes or adjustments to the input data. A variable is given a value with the syntax

variable name = value

Variable names may be chosen arbitrarily and can contain blanks. However, avoid using exclamation marks (!) or names that corresponds to standard ANSYS commands, such as “prep”, “title” or “ex”, since this could cause problems.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Parameters
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
h=50                    ! Web height, in mm
b=25                    ! Flange width, in mm
tw=2.5                  ! Web thickness, in mm
tf=1.5                  ! Flange thickness, in mm
L=1000                  ! Beam length, in mm
S=-100                  ! Load, in N
number=2                ! Factor that determines element mesh
```

Materials
You can define materials in several different ways. In our example, there is only one material, and then the best way is to define this directly in the file. The material properties are given using the command “MP” (Material Properties) and the syntax is

MP, material property, material number, value

The material properties have specific abbreviations, such as EX – elastic modulus, or DENS – density. These are given in the element manual. If you work with isotropic materials, you only need to specify the elastic modulus (EX) and the Poisson ratio (NUXY). ANSYS will then calculate the other properties, such shear modulus, bulk modulus, etc. since they can be extracted from the input data. Note that in English a decimal point is used, not a decimal comma, as in Swedish.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Material definition and parameters
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Ea=70000                    ! Elastic modulus for aluminium (variable definition)
nua=0.3                     ! Poisson ratio
MP, EX, 1, Ea               ! Gives EX for material 1 the value Ea
MP, NUXY, 1, nua            ! Gives NUXY for the material 1 the value nua
```

In this example, the parameter description is not really necessary. You could just as well write ”MP, NUXY, 1, 0.3” in one single line!

Element type

Next you should define the type of elements to be used. In this lab you will use SHELL93, which is an 8-node shell element (what is a shell element?). It uses quadratic interpolation (shape) functions, since it has 8 nodes, and has 6 dof’s per node, three translations and three rotations (UX, UY, UZ, ROTX, ROTY and ROTZ). There is a specific manual in which all elements are described (Elements Manual), but it may be better to use the on-line help manual (choose “help” from ANSYS menu), when you check the element properties. The syntax for the definition of elements is;

ET, local number, element type number

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Element type
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ET, 1, 93                   ! Element type 1 is SHELL93
R, 1, tw                    ! Web thickness
R, 2, tf                    ! Flange thickness
```

In this section, you also define so called *Real Constants*. These are element specific parameters and different elements require different real constants. For a beam, it could be bending stiffness, height, width, etc. and for SHELL93 it is only the thickness of the shell that is required. However, more real constants can be given for the SHELL93 element, but here we only need to specify the thickness. For example, “R, 1, tw” gives real constant number 1 the value tw (which is the web thickness as defined previously) in its first position. For shell elements, the first position is always the thickness, but since shells can have varying thickness you can give thicknesses for all four corners and then you need to give values in more positions. Read the manual for more information.

Numbering

This section does not influence the result in any way, but seems suitable to include for clarity.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Numbering
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/PNUM, KP, 1                ! Point numbering flag, 1=on, 0=off.
/PNUM, LINE, 1
/PNUM, AREA, 1
/PNUM, NODE, 0
```

/PNUM, ELEM, 0

If the command /PNUM is active, i.e., set to 1 for some group, for example lines (LINE), then ANSYS will write out the line numbers in all plots. This is useful when you are working with the build-up of the model

Geometry
Now you are going to define points that build-up the geometry. Lines are then defined between points, surfaces inscribed between lines can then be defined and finally solids can be defined which are inscribed between surfaces. First define points using the syntax:

K, number, x-coordinate, y-coordinate, z-coordinate

Thus, ”K,1,0,0,0” defines a point 1 in the origin (K comes from the word Key point).

When all the points are defined, you define lines between the points by

L, point 1, point 2

Again, ”L,1,2” defines a line between points 1 and 2. A line number is assigned automatically by ANSYS and is given in the order the lines are created. If you wish to keep track of your line numbers, then give them in the order you wish to have them.

You can create surfaces in many different ways in ANSYS, but the most straightforward approach is to create them using the lines that define their boundaries. The syntax is

AL, line 1, line 2, line 3(…etc)

You must use at least three lines to define a surface, but you can use more (almost any number). However, the lines must connect all the way around the surface, at mutual end-points and be given in the right order.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Geometry
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
K,1,b,0,0            ! Defines point 1 – at (x,y,z)=(b,0,0)
K,2,0,0,0
K,3,0,h,0
K,4,b,h,0
K,11,b,0,L           ! L here refers to the variable L, i.e., the beam length
K,12,0,0,L
K,13,0,h,L
K,14,b,h,L

L,1,2                ! Line 1 – between points 1 and 2
L,2,3
```

```
L,3,4
L,11,12
L,12,13
L,13,14
L,1,11
L,2,12
L,3,13
L,4,14

AL,8,5,9,2          ! Area1 – surface inscribed between the lines 8, 5, 9 and 2.
AL,7,1,8,4
AL,9,6,10,3
```

When you have come this far in the geometric modelling it may be time to check that the model is correct. Write an /EOF command after the last AL-command and let ANSYS run the file. Start ANSYS using the command "ansys" in a terminal window (ANSYS uses the terminal window so do not write "ansys &"). When you run ANSYS the program creates a temporary folder named /tmp/ansys-yourusername/ where everything from the run is saved. Move your input file to the /tmp/ansys-yourusername folder. Then just type the *filename* in the command prompt and press ENTER. A plot with some points and lines should appear in the window named *Graphics*. If you choose *PlotCtrls>Pan, Zoom, Rotate*, from the menu yet another window appears. In this one you can zoom and rotate the model to look at it more closely. If you mark *Dynamic Model Mode*¸ you can rotate and zoom in the model using the mouse. Check that the model seems correct and that all points and lines are where you expect them to be. Then remove the /EOF command and continue your work. All the commands performed by ANSYS will be printed in the terminal window and can be used when searching for errors.

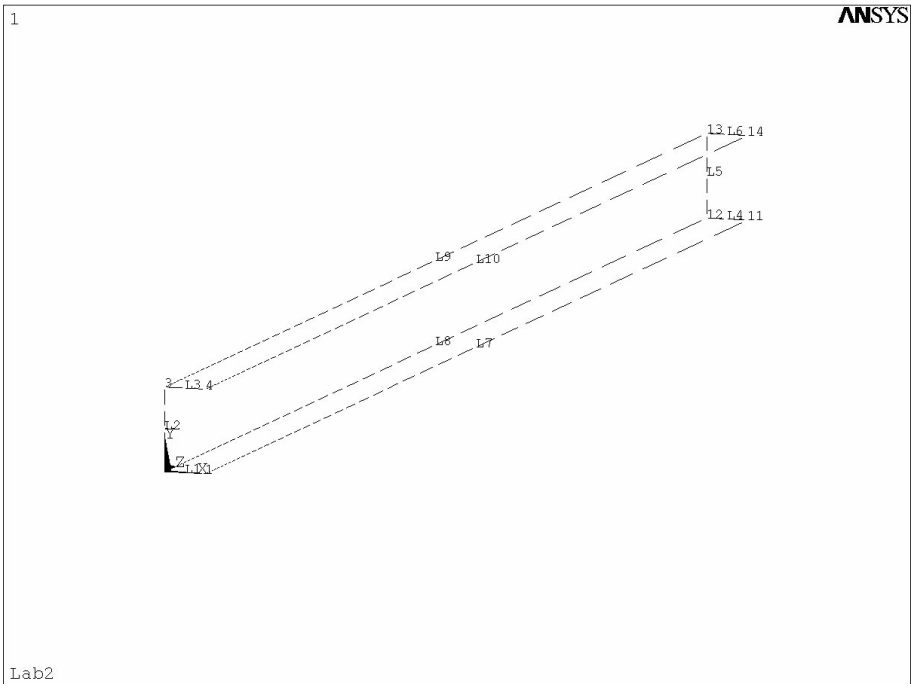


Figure B1: Lines and points in the beam model.

Element mesh

When the elements are generated it is important that the mesh becomes "nice" and that there are no largely deformed elements. The elements should be as near rectangular as possible and with all corners having near 90-degree angles! The side aspect ratio (longest side length/shortest side length) should not normally exceed 4:1 and never exceed 10:1. The best way is to control the meshing by prescribing the number of elements along each line. This is done using the command LESIZE with the syntax:

LESIZE, line number, , , number of elements, concentration factor

In the empty positions above you can give more parameters, but this requires some more descriptions that are not necessary here. They are explained in the ANSYS manual.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Element mesh
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
LESIZE, 1, , , number                ! Element division for line 1
LESIZE, 2, , , 2*number
LESIZE, 3, , , number
LESIZE, 4, , , number
LESIZE, 5, , , 2*number
LESIZE, 6, , , number
LESIZE, 7, , , X*number, Y
LESIZE, 8, , , X*number, Y
LESIZE, 9, , , X*number, Y
LESIZE, 10, , , X*number, Y
```

The *concentration factor* (*X**number) makes the mesh denser towards one end of the line. A factor larger than 1 gives a densification towards the first point in the line and a factor <1 generates the densification towards the second point in the line. For example, line 7 was created with L,1,11 and thus starts at point 1 and ends with point 11. The densification for line 7 is thus towards point 1. Ensure now that all lines (along the beam) are defined in the same direction so that the mesh densification becomes the same. The factor *Y*=5 for lines 7-10 in this example, densifies the mesh towards the boundary (symmetry plane) of the beam. You should test different values of *X* and *Y* to make your mesh look "nice" and adapted to your specific cross section. If you like you can parameterise *X* and *Y* by assigning their values together with your beam dimensions above.

When the element mesh is done for all lines, you call for the predefined element types, materials and real constant groups. The command AMESH divides the surfaces into elements with the active properties. This is done in the following way:

```
TYPE,1          ! Active element type is set to 1 (earlier defined as SHELL93)
REAL,1          ! Activates real constant group 1 (web thickness)
MAT,1           ! Activates material 1
AMESH,1         ! Divides the web-surface into elements (surface 1)
```

REAL,2 ! Activates the real constant group 2 (flange thickness)
AMESH,2,3 ! Divides the flanges surfaces into elements (surfaces 2 and 3)
When the meshing is done, you should check that everything looks as expected by plotting the model using the EPLO command. The model is then ready and you can quit the pre-processor.

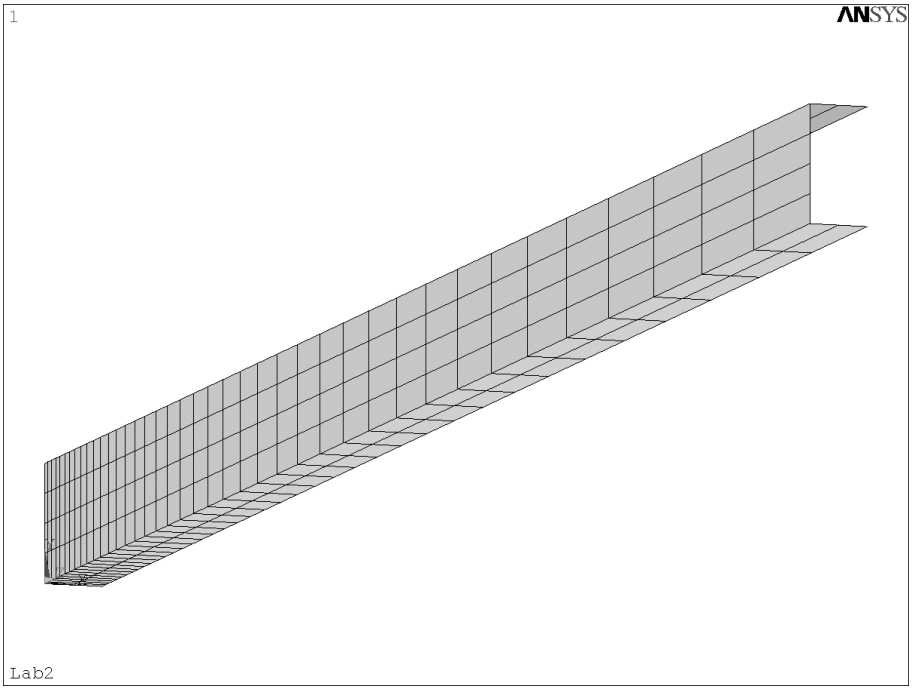


Figure B2: Element mesh plot

EPLO ! Plots the element mesh
FINISH ! Quits the pre-processor

Once again use the /EOF command and run your model. To run the input file go to *file>clear & start new* and then *file>read input from* and then choose the input file name from the list.

Boundary conditions
The second part of ANSYS, the solution part, is commenced with a /SOLUTION command and you have to describe how the model should be analysed. The boundary conditions are best defined directly to nodes. This is done by choosing nodes using the NSEL command and then apply the wanted restrictions on them using the D command. The command NSEL is accompanied by several flags; NSEL, S, LOC, Z, 0, for example is translated as

NSEL	choose nodes (Nodal Selection)
S	new group of nodes - new Set (to select a subset this parameter should be set to R)
LOC	position (LOCation)
Z	z-coordinate
0	0 (zero)

Thus, in there we choose nodes in the plane Z = 0. There are other ways to choose nodes and for more information, check the ANSYS manuals or the on-line help.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Boundary conditions
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/SOLUTION              ! Starts the solution part of ANSYS
NSEL, S, LOC, Z, 0     ! Choose all nodes at the beam edge, all nodes with Z=0
D, ALL, ALL            ! Lock all dof's for the chosen nodes
NSEL, ALL              ! Reactivate all nodes
```

The command D locks nodes (removes all or some dof's) and D, ALL, ALL, means that all selected nodes shall have all dof's locked. Once you have isolated and locked a group of nodes it is important to reselect all nodes again so that all nodes again are active. This is done with NSEL, ALL.

Loads

Boundary conditions and loads are the two things that create the most problems in FE-modelling. If they are applied incorrectly large errors can occur, things like stress concentrations could appear without the user knowing about them. To apply the point loading stipulated in this problem, we shall couple some nodes into a group and then apply a point load. We surely cannot apply a point load anywhere without introducing very large stress concentrations. In order to avoid this we use a so called *multi point constraint*. We shall simply connect the nodes on the web at the free edge of the beam and ensure that they all have the same displacement in the *y*-direction. By doing so, we merely smear out the point load to the entire web and thus avoid the otherwise unwanted stress concentration at a single point.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Load
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
NSEL, S , LOC, Z, L    ! Choose all nodes at the edge of the beam (Z=L)
NSEL, R, LOC, X, 0     ! Choose all nodes which has X=0 from the active node set
CP, NEXT, UY, ALL      ! Couple these nodes so that they have the same displacement in
                       ! the y-direction (UY)
NSEL, R, LOC, Y, h/2   ! Choose the node in the middle of the web (Y=h/2)
F, ALL, FY, S          ! Apply the load S (earlier defined) in the y-direction
NSEL,ALL               ! Reactivate all nodes
```

In this example, you have chosen nodes in three steps using NSEL. First all nodes on the free edge are chosen (those with Z = L). From this set you choose the ones in the web (with X = 0). These nodes are then constrained to have the same displacement in the *y*-direction. Finally you pick , the node in the middle (with Y=h/2) from the current set, and the point load is applied to this node. We you are finished all nodes should be reactivated again. Check that the load is applied correctly. You can do this using the menu *”PlotCtrls>Symbols>”*, where you change the *Surface Load Symbols* to *Pressures* and *Show press* to *arrows*.

Finish the input and start the solution procedure by simply adding the lines

```
SOLVE                   ! Solve the defined problem!!!!!
FINISH                  ! Quit the solution part of ANSYS
```

End with /EOF and run the model again. You can now again check the applied load by listing the total reaction forces. Click on the *General Postproc* in the *ANSYS Main Menu* (to the left on your screen), the choose *List Results>Reaction Solution* and pick either *All items* or just *FY* (reaction force in *y*-direction).

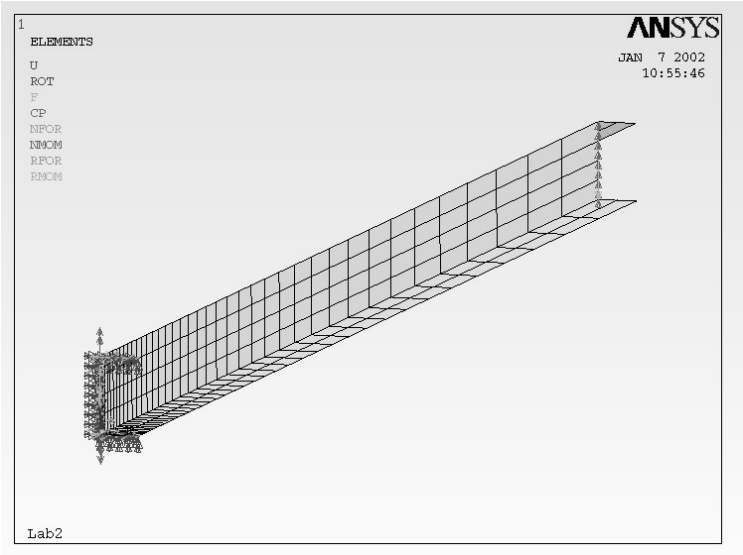


Figure B3: Boundary conditions and loads

Observe that the nodal coupling in the *y*-direction only distributes load acting in the *y*-direction. If the loading had been applied in the *x*-direction, it would still be treated as a point load with an accompanied stress concentration and following poor results.

Post processing

The problem is now solved but you need access to the results to be able to analyse them. This is done in the GUI, but can already be started in the input file. For example you can make a stress plot using the PLNSOL command.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Post processor
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/POST1                  ! Starts the post processor
PLNSOL, S, EQV          ! Plots von Mises stress

/VIEW, 1, -1, -.5, 3    ! Changes view
/ANG, 1                 ! Enables rotation of the plot
/AUTO, 1                ! Scales the plot to fit the window
/REPLOT
```

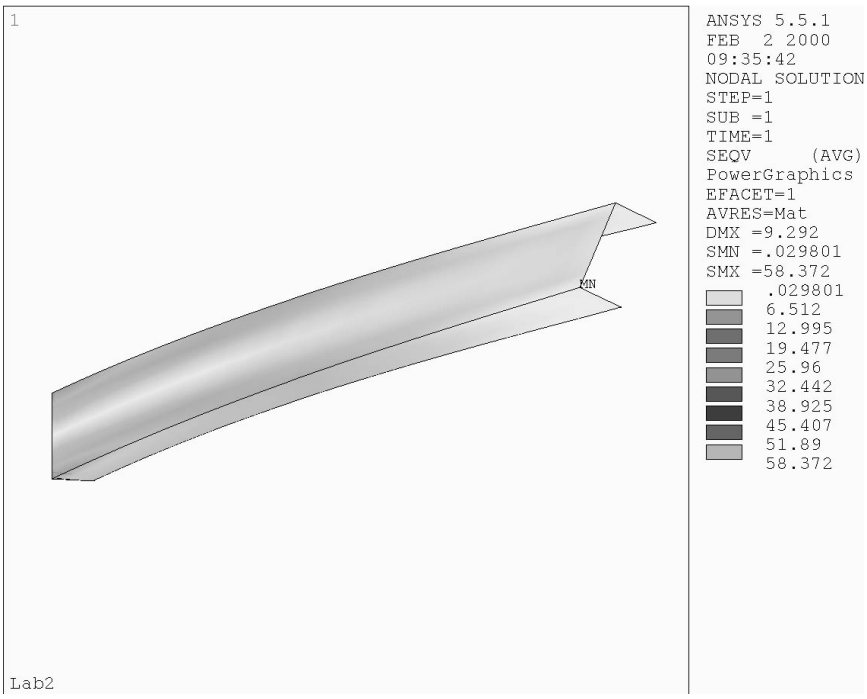


Figure B4: Stress plot

The PLDISP command plots the shape of the deformed structure and by adding the flag "2" (PLDISP, 2), then the contours of the undeformed shape is also added. Try this by typing the command PLDISP, 2 in the *ANSYS Input* menu. This type of plots of the entire structure may be useful to check that the beam bends and twists the way you expect it to. The displacements are not plotted to scale and are usually greatly exaggerated. This does not mean that the displacements are large or that ANSYS uses large deformation analysis.

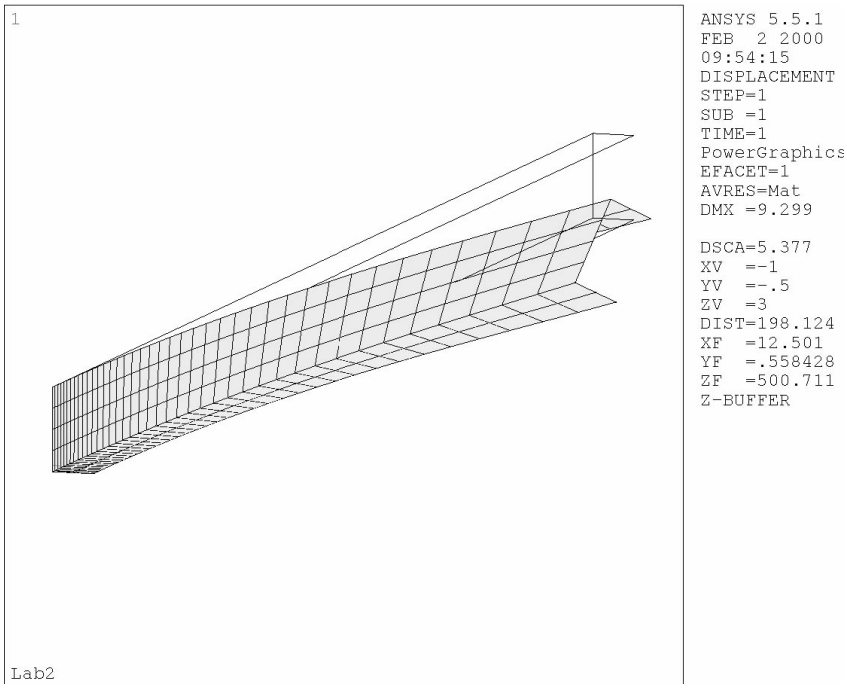


Figure B5: Deformed and undeformed shape of beam

Alternatively, go to *file>general postproc* in the ANSYS GUI. In here you can find the following helpful plot and list commands. There are many more, try them out by clicking around in the menus.

Select nodes or element. Type command in ANSYS command line (top white field)
e.g., NSEL,S,LOC,Z,L
to choose all nodes at the free edge of the beam

To get nodal coordinates
Click on the *list* command on the top row. Choose *nodes* and *Coord*
A list of node coordinates will appear.

Create lists of deformations of selected nodes
Select nodes of interest in ANSYS command menu by using the NSEL command
General Postproc>List Results>nodal solution>UY
to get list of displacements in the *y*-direction

Plot deformed and undeformed shape on the screen
General Postproc>Plot Results>deformed shape

Plot stress components on the screen
General Postproc>Plot Results>Contour Plot>Element Solution
where you can choose the stress component of interest.

Strain energy
Go into the *ANSYS Main Menu*, and choose *General Postproc*, then *Element Table>Define Table>Add>Energy>Strain enrg SENE*, then pick *Sum of Each Item* to get the total strain energy in the model.

Normal and shear stress distributions
In ANSYS you can get stress distributions by selection of nodes and node numbers in the cross-section of interest, choose these using "*Select>Entities*" (ANSYS top row menu). A window will appear where you can choose nodes in different ways, e.g., "*By num/pick*", in which you can write down a list of node numbers, or using "*By Location*" where you instead define a space interval and all the nodes within this space are selected. You can also choose nodes by clicking on them with the mouse, but this may be a little dubious. You can check the coordinates of the chosen nodes by *List>Nodes (coordinates only)*.
Now go to *"List>Results>Nodal Solution"* and choose *"stress"*, with the choice of components (stress components in global coordinates) or principals (principal stress components). A list of all stress components chosen will then be printed on the screen. Now you have to figure out which ones are normal stresses and which are shear stresses! If you instead of a list want to have a graph, you may use "*Path Operations*". You then define a path using *"General Postproc>Path Operations>Define Path"*, and then click on the nodes, which is in the path and direction you wish to use. Then you choose what you wish to have plotted along this path. Do this with *"General Postproc>Path Operations>Map onto Path"*. Now you are ready to draw the graph with *"General Postproc>Path Operations>Plot Path*

item On Graph". When you are done, remember to use *"Select>Everything"*, to reselect everything in the model.
Can you find the shear flow in a cross-section this way? Give this some good thought. Does this work or not, and if not, why? You can only get numbers for the shear stress, but is this only consisting of a shear flow?

Warping
You can get the displacements in the *z*-direction (warping) using *"List>Results>Nodal Solution"* and choose *"DOF solutions, All DOFs"*. Before doing so, choose the nodes of interest using the NSEL command in the ANSYS command menu. Calculate the warping analytically and compare. When you plot or list displacements in the *z*-direction, is this the true warping deformation? Think about the entire problem! What causes displacements in the *z*-direction? Is it only warping?

Hard copies of plots
Once you have made a plot and you wish to make a hard copy, perhaps to include in your report, go to *PlotCtrls* and choose *Hard Copy*. Now you can choose the format (.eps, .tif, .jpg or similar) and save to a file.

Input file in summary

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Cantilever beam model
!   by Anna and Bob
!   2006-09-01
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

FINISH                  ! Finishes the current job
/CLEAR                  ! Clears the memory in ANSYS

/PREP7                  !Entering pre-processor
/TITLE,Lab2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Parameters
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
h=50                    !Web height, mm
b=25                    !Flange width, mm
tw=2.5                  !Web thickness, mm
tf=1.5                  !Flange thickness, mm
X=1                     !Concentration factor (x)
Y=1                     !Concentration factor (y)
L=1000                  !Beam length, mm
S=-100                  !Load, N
number=2                !Number used in meshing

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!   Material parameters and definitions
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Ea=70000                    !Young's modulus for aluminium
nua=0.3                     !Poisson's factor
MP,EX,1,Ea                  !Assigning Ea to EX of material no 1
MP,NUXY,1,nua               !Assigning nua to NUXY of material no 1

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Element type
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ET,1,93                     !Shell93 -shell element
R,1,tw                      !Web thickness
R,2,tf                      !Flange thickness

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Some numbering
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/PNUM,KP,1                  !Keypoint numbering toggle 1=on 2=off
/PNUM,LINE,1                !Line numbering toggle
/PNUM,AREA,1                !Area numbering toggle
/PNUM,NODE,0                !Node numbering toggle
/PNUM,ELEM,0                !Element numbering toggle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Geometry
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
K,1,b,0,0                   !Keypoint number 1 at (x,y,z)=(b,0,0)
K,2,0,0,0
K,3,0,h,0
K,4,b,h,0
K,11,b,0,L                  ! L on this line refers to the defined variable
K,12,0,0,L                  ! ”L” the length of the beam.
K,13,0,h,L
K,14,b,h,L

L,1,2                       !Line between keypoint 1 and 2. L signals here the
L,2,3                       !command ”Line”.
L,3,4
L,11,12
L,12,13
L,13,14
L,1,11
L,2,12
L,3,13
L,4,14
```

```
AL,8,5,9,2                !Area surrounded by line 8, 5, 9 and 2
AL,7,1,8,4
AL,9,6,10,3

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Meshing
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
LESIZE,1,,,number         !Element division on line 1
LESIZE,2,,,2*number
LESIZE,3,,,number

LESIZE,4,,,number
LESIZE,5,,,2*number
LESIZE,6,,,number

LESIZE,7,,,X*number,Y
LESIZE,8,,,X*number,Y
LESIZE,9,,,X*number,Y
LESIZE,10,,,X*number,Y

TYPE,1                    !Selecting element type
REAL,1                    !Selecting web thickness
MAT,1                     !Selecting material

AMESH,1                   !Meshing flanges

REAL,2                    !Selecting flange thickness
AMESH,2,3                 !Meshing web

EPLO                      !Plotting elements
FINISH                    !Finishing pre-processor

/SOLUTION                 !Starting solution mode

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Boundary conditions
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
NSEL,S,LOC,Z,0            !Selecting nodes at beam end
D,ALL,ALL                 !Clamped beam end
NSEL,ALL                  !Reselecting all nodes

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Load
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
NSEL,S,LOC,Z,L
NSEL,R,LOC,X,0
```

```
CP,NEXT,UY,ALL
NSEL,R,LOC,Y,h/2
F,ALL,FY,S
NSEL,ALL

SOLVE                   !Solving the problem
FINISH                  !Finishing solution mode

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Post-processing
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/POST1                  !Entering postprocessor
PLNSOL,S,EQV            !Plotting von Mises stress

/VIEW,1,-1,-0.5,3       !Changes the view point
/ANG,1
/AUTO,1
/REPLOT
```