Exersice 8.1 : Rotating Disc

F7024T - Multiphysics, Simulation and Computation

Brolin, Kask

7 mars 2018

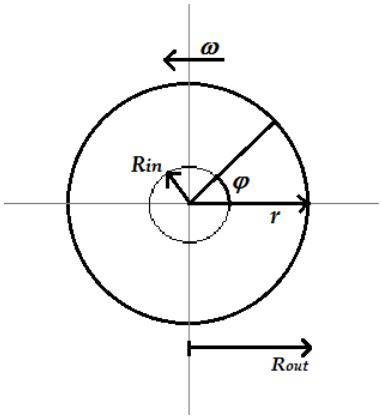
Daniel Brolin
danbro-3@student.ltu.se
Nathalie Kask
natkas-2@student.ltu.se

1 Abstract

This report is the written documentation of the project conducted in the course Multiphysics, Simulation and Computation ($F7024T$) at Luleå University of Technology. The objective of this project is to solve the axisymmetric problem of a rotating disc by finding an numerical solution using a two node element. The result from the numerical solution is then evaluated by comparison with an analytic solution through a convergence study. The comparison is made both for the radial displacement u and the radial- and hoop stress. Further, the error is evaluated in terms of the energy norms as given in equations (1) and (2), for the analytic and numerical solution respectively. The solution is implemented in the software MATLAB for computational and visual support. The result shows that a numerical solution obtained from 100 nodes gives an approximate solution sufficiently close to the analytic solution. The results also shows that by using more nodes in the numerical calculation the solution will have a higher condition number, while no significant difference is given to the resulting accuracy of the numerical solution.

2 Introduction

The objective of this computational exercise is to find an approximate solution to the stress, in r-direction, and the strain, in φ-direction, for a thin disc in rotation. The approximate solution is found by using a two node element. In this exercise the problem is simplified by the assumption that the disc is sufficiently thin to regard stress perpendicular to the radial axis as zero. Furthermore, another factor that is not taken into consideration for this solution is the gravitational force acting on the disc. A visualization of the system in consideration is presented in figure 1. In figure 1 the angular velocity is represented by ω, together with the inner- and outer radius R_{in} and R_{out}, respectively, and finally the angular displacement φ and the radius r.



Figur 1: The system and variables in consideration for solving the problem of a disc in rotation.

The error of the numerical solution is also to be evaluated. This is done in terms of the energy norm, which is obtained from equation (1) for the analytic method and from equation (2) for the numerical method. The error is also evaluated through a convergence study for an overview on how the number of nodes effects the precision of the numerical result.

$$\Pi = \int_{R_{in}}^{R_{out}} \frac{1}{2}\epsilon^T \sigma \cdot r\, dr - \int_{R_{in}}^{R_{out}} u^T f_r \cdot r\, dr \qquad (1)$$

$$\Pi = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} - \mathbf{U}^T\mathbf{F}_e xt \qquad (2)$$

3 Method

This section presents the process, from which, the analytic and numerical solutions are derived. Further, both solutions are implemented in MATLAB and the material that is used for the study of this problem is Aluminum.

3.1 Analytic method

From differential equation (8.1) in the compendium [1] the equilibrium equation for the problem is given by equation (3).

$$L(u) = \frac{d\sigma_r}{dr} + \frac{1}{r}(\sigma_r - \sigma_\phi) + f_r = 0 \qquad (3)$$

Where r is defined as in equation (4) and f_r is the volumetric loading, which in the case of a rotating

disc, is given by equation (5), where ω is the angular velocity $[rad/s]$ of the disc and ρ is the density $[Kg/m^3]$ of the material, of which, the disc consist.

$$r \in [R_{in}, R_{out}] \tag{4}$$

$$f_r = \rho r \omega^2 \tag{5}$$

In order to express equation (3) as Navier's equation, the stress σ and strain ε needs to be eliminated. This is done by using the relations of strain and stress in Hook's law presented in equation (6) and the strain displacement given by equation (7). In equation (6), E is the elastic modulus and ν is Poisson's ratio.

$$\begin{bmatrix} \sigma_r \\ \sigma_\varphi \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu \\ \nu & 1 \end{bmatrix} \begin{bmatrix} \epsilon_r \\ \epsilon_\varphi \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \epsilon_r \\ \epsilon_\varphi \end{bmatrix} = \begin{bmatrix} \frac{du}{dr} \\ \frac{u}{r} \end{bmatrix} \tag{7}$$

By insertion of equation (7) in equation (6) the resulting equation obtained is consisting of two expressions for σ_r and σ_φ. By rewriting (3) with the resulting expressions, Navier's equation can be expressed as in equation (8).

$$L(u) = \frac{d^2u}{dr^2} + \frac{1}{r}\frac{du}{dr} - \frac{u}{r^2} + \frac{1-\nu^2}{\mathbf{E}} f_r = 0 \tag{8}$$

The boundary conditions are applied at the inner, R_{in}, and outer, R_{out}, radius and are given by equations (9) and (10), respectively.

$$B_n(R_{in}) = \sigma_r(R_{in}) = 0 \tag{9}$$

$$B_n(R_{out}) = \sigma_r(R_{out}) = 0 \tag{10}$$

The solution is given on the form, as presented in equations (11) and (12). Where the previously presented boundary conditions gives A and B as in equations (13) and (14), respectively. By insertion of A and B, the resulting stress components for σ_r and σ_φ are presented in equations (15) and (16).

$$\sigma_r = A - \frac{B}{r^2} - \frac{3+\nu}{8}\rho\omega^2 r^2 \tag{11}$$

$$\sigma_\varphi = A + \frac{B}{r^2} - \frac{1+3\nu}{8}\rho\omega^2 r^2 \tag{12}$$

$$A = \frac{3+\nu}{8}\rho\omega^2 \frac{R_{in}^2 R_{out}^2 - R_{in}^4}{R_{in}} \tag{13}$$

$$B = \frac{3+\nu}{8}\rho\omega^2 R_{in}^2 R_{out}^2 \tag{14}$$

$$\sigma_r = \frac{3+\nu}{8}\rho\omega^2 \left(R_{out}^2 - R_{in}^2 - \frac{R_{in}^2 R_{out}^2}{r^2} - r^2 \right) \tag{15}$$

$$\sigma_\varphi = \frac{3+\nu}{8}\rho\omega^2 \left(R_{out}^2 - R_{in}^2 + \frac{R_{in}^2 R_{out}^2}{r^2} \right) - \frac{1-3\nu}{8}\rho\omega^2 r^2 \tag{16}$$

From this the analytic solution is obtained as presented in equation (17).

$$u = \frac{(3+\nu)(1-\nu)}{8\rho\omega^2 E r} \left(R_{in}^2 + R_{out}^2 - \frac{1-\nu}{3+\nu} r^2 + \frac{(1+\nu) R_{in}^2 R_{out}^2}{(1-\nu) r^2} \right) \tag{17}$$

3.1.1 Analytic solution: Energy norm

Continuing on to the *energy norm* of the analytic solution. As mentioned earlier, the potential energy is calculated from equation (1). But first solving is required for the strain components. This can be done from Hook's law in equation (6), where ε_r and ε_φ are acquired as presented in equations (18) and (19).

$$\varepsilon_r = \frac{\sigma_r - \nu\sigma_\varphi}{E(1+\nu)} \tag{18}$$

$$\varepsilon_\varphi = \frac{\sigma_\varphi - \nu\sigma_r}{E(1+\nu)} \tag{19}$$

3.2 Numerical method

For dynamic equilibrium of rigid bodies the principle of virtual power states that: *When a rigid body that is in equilibrium is subject to virtual compatible displacements, the total virtual work of all external forces is zero.* Further, by considering an infinitely small element, some mathematical assumptions can be made. Such that, the sum of internal and external work is zero at equilibrium, giving equation (20).

$$\delta W_{int} = \delta W_{ext} \tag{20}$$

The internal work of the system are the stress and strain components of the material of the body in consideration. The work is obtained by the integration of the volume of the element, as presented in equation (21).

$$\delta W = \int_{V^e} (\delta\varepsilon_r \sigma_r + \delta\epsilon_\varphi \sigma_\varphi) dV \tag{21}$$

The internal work given for study of one element is presented in equation (22). Further, the internal work for a thin disc, with inner radius R_{in} and outer radius R_{out}, is given by equation (23).

$$\delta W_{int} = \int_{V^e} \delta\bar{\varepsilon}^T \bar{\sigma} dV \tag{22}$$

$$\delta W_{int} = \int_{R_{in}}^{R_{out}} \delta\bar{\varepsilon}^T \bar{\sigma} \cdot 2\pi r dr \tag{23}$$

Continuing to the external work, the work for one element, is given by equation (24) and for a thin disc, with inner raduis R_{in} and outer radius R_{out}, the external work is given by equation (25).

$$\delta W_{ext} = \int_{V^e} \delta\bar{u}^T f_r dV \tag{24}$$

$$\delta W_{ext} = \int_{R_{in}}^{R_{out}} \delta\bar{u}^T f_r \cdot 2\pi r dr \tag{25}$$

By insertion of equations (23) and (25) in equation (20), equation (20) can be rewritten to equation (26)

$$\int_{R_{in}}^{R_{out}} \delta\varepsilon^T \cdot 2\pi r dr = \int_{R_{in}}^{R_{out}} \delta\bar{u}^T f_r \cdot 2\pi r dr \tag{26}$$

By change of variables and interpolation of the element displacement, equation (27) is obtained. Further, the strain related to the displacement from equation (7) is as presented in equation (28).

$$u^e(s(x)) = \mathbf{N}(s)\bar{u} = \begin{bmatrix} \frac{1-s}{2} & \frac{1+s}{2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{27}$$

$$\bar{\varepsilon} = \begin{bmatrix} \varepsilon_r \\ \varepsilon_\varphi \end{bmatrix} = \mathbf{B}\bar{u} = \begin{bmatrix} \frac{dN_1}{dr(s)} & \frac{dN_2}{dr(s)} \\ \frac{N_1}{r(s)} & \frac{N_2}{r(s)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{28}$$

By insertion of equations (27) and (28) in equation (26) and the cancellation of 2π, the following expression is obtained.

$$\int_{R_{in}}^{R_{out}} \delta(\mathbf{B}\bar{u})^T \bar{\sigma} r dr = \int_{R_{in}}^{R_{out}} \delta(\mathbf{N}\bar{u})^T f_r r dr \tag{29}$$

Further, by rewriting the previous equation, the following equation is obtained.

$$\int_{R_{in}}^{R_{out}} \mathbf{B}^T \bar{\sigma} r dr = \int_{R_{in}}^{R_{out}} \mathbf{N}^T f_r r dr \tag{30}$$

The connection between the stress and strain is given by Hook's law as presented in equation (31). Where $\mathbf{E}$ is given by the matrix in equation (32)

$$\sigma = \mathbf{E}\varepsilon \tag{31}$$

$$\mathbf{E} = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} \end{bmatrix} \tag{32}$$

By insertion of equation (31) in equation (30), and rewriting the result with equation (28) the expression in equation (33) is obtained.

$$\int_{R_{in}}^{R_{out}} \mathbf{B}^T \mathbf{E} \mathbf{B} r dr \bar{u} = \int_{R_{in}}^{R_{out}} \mathbf{N}^T f_r r dr \tag{33}$$

Furthermore, equation (33) can be rewritten to the expression in equation (34), and thus the stiffness matrix can be obtained, as given in equation (35). Where $\bar{f}_{ext}$ is the load vector, due to different kinds of loading such as possible natural boundary conditions. The load vector is given by equation (36).

$$\mathbf{ku} = \mathbf{f}_{ext} \tag{34}$$

$$\mathbf{k} = \int_{l^e} \mathbf{B}^T \mathbf{E} \mathbf{B} r dr \tag{35}$$

$$\mathbf{f}_{ext} = \int_{l^e} \bar{N}^T f_r r dr \tag{36}$$

Where l^e is the length of the element. By substitution of dr with $\bar{J}ds = \frac{1}{2}ds$, where $\bar{J} = \frac{l^e}{2}$ is the Jacobian of the variable substitution, $\bar{k}$ and f_{ext} can be obtained by the following expressions.

$$\mathbf{k} = \frac{1}{2} \int_{-1}^{1} \bar{B}^T \mathbf{B} \mathbf{E} r(s) ds \tag{37}$$

$$\mathbf{f}_{ext} = \frac{1}{2} \int_{-1}^{1} \bar{N}^T f_r r ds \tag{38}$$

3.2.1 Numeric solution: Energy norm

In order to calculate the *energy norm* of the potential energy for the numerical method, with a varying number of nodes, equation (39) is used.

$$\Pi = \frac{1}{2} \mathbf{U}^T \mathbf{K} \mathbf{U} - \mathbf{U}^T \mathbf{F}_{ext} \tag{39}$$

In this case the element displacement is given by the vector $\mathbf{U}$ as in equation (40). Where the M denotes the number of elements and N the number of nodes, giving the vector $\mathbf{U}$ the length N.

$$\mathbf{U} = \begin{bmatrix} u_1^1 \\ u_1^2 \\ \vdots \\ u_1^N \\ u_M^N \end{bmatrix} \tag{40}$$

Further, $\mathbf{K}$ is the Zero global matrix given by the sparse matrix presented in equation (41).

$$\mathbf{K}=\begin{bmatrix}
k_{11}^1 & k_{12}^1 & 0 & \dots & & & & & \dots & 0 \\
k_{12}^1 & k_{22}^1 & \ddots & \dots & & & & & \dots & 0 \\
0 & \ddots & \ddots & \ddots & & & & & \dots & 0 \\
\vdots & & \ddots & \ddots & k_{N(N-1)}^1 & & & & \dots & 0 \\
0 & \dots & & k_{N(N-1)}^1 & k_{NN}^1+k_{11}^2 & k_{12}^2 & & & \dots & 0 \\
0 & \dots & & & k_{12}^2 & k_{22}^2 & & & & 0 \\
0 & \dots & & & \ddots & \ddots & \ddots & & & 0 \\
0 & \dots & & & & & k_{NN}^{M-1}+k_{11}^M & k_{12}^M & & \vdots \\
0 & \dots & & & & & k_{12}^M & k_{22}^M & \ddots & 0 \\
0 & \dots & & & & & \ddots & & \ddots & \ddots \\
0 & \dots & & & & & & & \ddots & k_{NN}^M
\end{bmatrix} \tag{41}$$

Where

$$\begin{bmatrix} k_{11}^e & k_{12}^e & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & k_{N(N-1)} & k_{NN}^e \end{bmatrix} = \int_e \begin{bmatrix} \frac{dN_1}{dx} & \frac{dN_2}{dx} & \dots & \frac{dN_N}{dx} \end{bmatrix} \begin{bmatrix} \frac{dN_1}{dx} \\ \frac{dN_2}{dx} \\ \vdots \\ \frac{dN_N}{dx} \end{bmatrix} dx \tag{42}$$

3.3 Matlab code

The MatLab code is assembled in five '.m' files; each containing a script, function or multiple functions.

3.3.1 main.m

The main script, see appendix A.2.1, sets the process up by describing the material, choosing the amount of test nodes.
The main function then runs the three functions which gets this data and use it to solve the problem. These functions are "an_sol.m", see appendix A.2.2, used to get the exact numerical soloution; "Num_sol.m", see appendix A.2.3, used to get the numerical solution and "E_sol.m", see appendix A.2.4, used to calculate the potential energy.
When all data has been gathered the system plots all results in one figure, this figure uses the script "tight_subplot.m", see appendix A.2.5, which uses the standard subplot library within matlab and gives the user an easy way to modify the looks of the plot. For simplicity and for a nice square grid only the nomerical potential energy is plotted.

3.3.2 an_sol.m

The exact solution can be found by using the function "an_sol" which sets up a radius vector 'r' with the amount of nodes, 'N', specified in the main script. It then follows the solution written about in subsection 3.1, iterating 'N' times and interpolating between the points.

3.3.3 Num_sol.m

The numerical method used is described in subsection 3.2. Most of the script implements functions found in the compendium[1, section 6,7] to use gauss integration and the Ritz method.
The scripts iterates two calculations. The first loop calculates K and F_an by running an internal function "f_an_comp" for two nodes, where the nodes are the integration points and weights $1, 2$[1, table 7.1] for each set of two adjacent nodes in the radius vector 'r', i.e. for $r(i)$ and $r(i+1)$ where $i = \sum_j^{N-1} j$. Each loop updates all adjacent elements in K with the respective output k from "f_an_comp", and the same with F_{an} and the output f_{an}. This creates the sparse zero global matrix K mentioned in Equation 41.
This gives the return values U, K, F_{an} as $U = K \backslash F_{an}$.
The second loop works almost exactly the same, but initializes variables needed by the second internal function, "sig_comp_num". By using relations $\epsilon = B * u$ and $\sigma = E * \epsilon$ within the two-node loop we can iterate over the entire radius and similarly to the first loop update $R, \sigma_{r,num}, \sigma_{\phi,num}$.

3.3.4 E_sol.m

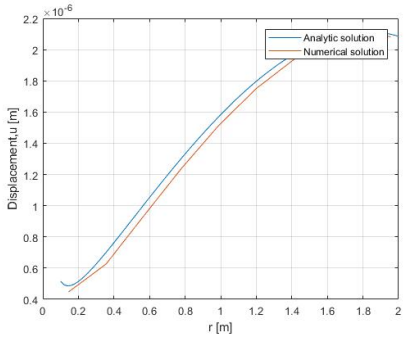
The energy is calculated with less nodes than the rest of the system, this is simply because the scripts takes longer to run. For all "energy nodes", (N_E), the numerical solution script runs with $N = \sum_{i=2}^{N_E}$ and sets all the values to the initiated vector to $E_N = u'_{num} * K * u_{num} - u'_{num} * F_{an}$. This is the numerical potential energy.
The analytical potential energy is gathered by integrating two internal functions calculating potential energy, "E_p_an1" & "E_p_an2".
The first integral uses the common method seen in compendium chapter 8[1, section 8]. Gets A, B from compendium equations $(8.10, 8.11)$; $\sigma_r, \sigma_\phi, \epsilon_r, \epsilon_\phi$ from compendium equations (8.7 through 8.9) and the rest can be derived from compendium equation (8.15).
The second energy solution integral uses the same solution as the function calculating the analytical solution, see subsection 3.1.
Both the numerical energy and the analytical energy is returned.

4 Results

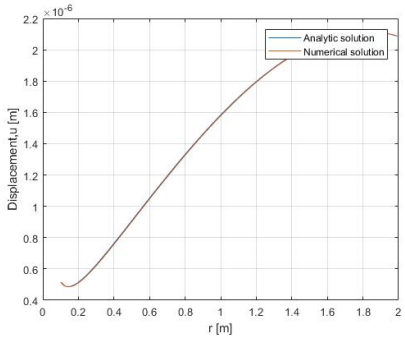
The solutions are tested for 10, 100, 1000 nodes in the numerical calculations. In these following sections the resulting displacement u, and the stress components σ_r and σ_φ and, finally the convergence study conducted over different number of nodes will be presented in table 1.

4.1 Displacement u

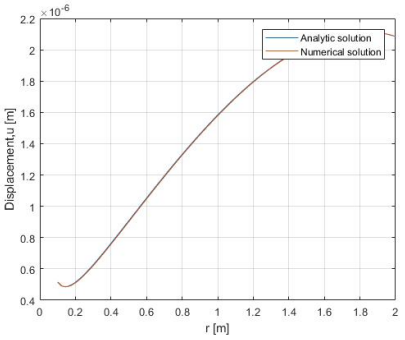
In this section the calculated displacement, u, of the analytic- and the numerical solution is displayed for evaluation. Figures 2, 3 and 4 present the result for numerical calculation with 10, 100 and 1000 nodes, respectively.



Figur 2: This figure shows the comparison of the resulting displacement, u, for when the numerical calculations are made for 10 *nodes.*



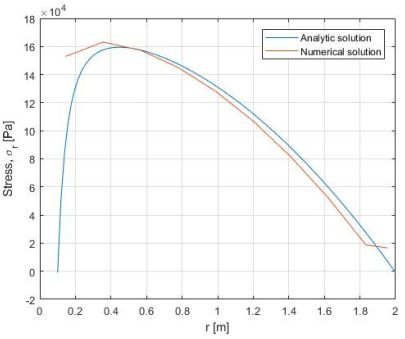
Figur 3: This figure shows the comparison of the resulting displacement, u, for when the numerical calculations are made for 100 *nodes.*



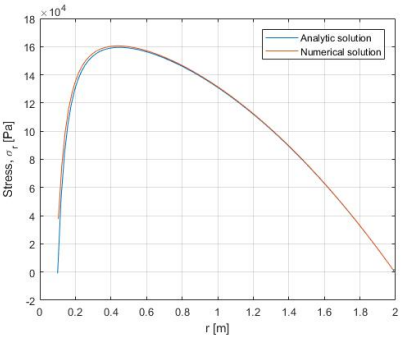
Figur 4: This figure shows the comparison of the resulting displacement, u, for when the numerical calculations are made for 1000 *nodes.*

4.2 Stress σ_r

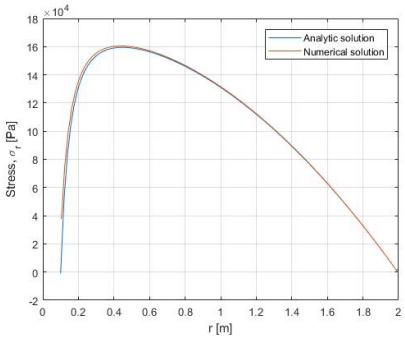
In this section the calculated stress σ_φ of the analytic- and the numerical solution are displayed for evaluation. Figures 5, 6 and 7 present the result for numerical calculation with 10, 100 and 1000 nodes, respectively.



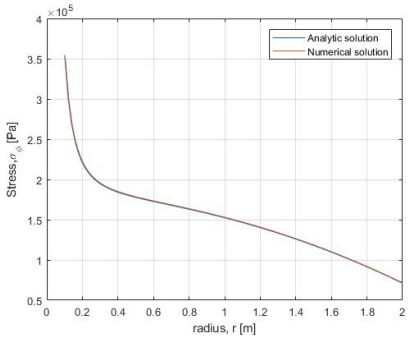
Figur 5: This figure shows the comparison of the resulting stress, σ_r, for when the numerical calculations are made for 10 nodes.



Figur 6: This figure shows the comparison of the resulting stress, σ_r, for when the numerical calculations are made for 100 nodes.



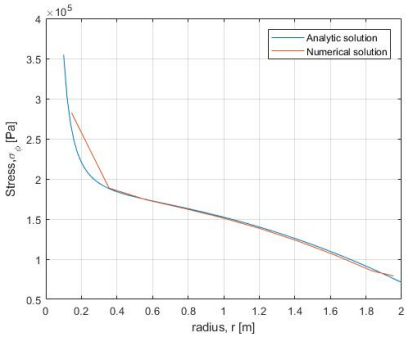
Figur 7: This figure shows the comparison of the resulting stress, σ_r, for when the numerical calculations are made for 1000 nodes.



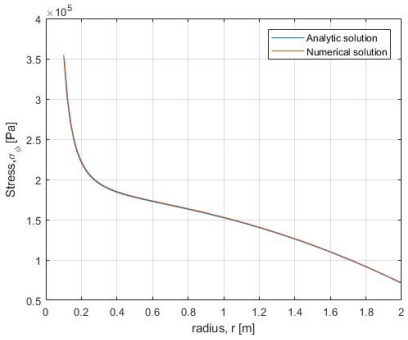
Figur 10: This figure shows the comparison of the resulting stress, σ_φ, for when the numerical calculations are made for 1000 nodes.

4.3 Stress σ_φ

In this section the calculated stress σ_φ of the analytic- and the numerical solution are displayed for evaluation. Figures 8, 9 and 9 present the result for numerical calculation with 10, 100 and 1000 nodes, respectively.



Figur 8: This figure shows the comparison of the resulting stress, σ_r, for when the numerical calculations are made for 10 nodes.



Figur 9: This figure shows the comparison of the resulting stress, σ_r, for when the numerical calculations are made for 100 nodes.

4.4 Convergence study

In Table (1) the result of the convergence study is presented.

Convergence study:		
Nodes	Error	Condition number
10	7.8510e-09	140.7487
20	1.7051e-09	637.0929
30	6.3367e-10	1.4898e+03
40	3.0623e-10	2.6990e+03
50	1.7292e-10	4.2643e+03
60	1.0816e-10	6.1862e+03
70	7.2697e-11	8.4639e+03
80	5.1532e-11	1.1098e+04
90	3.8052e-11	1.4089e+04
100	2.9020e-11	1.7436e+04
1000	8.4829e-14	1.7775e+06

Tabell 1: This table display the error between the analytic- and numerical solution together with the condition number, for calculation with 10 to 1000 nodes.

5 Conclusion

As can be seen in figure 2 to 10, the numerical method gives an approximation visibly close to the analytic solution with 100 nodes. This is can also be seen in Table 1. Also, from Table 1 it can be seen that there is an increase in the condition number, when increasing the number of nodes.

6 Discussion

As mentioned in the previous section, a higher number of nodes used in the numerical calculations gives

an increase in the conditioning number. Further, as seen in figures 3 and 4, 6 and 7, and finally 9 and 10, the additional resolution from 100 to 1000 nodes does not give any significant difference for the result. Since the conditioning number indicates the ill-condition of the solution and the sensitivity to perturbations, the use of 1000 nodes in the calculation is not advised. The two node element solution was difficult to implement in MATLAB and immense efforts was invested in order to make it work. The resulting solution, however, provide high precision. The only drawback experienced was the memory available on the computer used for the calculations. There are several points where the code could be improved to run more smoothly. A good example could be to combine the internal functions in the numerical solution, since these are really similar and using two loops is rather unnecessary. Another improvement could be a more compact way to solve for the energy, the code is rather sparse and could need a re-write.

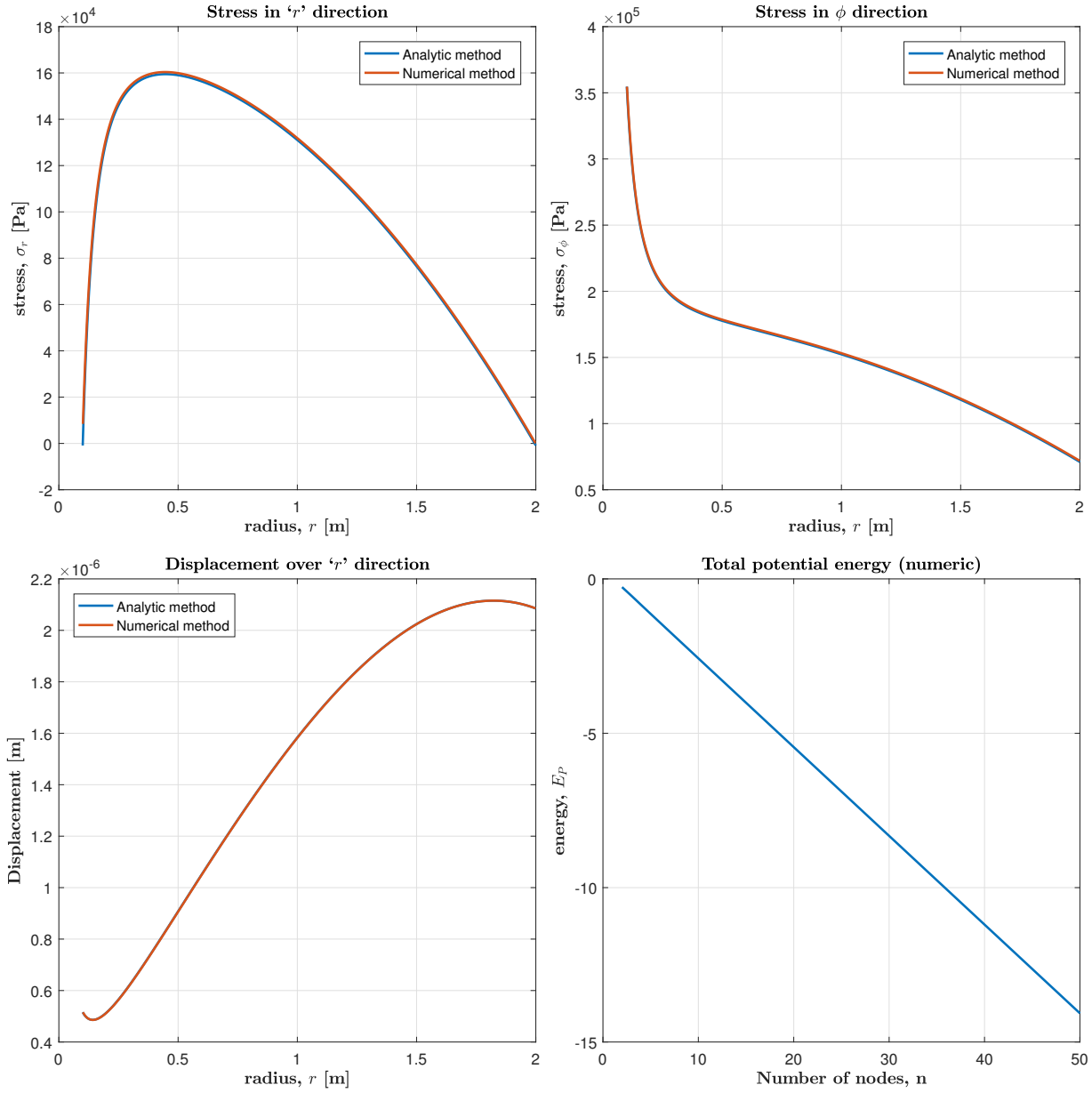
Referenser

[1] Lars-Erik Lindgren. From weighted residual methods to finite element methods. Technical report, Department of Engineering Science and Mathematics, `https://www.ltu.se/cms_fs/1.47275!/mwr_galerkin_fem.pdf`, 2009.

A APPENDIX

A.1 Figures

A.1.1 Result by running main program



Figur 11: Resulting output at $N = 500$ and N_{Energy} for material "Aluminium"

A.2 MATLAB functions and scripts

A.2.1 "main.m": Main script for settings, running and plotting

```
%% Setup
% Clear old workspace
close all
clear, clc

Omega = 2*pi;    % rad/s
Ri = 0.1;        % m
Ry = 2.0;        % m
N = 500;         % nodes for analytical and numerical solution
N_E_pot = 50;    % nodes for potential energy

%% Material values: Platinum
E = 168E9;       % 015376 %[Pa] Elasticity, Youngs module
rho = 21450;     % [kg/m^3] Density
pois = 0.39; % Poissons ratio

%% Calculations
% Find the analytical solution, numerical solution and potential energy
[r_an, u_an, sig_r_an, sig_phi_an]           = an_sol(E, rho, Ri, Ry, Omega,pois, N);
[r_num, u_num, sig_r_num, sig_phi_num, K, ~]= Num_sol(E, rho, Ri, Ry, Omega,pois, N);
[E_pot_num, ~]                               = E_sol(Ri, Ry, E, pois, Omega, rho, N_E_pot);

%% Plotting
[ha, ~] = tight_subplot(2,2,[.08 .04],[.05 .03],[.03 .02]);
%subplot(2,2,1)
axes(ha(1))
plot(r_an, sig_r_an, r_num, sig_r_num, 'Linewidth', 1.5);
xlabel('\bf{radius, $r$ [m]}', 'Interpreter','latex');
ylabel('\bf{stress, $\sigma_r$ [Pa]}', 'Interpreter','latex');
legend('Analytic method', 'Numerical method');
title('\bf{Stress in `$r$'' direction}', 'Interpreter','latex'); grid on
%subplot(2,2,2)
axes(ha(2))
plot(r_an, sig_phi_an, r_num, sig_phi_num, 'Linewidth', 1.5);
xlabel('\bf{radius, $r$ [m]}', 'Interpreter','latex');
ylabel('\bf{stress, $\sigma_{\phi}$ [Pa]}', 'Interpreter','latex');
legend('Analytic method', 'Numerical method');
title('\bf{Stress in $\phi$ direction}', 'Interpreter','latex'); grid on
%subplot(2,2,3)
axes(ha(3))
plot(r_an, u_an, r_num, u_num, 'Linewidth', 1.5);
xlabel('\bf{radius, $r$ [m]}', 'Interpreter','latex');
ylabel('\bf{Displacement [m]}', 'Interpreter','latex');
legend('Analytic method', 'Numerical method', 'location', 'NorthWest');
title('\bf{Displacement over `$r$'' direction}', 'Interpreter','latex'); grid on
%subplot(2,2,4)
axes(ha(4))
plot(2:N_E_pot, E_pot_num, 'Linewidth', 1.5);
xlabel('\bf{Number of nodes, n}', 'Interpreter','latex');
ylabel('\bf{energy, $E_P$}', 'Interpreter','latex');
title('\bf{Total potential energy (numeric)}', 'Interpreter','latex'); grid on
```

A.2.2 "an_sol.m": Function solving system analytically

```
%% Analytical solution of disc problem
function [r, u, sig_r, sig_phi] = an_sol(E, rho, Ri, Ry, Omega, pois, N)
% Analytical solution of disc problem
    r = linspace(Ri, Ry, N);
    u = zeros(1, N);
    sig_r = zeros(1, N);
    sig_phi = zeros(1, N);
    A = ((3+pois)/8)*rho*(Omega^2)*(Ry^2 - Ri^2);
    B = ((3+pois)/8)*rho*(Omega^2)*(Ri^2)*(Ry^2);
    for i = 1:N
        u(i) = ((3+pois)/8)*(1-pois)*((rho*Omega^2)/E)*r(i)*(Ri^2 + Ry^2-((1+pois)/(3+pois))
    *r(i)^2+((1+pois)/(1-pois))*((Ri^2 * Ry^2)/r(i)^2));
        sig_r(i) = A-B/(r(i)^2)-((3+pois)/8)*rho*Omega^2*r(i)^2;
        sig_phi(i) = A + B/(r(i)^2)-((1+3*pois)/8)*rho*Omega^2*r(i)^2;
    end
end
```

A.2.3 "Num_sol.m": Function solving system numerically

```

%% Numerical solution of disc problem
function [R, U, sig_r_num, sig_phi_num, K, F_an] = Num_sol(E, rho, Ri, Ry, Omega, pois, N)
% Numerical solution of disc problem
    %% setup
    K = zeros(N);
    F_an = zeros(N,1);
    r = linspace(Ri, Ry, N);
    sig_r_num = zeros(N,1);
    sig_phi_num = zeros(N,1);
    R = zeros(1,N);
    %% formulas from compendium
    s = [-1/sqrt(3), 1/sqrt(3)];                          % From table (7.1)
    w = [1, 1];                                           % From table (7.1)
    Emod = [E/(1-pois^2), E*pois/(1-pois^2);
            E*pois/(1-pois^2), E/(1-pois^2)];             % From equation (8.6)

    %% calc
    for i = 1:N-1
        k = zeros(1,2); f_an = zeros(2,1);
        B = [-1/(r(i+1)-r(i)), 1/(r(i+1)-r(i)); 0, 0];  % From equation (6.57)
        for id = 1:2 % node id
            [k, f_an] = f_an_comp(s(id), r(i), r(i+1), B, Emod, w(id), k, f_an, Omega, rho);
        end
        K(i,i) = K(i,i)+k(1,1);
        K(i,i+1) = K(i,i+1)+k(1,2);
        K(i+1,i) = K(i+1,i)+k(2,1);
        K(i+1,i+1) = K(i+1,i+1)+k(2,2);

        F_an(i) = F_an(i)+f_an(1);
        F_an(i+1) = F_an(i+1)+f_an(2);
    end

    U = K\F_an;

    for i = 1:N-1
        r_sig = zeros(1,2); sig_r = zeros(1,2); sig_phi = zeros(1,2);
        B = [-1/(r(i+1)-r(i)), 1/(r(i+1)-r(i)); 0, 0];  % From equation (6.57)
        u = [U(i); U(i+1)];
```

```
        for id = 1:2 % node id
            [sig_r(id), sig_phi(id), r_sig(id), B] = sig_comp_num(s(id), r(i), r(i+1), B,
    Emod, u);
        end

        R(i) = r_sig(1);
        R(i+1) = r_sig(2);
        sig_r_num(i) = sig_r(1);
        sig_r_num(i+1) = sig_r(2);
        sig_phi_num(i) = sig_phi(1);
        sig_phi_num(i+1) = sig_phi(2);
    end
end

%% Internal functions
%Computating k and f_an stuff
function [k, f_an] = f_an_comp(s, r1, r2, B, E, w, k, f_an, Omega, rho)
    N = [(1-s)/2 , (1+s)/2];
    r = r1*N(1) + r2*N(2);
    B(2,:) = [N(1)/r ; N(2)/r];
    k = k + B'*E*B*r*w;
    f_an = f_an + N'*rho*(Omega^2)*(r^2)*w;
end

%Computating sigma stuff
function [sig_r, sig_phi, r, B] = sig_comp_num(s, r1, r2, B, E, u)
    N = [(1-s)/2, (1+s)/2];
    r = N(1)*r1+N(2)*r2;
    B(2,:) = [N(1)/r , N(2)/r];
    epsilon = B*u;
    sig = E*epsilon;
    sig_r = sig(1);
    sig_phi = sig(2);
end
```

A.2.4 "En_sol.m": Function solving system in regards to potential energy

```
%% Calculating the potential energy
function [E_pot_num, E_pot_an] = E_sol(Ri, Ry, E, pois, Omega, rho, N_E_pot)
    E_pot_num = zeros(1,N_E_pot-1);

    for N = 2:N_E_pot
        [~, u_num, ~, ~, K, f_an] = Num_sol(E, rho, Ri, Ry, Omega, pois, N);
        E_pot_num(N-1) = 0.5.*u_num'*K*u_num - u_num'*f_an;
    end

    E_pot_an = integral(@(r) E_p_an1(r, pois, E, rho, Omega, Ri, Ry),Ri, Ry)...
               -integral(@(r) E_p_an2(r, rho, Omega, pois, E, Ri, Ry), Ri, Ry);
end

%% Internal functions
% Potential energy analytical 1
function E_p1 = E_p_an1(r, pois, E, rho, Omega, Ri, Ry)
    A = ((3+pois)/8)*rho*(Omega^2)*(Ry^2 - Ri^2);
    B = ((3+pois)/8)*rho*(Omega^2)*(Ri^2)*(Ry^2);
    sig_r = A - B./(r.^2) - ((3+pois)/8)*rho*Omega^2*r.^2;
    sig_phi = A + B./(r.^2) - ((1+3*pois)/8)*rho*Omega^2*r.^2;
    eps_r = (sig_r - pois*sig_phi)/(E*(1+pois));
```

```
    eps_phi = (sig_phi - pois*sig_r)/(E*(1+pois));
    E_p1 = (1/2) * (eps_r .* sig_r + eps_phi .* sig_phi) .* r;
end

% Potential energy analytical 2
function E_p2 = E_p_an2(r, rho, Omega, pois, E, Ri, Ry)
    u = ((3+pois)/8)*(1-pois)*((rho*Omega^2)/E).*r.* ...
        (Ri^2+Ry^2-((1+pois)/(3+pois)).*r.^2+((1+pois)/(1-pois))*((Ri^2*Ry^2)./r.^2));
    E_p2 = u .* rho * Omega^2 .* r.^2;
end
```

A.2.5 "tight_subplot.m": Function used to plot system

```
function [ha, pos] = tight_subplot(Nh, Nw, gap, marg_h, marg_w)
% tight_subplot creates "subplot" axes with adjustable gaps and margins
%
% [ha, pos] = tight_subplot(Nh, Nw, gap, marg_h, marg_w)
%
%   in:  Nh      number of axes in hight (vertical direction)
%        Nw      number of axes in width (horizontaldirection)
%        gap     gaps between the axes in normalized units (0...1)
%                   or [gap_h gap_w] for different gaps in height and width
%        marg_h  margins in height in normalized units (0...1)
%                   or [lower upper] for different lower and upper margins
%        marg_w  margins in width in normalized units (0...1)
%                   or [left right] for different left and right margins
%
%  out:  ha     array of handles of the axes objects
%                   starting from upper left corner, going row-wise as in
%                   subplot
%        pos    positions of the axes objects
%
%  Example: ha = tight_subplot(3,2,[.01 .03],[.1 .01],[.01 .01])
%           for ii = 1:6; axes(ha(ii)); plot(randn(10,ii)); end
%           set(ha(1:4),'XTickLabel',''); set(ha,'YTickLabel','')

% Pekka Kumpulainen 21.5.2012   @tut.fi
% Tampere University of Technology / Automation Science and Engineering

if nargin<3; gap = .02; end
if nargin<4 || isempty(marg_h); marg_h = .05; end
if nargin<5; marg_w = .05; end

if numel(gap)==1;
    gap = [gap gap];
end
if numel(marg_w)==1;
    marg_w = [marg_w marg_w];
end
if numel(marg_h)==1;
    marg_h = [marg_h marg_h];
end

axh = (1-sum(marg_h)-(Nh-1)*gap(1))/Nh;
axw = (1-sum(marg_w)-(Nw-1)*gap(2))/Nw;

py = 1-marg_h(2)-axh;

```

```
% ha = zeros(Nh*Nw,1);
ii = 0;
for ih = 1:Nh
    px = marg_w(1);

    for ix = 1:Nw
        ii = ii+1;
        ha(ii) = axes('Units','normalized', ...
            'Position',[px py axw axh], ...
            'XTickLabel','', ...
            'YTickLabel','');
        px = px+axw+gap(2);
    end
    py = py-axh-gap(1);
end
if nargout > 1
    pos = get(ha,'Position');
end
ha = ha(:);
```