Project 2

Numerical Methods for Differential Equations

Isabelle Frodé & Sara Enander
Date: December 3 2019

1 Two-point Boundary Value Problems

1.1 Constructing a Two-point Boundary Value Solver

The first part of the project aimed to study a two-point boundary value problem by creating a solver in MATLAB that uses finite-differences method. The first task was to create a solver, solving problems such as the one stated in Equation 1. [1]

$$\begin{aligned} y'' &= f(x, y) \\ y(0) = \alpha,\ y(L) &= \beta \end{aligned} \tag{1}$$

The finite-difference method uses finite difference equations to approximate the derivatives. To be able to do so, an equidistant grid $\Delta x = L/(N+1)$ was introduced on the interval [0, L], whereas N is the total number of unknown vector components to the solution y = {y_1, y_2,..., y_N}. The FDM discretization is then obtained by Equation 2. [1]

$$\begin{aligned} \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} &= f(x_i, y_i) \\ y_0 = \alpha, ,\ y_{N+1} &= \beta \end{aligned} \tag{2}$$

When studying this type of linear problems f(x_i,y_i) is equivalent to f(x_i). This could be expressed as F(y)=0 and the linear problem, presented in Equation 3.

$$\begin{aligned} F_1(y) &= \frac{\alpha - 2y_1 + y_2}{\Delta x^2} - f(x_1, y_1) \\ F_i(y) &= \frac{y_{i-1} - 2y_i + y_{i+1}}{\Delta x^2} - f(x_i, y_i) \\ F_N(y) &= \frac{y_{N-1} - 2y_N + \beta}{\Delta x^2} - f(x_N, y_N) \end{aligned} \tag{3}$$

The linear problem could be written as matrices and Equation 3 could then be used to create the the solver. The constructed two-point boundary value problem solver is presented below.

```
    function y = twoBVP(fvec, alpha, beta, L, N)
h = L./(N+1);
sub = ones(1, N-1);
main = -2.*ones(1, N);
sup = ones(1, N-1);
A = 1./(h.^2).*(diag(sub,-1) + diag(main,0) + diag(sup,1));
F(1,1) = -alpha./(h.^2)+fvec(1);
for i=2:N-1
    F(i,1) = fvec(i);
end
```

```
F(N, 1) = -beta./(h.^2)+ fvec(N);
y(1) = alpha;
b=linsolve(A, F).';
y(2: N+1) = b;
y(N+2) = beta;
end
```

To test the solver, an approximation of the solutions y = y_1, y_2,..., y_N was plotted for a test equation $f_{test} = cos(x)$. An approximation of the solution (for number of steps N = 10) and the exact solution to the test equation was plotted and the result is shown in Figure 1.

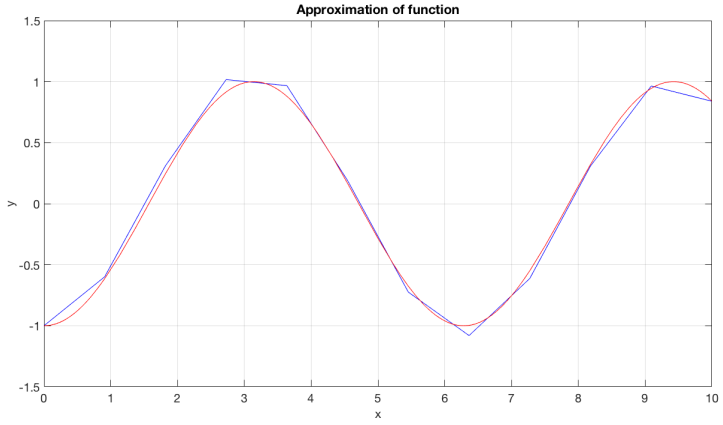


Figure 1: Approximation of solutions to the function y"=cos(x), N=10

To formally verify that the solver works properly, the error was plotted in a loglog plot as a function of stepsize. This was constructed in MATLAB by creating a separate error function. The scrips are found in Appendix. The plot itself is presented in Figure 2. This shows that the approximation was better for smaller stepsizes. By further calculating the k-value of the curve (the slope of the curve) in the lolog plot it was confirmed that the error is indeed O(h^2) and the second order convergence is proven accordingly. The plot shows some stability problems for h less than approximately 10^{-2}, and this was not included in the calculation.

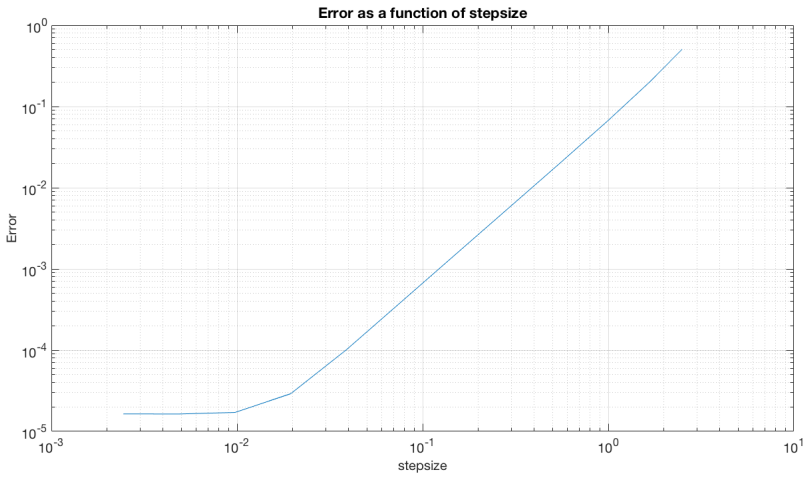


Figure 2: Error as a function of stepsize

1.2 The Beam Equation

In this chapter, the deflection of a beam was studied. The length of the beam $L = 10$ and the load $q(x) = 50$ kN/m was given. [1] The deflection of the beam could be calculated by solving the beam equation, presented in Equation 4. [1]

$$\begin{aligned} M'' &= q(x) \\ u'' &= \frac{M(x)}{EI} \end{aligned} \tag{4}$$

The boundary conditions for the problem is $u(0) = u(L) = 0$, since the beam is supported at both ends. The elasticity module for the beam is $E = 1.9 \cdot 10^{11}$ kN/m and the cross section moment $I(x)$ varies along the beam according to Equation 5. [1]

$$I(x) = 10^{-3} \cdot \left(3 - 2\left(cos\frac{\pi x}{L}\right)^{12}\right) \tag{5}$$

The deflection of the beam was computed with the two-point boundary value problem solver created in previous task. The result is plotted in Figure 3. The plot shows that the deflection is symmetrical, having a maximum deflection at the midpoint. The beam is supported at both ends, the load is constant over the beam and the cross section moment is symmetrical over the beam, hence the deflection must be symmetrical with the maximal deflection at the midpoint. The approximated values of the deflection are aligned with the theory.

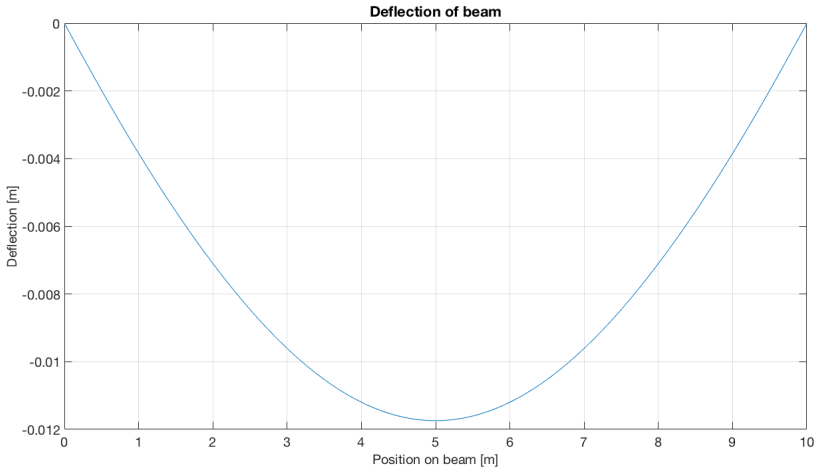


Figure 3: Deflection of beam

The deflection at the beam's midpoint was then calculated to 11.741964 mm, when using a number of $N = 99$ interior grid points. This is considered to be realistic, due to the fact that steel is a strong material, in this case carrying quite a heavy load. Since the beam is 10 m long, it is expected with some deflection.

2 Sturm-Liouville Eigenvalue Problems

2.1 Constructing a Sturm-Liouville Solver

The Sturm-Liouville problem is very diverse and has applications in many areas.[1] In this chapter the Sturm-Liouville problem, shown in equation 6, was studied.

$$\begin{gathered} \frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) - q(x)y = \lambda y \\ y(a) = 0; \;\; y(b) = 0 \end{gathered} \tag{6}$$

For the first task a Sturm-Liouville solver was constructed for a simple case where p(x)=1 and q(x)=0, shown in Equation 6. [1]

$$\begin{aligned} u'' &= \lambda u \\ u(0) = u(1) &= 0 \end{aligned} \tag{7}$$

First, the solution was calculated analytically. The eigenfunctions were calculated to sin(kπx) and eigenvalues to -kπ^2, where k = 1, 2,.. . Equation 7 then was discretized to obtain a matrix eigenvalue problem:

$$T_{\Delta x} y = \lambda_{\Delta x} y \tag{8}$$

This problem has real eigenvalues $\lambda_{\Delta x} = \lambda + O(\Delta x^2)$ because of symmetry. The solver was created by modifying the two-point boundary value solver from previous task. One modification was that this solver returns a vector of eigenvalues and a matrix of eigenfunctions. The solver created is presented below.

```
function [eigvalv, eigfuncv] = SLsolver(N, L)
% Sturm-Liouville solver
h=L/(N+1);
sub=ones(1,N-1);
main=-2.*ones(1,N);
sup=ones(1,N-1);

T = 1./(h.^2).*(diag(sub,-1) + diag(main,0) + diag(sup,1));

[eigfunc,V]=eig(T,'vector');
eigfuncv(1,:)=0;
eigfuncv(2:N+1, 1:N)=eigfunc;
eigfuncv(N+2,:)=0;
eigval=V.';
eigvalv(1)=0;
eigvalv(2:N+1)=eigval;
eigvalv(N+2)=0;
end
```

To verify that the solver functions properly the error for the approximation of the eigenvalues was plotted in a loglog diagram. By calculating the slope of the curve, to a value of -2, it was confirmed that the error is indeed $O(\Delta x^2)$ and the second order convergence was proved. By computing and comparing the approximated eigenvalues it was discovered that the "eig" function in MATLAB put the lowest eigenvalue last. The error for the first three eigenvalues was plotted as a function of N, the number of interior gridpoints, presented in figure 4.

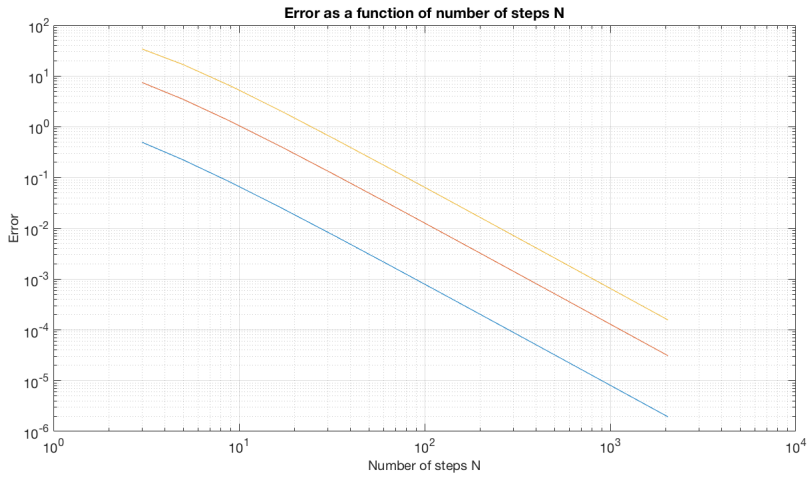


Figure 4: Error of the three smallest eigenvalues

The first three eigenvalues corresponding to Δx= $2{\cdot}10^{-3}$ was then computed using the solver above. The approximated eigenvalues are $\lambda_1 = 9.8695719$, $\lambda_2 = 39.477898$ and $\lambda_3 = 88.823810$.

The eigenmodes was plotted in a graph, shown in Figure 5, where the first eigenfunction corresponds to the function with no node. The second eigenfunction have one node and the third two, in accordance to the theory. The first two eigenfunctions are negative because of how MATLAB's function "eig" works, but it does not affect the result for the purpose of this task.

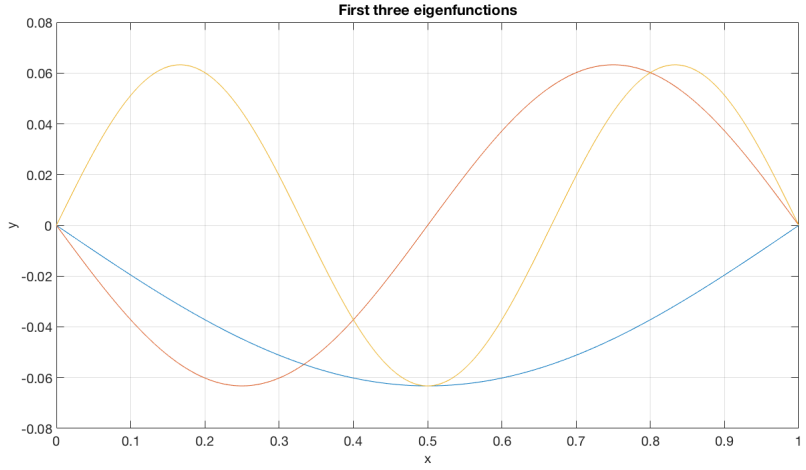


Figure 5: Plot of the first three eigenfunctions

2.2 The Schrödinger Equation

In this chapter a particle in an ∞-potential well is studied using the Schrödinger Equation, which in this case could be reduced to a Sturm-Liouville problem, $\psi'' - V(x)\psi = -E\psi$, since the boundary conditions for such a well is $\psi(0) = \psi(1) = 0$.[1] The Sturm-Liouville solver was modified accordingly. The constructed Schördinger Equation solver is presented below.

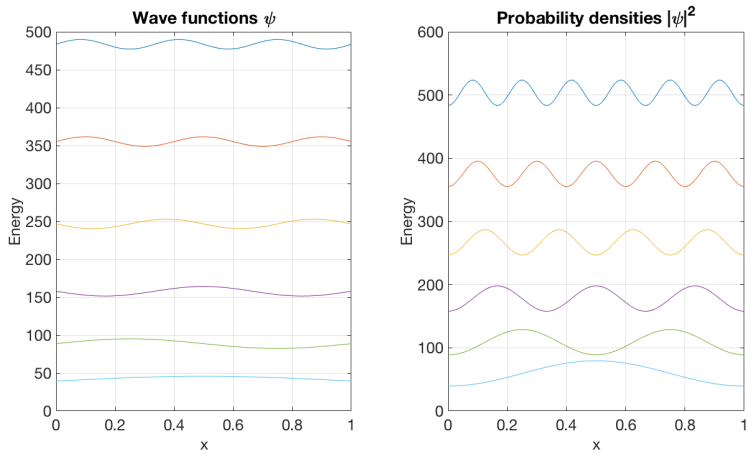
```
function [eigvalv, eigfuncv] = SEsolver(N, L,V)
% Schrödinger equation solver

h=L/(N+1);
sub= 1./(h.^2).*ones(1,N-1);
main= 1./(h.^2).*(-2.*ones(1,N))-V;
sup= 1./(h.^2).*ones(1,N-1);

T =(diag(sub,-1) + diag(main,0) + diag(sup,1));

[eigfunc,D]=eig(T,'vector');
eigfuncv(1,:)=0;
eigfuncv(2:N+1, 1:N)=-eigfunc;
eigfuncv(N+2,:)=0;
eigval=D.';
eigvalv(1)=0;
eigvalv(2:N+1)=-eigval;
eigvalv(N+2)=0;
end
```

The S.E solver was tested by plotting the wave functions ψ and probability densities $|\psi|^2$ for the first six eigenvalues, corresponding to the first six energy levels for a quantum particle trapped in a one-dimensional well with the potential V(x) = 0 inside the interval [0,1] and V(x) = ∞ outside the interval. The result is shown in Figure 6 (a) and Figure 6 (b). The plots were then compared with the expected wave functions and probability functions for the first six energy levels, stated by Gunnar Ohlén in Kvantvärldens Fenomen.[2] The plots were made easier to interpret by normalizing the functions with a constant N = 100 and "lifting" the wave functions and probability densities by adding the value of the energy level obtained by the eigenvalues, hence one could see that the approximations corresponds well with the theory.

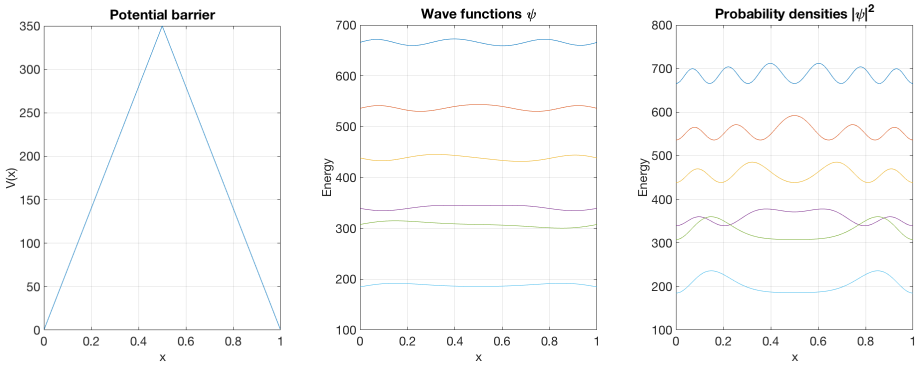


(a) First six wave functions for the lowest energy levels

(b) Probability densities for the lowest energy levels

Figure 6: Wave functions and probability densities for the first six energy levels. The well has its walls at x = 0 and x = 1

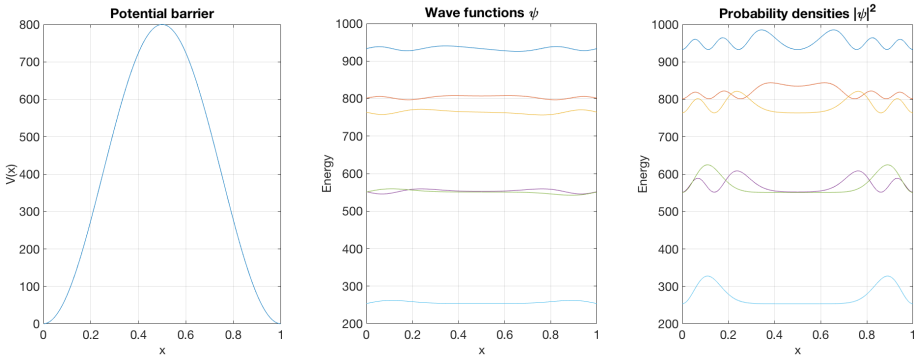
By varying $V(x)$, different potential barriers in the ∞-potential well could be studied. To start with, the potential barrier of $V(x) = 700(0.5 - |x - 0.5|)$ was plotted. Figure 7 (a) shows the potential inside the well, (b) shows the wave functions and (c) the probability densities. As expected, the probability density plot shows that the probability to find the particle at $x = 0.5$ becomes significant when the energy level of the particle is larger than the potential barrier and the parity is uneven. This is clearly shown in Figure 7 (c), for a particle at the first 2 energy levels the probability of finding the particle is zero or almost zero, and for the larger energy levels the probability is significant for the energy levels with even parity. Figure 7 (b) and (c) show the second and third energy levels at almost the same energy. This is aligned with the theory of the "doublet state" which refers to the two states of almost the same energy with different parity.[1]



(a) Potential barrier inside the potential well **(b)** Wave functions for the six lowest energy levels **(c)** Probability densities for the six lowest energy levels

Figure 7: Wave functions and probability densities for the first six energy levels in a potential well with a potential barrier: $V(x) = 700 \cdot (0.5 - |x - 0.5|)$.

Furthermore, another potential was plotted: $V(x) = 800(sin(\pi x))^2$. The potential barrier is shown in Figure 8 (a). The wave functions and the probability densities for the six lowest energy levels are presented in Figure 8 (b) and Figure 8 (c). The solver works as expected, the plots in the Figures shows that the probability of finding a particle at x=0.5 in the first four energy levels is zero or almost zero, and for the larger energy levels the probability is significant. Figure 8 (b) and (c) shows two pairs of energy levels at almost the same energy, corresponding to "doublet states".

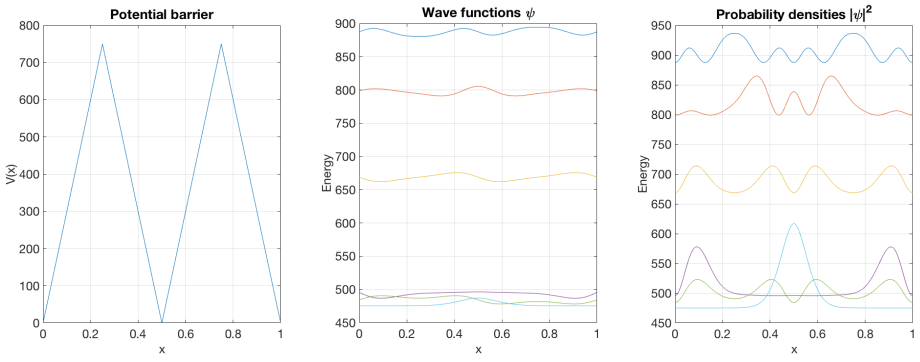


(a) Potential barrier inside the potential well **(b)** Wave functions for the six lowest energy levels **(c)** Probability densities for the six lowest energy levels

Figure 8: Wave functions and probability densities for the first six energy levels in a potential well with a potential barrier: V(x)=800$\cdot$ $(\sin(\pi x))^2$

A third well was studied with the potential barrier of $V(x) = 3000(0.5 - |x - 0.25| - |x - 0.75| + |x - 0.5|)$. The potential is plotted in Figure 9 (a), showing potential barriers which could be interpret as three wells if particle

is found at an energy level beneath the maximal potential. Figure 9 (b) and (c) shows the first three energy levels at almost the same energy levels, corresponding to the "triplet state". The Figure shows that a particle at the lowest energy level will be trapped in the "middle well", and the third can only be found in the two outer wells. This is due to the parity of the energy levels, aligned with the theory. [2]



(a) Potential barrier inside the potential well

(b) Wave functions for the six lowest energy levels

(c) Probability densities for the six lowest energy levels

Figure 9: Wave functions and probability densities for the first six energy levels in a potential well with a potential barrier: V(x)= $3000 \cdot (0.5 - |x - 0.25| - |x - 0.75| + |x - 0.5|)$.

3 Project summary

This project aimed to construct several two point boundary value problem solvers to approximate solutions to problems which engineers can encounter. The results shows that our solvers were good for a various kinds of applications, not least the common Sturm-Liouville equation.

The project per se was challenging but we learned how to tackle problems with a positive attitude and found errors quickly using the debugger. We both wrote all code and compared our programs and solvers and then worked together to improve them. We would also like to thank Tony for helping us confirm our results, and once again our fellow classmates Louise Karsten, Felicia Segui, Ebba Rickard and Anna Sjerling for discussing problems with us.

References

[1] T Stillfjord, G Söderlind, Project manual: *Project 2 in FMNN10 and NUMN12*, LTH, 2019

[2] G Ohlén *Kvantvärldens Fenomen - teori och begrepp*, 2005, Studentlitteratur, Lund

4 Appendix 1

4.1 Testfunction

```
    %% Task 1.1
clear all
L=10;
N=10;
h=L/(N+1);
x=linspace(0,L,N+2);
f= @(x) cos(x);

for i=1:N
fvec(i)=f(x(i+1));
end
alpha=-1;
beta= 0.8391;

y=twopBVP(fvec, alpha, beta, L, N)
plot(x,twopBVP(fvec, alpha, beta, L, N), 'b')
hold on
exakt = @(x) -cos(x);
fplot(exakt,[0,L], 'r')
    title('Approximation of function','fontsize', 20)
    ax = gca;
    ax.FontSize = 20;
    xlabel('x', 'fontsize', 20)
    ylabel('y','fontsize', 20)
    grid on
for i=1:N+2
exaktv(i)=exakt(x(i));
end

errfunk(f,alpha, beta, L, N, exakt)
```

4.2 Errorfunction

```
    function [] = errfunk(f, alpha, beta, L, N, exakt)
steps= N;

for k=1:steps+2
    N=2^k+1;
     x=linspace(0,L,N+2);
    for i=1:N
    fvec(i)=f(x(i+1));
    end

    for i=1:N+2
    exaktv(i)=exakt(x(i));
    end
    y=twopBVP(fvec, alpha, beta, L, N);
    hv(k)= L/(N+1);
    errv(k)=rms(exaktv-y);
end

   figure(2)
    loglog(hv,errv)
    title('Error as a function of stepsize','fontsize', 20)
    ax = gca;
    ax.FontSize = 20;
    xlabel('stepsize', 'fontsize', 20)
    ylabel('Error','fontsize', 20)
    grid on
end
```

4.3 Beam equation

```
    %% Task 1.2 - Beam equation
E=1.9e11;
Mbis=-50.*1000;
alpha=0;
beta=0;
N=99;
L=10;
I= @(x) 1e-3.*(3-2.*(cos(pi.*x./L))^12);
x=linspace(0,L,N+2);
fvec=Mbis.*ones(1,N);
M=twopBVP(fvec, alpha, beta, L, N);
for i=1:N
ubis(i)=M(i+1)./(E.*I(x(i+1)));
end
u=twopBVP(ubis, alpha,beta, L,N);
plot(x,u)
    title('Deflection of beam','fontsize', 20)
   ax = gca;
   ax.FontSize = 20;
   xlabel('Position on beam [m]', 'fontsize', 20)
   ylabel('Deflection [m]','fontsize', 20)
   grid on
answer= u(51)
```

5 Appendix 2

5.1 Sturm Liouville solver

5.2 Errorplot

```
    function [] = errSL(N,L)
%Computes error of SLsolver
steps=N;

for k=1:steps+2
    N=2^k+1;
    x=linspace(0,L,N+2);
    [eigval,~]=SLsolver(N,L)
    B(k,1:4)=eigval(length(eigval)-3:length(eigval));
    errv1(k)=rms(-pi^2-B(k,3));
    errv2(k)=rms(-4.*pi^2-B(k,2));
    errv3(k)=rms(-9.*pi^2-B(k,1));
    Nvec(k)=N;

end
figure(3)
loglog(Nvec, errv1, Nvec, errv2,Nvec,errv3)
    title('Error as a function of number of steps N','fontsize', 20)
    ax = gca;
    ax.FontSize = 20;
    xlabel('Number of steps N', 'fontsize', 20)
    ylabel('Error','fontsize', 20)
    grid on
end
```

5.3 Testfunction

För errorplotten:

```
     N=9;
 L=1;
 x=linspace(0,L,N);
[eigval, eigfunc] = SLsolver(N, L);
errSL(N,L)
```

För uträkning av egenvärdena:

```
     N=499;
 L=1;
```

```
 x=linspace(0,L,N);
[eigval, eigfunc] = SLsolver(N, L);
errSL(N,L)
```

För att plotta de tre egenfunktionerna:

```
 N=499;
 L=1;
 x=linspace(0,L,N);
[eigval, eigfunc] = SLsolver(N, L);
plot(x,eigfunc(:,N), x, eigfunc(:,N-1),x,eigfunc(:,N-2))
 title('First three eigenfunctions','fontsize', 25)
    ax = gca;
    ax.FontSize = 25;
    xlabel('x', 'fontsize', 25)
    ylabel('y','fontsize', 25)
    grid on
```

Lösning av Schrödingerekvationen V(x)=0

```
N=499;
L=1;
x=linspace(0,L,N+2);

[eigval, eigfunc] = SEsolver(N, L);
plot(x,eigfunc(:,N-6:N).*105+eigval(N-6:N))
figure(2)
plot(x,abs(eigfunc(:,N-6:N).*105).^2+eigval(N-6:N), 'r')
```

SE-solvern för V(x)=0:

```
function [eigvalv, eigfuncv] = SEsolver(N, L)
% Schrödinger equation solver
h=L/(N+1);
sub=ones(1,N-1);
main=-2.*ones(1,N);
sup=ones(1,N-1);

T = 1./(h.^2).*(diag(sub,-1) + diag(main,0) + diag(sup,1));
[eigfunc,V]=eig(T,'vector');
eigfuncv(1,:)=0;
eigfuncv(2:N+1, 1:N)=-eigfunc;
eigfuncv(N+2,:)=0;
eigval=V.';
eigvalv(1)=0;
eigvalv(2:N+1)=-eigval;
```

```
eigvalv(N+2)=0;
end
```

Med potentialer:

```
    clear all
N=499;
L=1;
x=linspace(0,L,N+2);
potential=@(x) 800.*(sin(pi.*x)).^2;

for i=1:N
    V(i)=potential(x(i+1));
end

[eigval, eigfunc] = SEsolver(N, L,V);

figure(1)
plot(x,eigfunc(:,N-5:N).*100+eigval(N-5:N))
 title('Wave functions \psi','fontsize', 25)
    ax = gca;
    ax.FontSize = 25;
    xlabel('x', 'fontsize', 25)
    ylabel('Energy','fontsize', 25)
    grid on

figure(2)
plot(x,abs(eigfunc(:,N-5:N).*100).^2+eigval(N-5:N))
 title('Probability densities |\psi|^2','fontsize', 25)
    ax = gca;
    ax.FontSize = 25;
    xlabel('x', 'fontsize', 25)
    ylabel('Energy','fontsize', 25)
    grid on

    figure(3)
    plot(x(2:N+1),V)
     title('Potential barrier','fontsize', 25)
    ax = gca;
    ax.FontSize = 25;
    xlabel('x', 'fontsize', 25)
    ylabel('V(x)','fontsize', 25)
    grid on
```