Technical University of Denmark

Niklas Flink Hansen – s092487
Alireza Nedaei – s113681

Mathias Thage Hansen – s113680

Ran Wang – s111503

1. Obligatoriske Aflevering

02601 – Introduktion til numeriske algoritmer

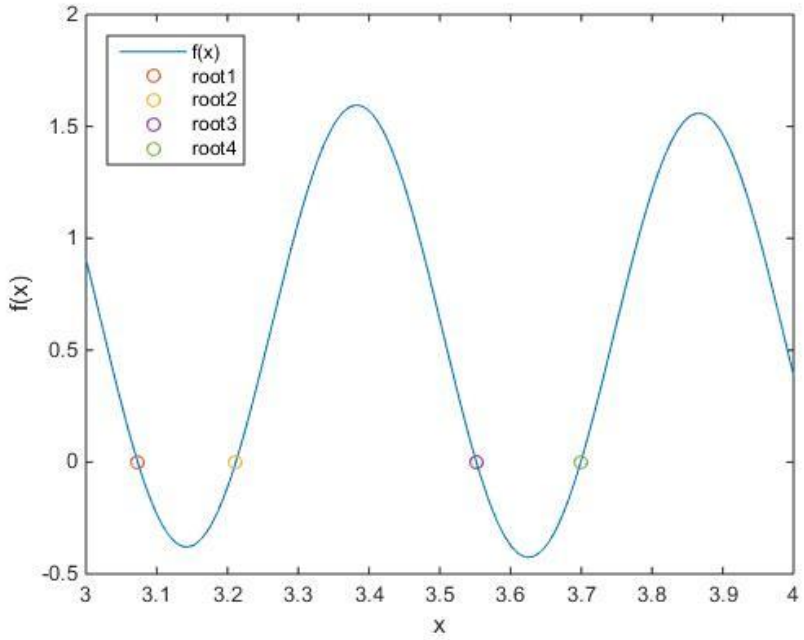
Forår 2015

1. BEREGNING AF NULPUNKT MED FZERO

1.1

$$f(x) = \cos(13x) + \frac{1.4}{\cosh(x) + 0.5}$$

Ovenstående funktion plottes i matlab i mellem intervallet $x = [3; 4]$ og med stopkriteriet $TolX = 10^{-6}$, se Figur 1.



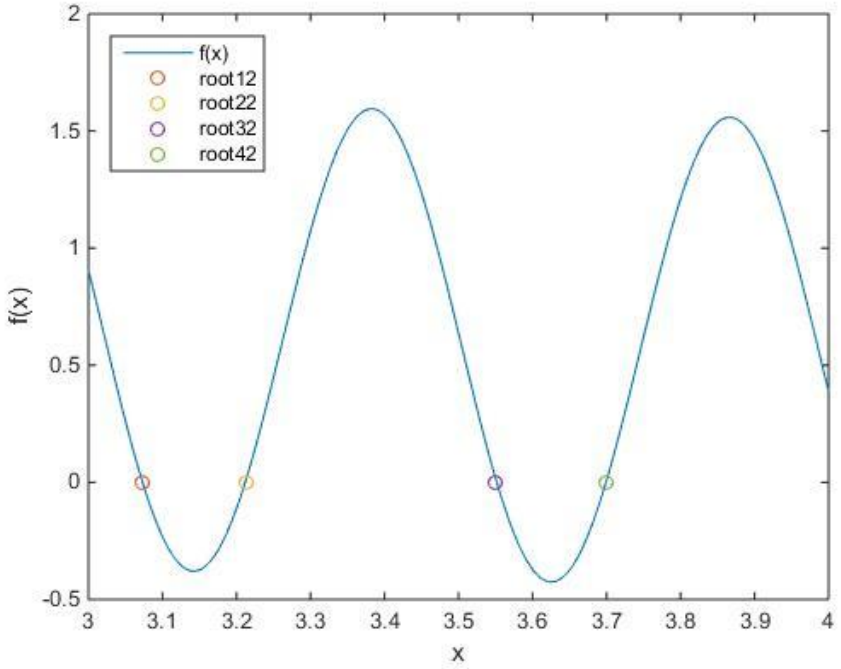
Figur 1 - Plot af f(x) med tilhørende rødder i intervallet x=[3;4] med stopkriteriet $TolX = 10^{-6}$

fzero02601 funktionen fra CampusNet bruges til at beregne samtlige rødder i det defineret interval for x. I nedenstående tabel ses rødderne, samt startgæt-intervallet og de beregnede rødder.

Rod	Startgæt-interval	Beregnede rødder
root1	x=[3;3.1]	3,0731
root2	x=[3.11;3.25]	3,2117
root3	x=[3.5;3.6]	3,5517
root4	x=[3.61;3.8]	3,6992

Tabel 1 - Startgætintervallet, samt værdien af den beregnede rod for stopkriteriet TolX=10^-6

Nu ændres stopkriteriet til $TolX = 10^{-3}$, og igen plottes funktionen for $f(x)$, se Figur 2.



Figur 2 - Plot af f(x) med tilhørende rødder i intervallet x=[3;4] med stopkriteriet $TolX = 10^{-3}$

De nye beregnede rødder med stopkriteriet $TolX = 10^{-3}$, se Tabel

Rod	Startgæt-interval	Beregnede rødder
root12	x=[3;3.1]	3,0723
root22	x=[3.11;3.25]	3,2141
root32	x=[3.5;3.6]	3,5491
root42	x=[3.61;3.8]	3,7001

Tabel 2 - Startgætintervallet, samt værdien af den beregnede rod for stopkriteriet TolX=10^-3

En vektor opstilles i matlab og differencen mellem rødderne beregnes. Resultaterne kan ses i Tabel 3.

Δ Rod	Startgæt-interval	Diff.
root1 - root12	x=[3;3.1]	0,000800
root2 - root22	x=[3.11;3.25]	-0,002400
root3 - root32	x=[3.5;3.6]	0,002600
root4 - root42	x=[3.61;3.8]	-0,000900

Tabel 3 - Difference mellem de udregnede rødder i forhold til størrelsen af stopkriteriet

1.2

I de angivet startgætintevaller $[\mathrm{x}_L, x_U] = [3.3; 3.8], [3.1; 3.8]$ og $[3.1; 3.705]$ er der tre rødder jf. Figur 1. Hvis man kører fzero-funktionen for startgætintervallet $[\mathrm{x}_L, x_U] = [3.3; 3.8]$ kan denne ikke finde nogen rødder selvom at det ses i Figur 1, at der er to rødder. Dette skyldes at begge rødder i intervallet er positive, hvilket fzero-funktionen ikke kan løse.

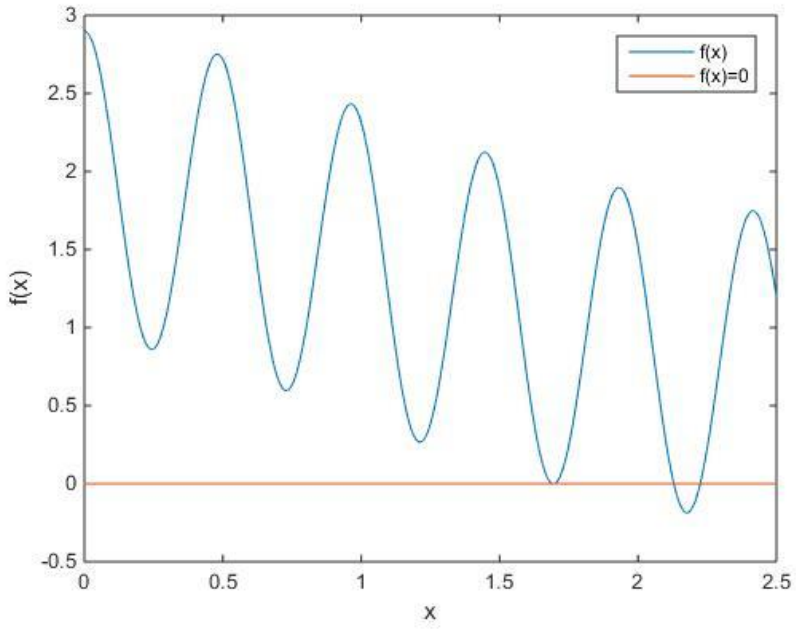
```
Error using fzero02601 (line 274)
The function values at the interval endpoints must differ in sign.

Error in opgave1 (line 49)
root121 = fzero02601(fx,[3.3 3.8],option1);
```

Figur 3 - fzero-funktion i startintervallet $[x_L, x_U] = [3.3; 3.8]$ - Matlab kode: fzero02601(fx,[3.3 3.8],option1)

I de sidste to intervaller kan fzero-funktionen beregne rødderne, men angiver kun en rod, selvom det ses i Figur 1, at der er tre rødder i de angivende intervaller. Man skal derfor være forsigtig med at bruge fzero-funktionen, da den ikke giver nogle hints om hvor mange rødder der egentlig er i startgætintervallet. Samtidig kan man se at det ikke er den samme rod som den finder, men to forskellige. Dette skyldes metoden som der bruges - bisektion eller invers kvadratisk interpolation.

1.3

Funktionen f(x) plottes nu i intervallet $x = [0; 2.5]$, se Figur 4. Ud fra en grafisk betragtning af grafen kan startgætintervallet bestemmes. Intervallet bliver bestemt til $[\mathrm{x}_L, x_U] = [2; 2.2]$. Ud fra fzero-funktionen bestemmes roden i dette startgætinterval med stopkriteriet $TolX = 10^{-9}$ til at være $root131 = 2.12920899$



Figur 4 - f(x) plottet i intervallet $x = [0; 2.5]$

Ud fra bogen[1] findes formlen for beregning af antallet af iterationer som fzero-funktionen skal bruge for at finde den mindste rod.

$$n = log_2\left(\frac{\Delta x^0}{E_{a,d}}\right)$$

hvor startgættet er $\Delta x^0 = x_u - x_l$ og $E_{a,d}$ er den maksimale fejl. n er antallet af iterationer

```
%Bisektion formel som ses på s. 138 lign. 5.6 i bogen anvendes.
%Først sættes xu og xi:
x_u=2.2;
x_l=2;
%Delta x bestemmes
Deltax=x_u-x_l;
%Den maksimale fejl defineres;
E_a=10^(-9);
%Antal iterationer for bisection:
 n=log2(Deltax/E_a);
%Det ses at i root131 der er brug for 10 itteration, trods i bisection
%skal det være 27.5454...=28 iterationer.
```

Figur 5: MATLAB koden til beregning af antallet af iterationer i forhold til bestemmelse af den mindste rod

Der skal bruges 28 iterationer (rundes op) til at bestemme den mindste rod med en maksimal fejl på $E_{a,d} = 10^{-9}$.

[1] S. 138, lign. 5.6, Applied Numerical Methods with MATLAB® for Engineers and Scientists, Steven C. Chapra, McGraw-Hil International Edition, Third Edition 2012.

2. KONVERGENS AF NUPUNKTSMETODER

2.1

Ved at omskrive Newton-Raphson funktion således den udskriver alle xi værdier i følgende:

```
function [root,ea,iter,xi]=newtraph(func,dfunc,xr,es,maxit,varargin) % har tilføjet xi i outputtet
% newtraph: Newton-Raphson root location zeroes
% [root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,p1,p2,...):
%   uses Newton-Raphson method to find the root of func
% input:
%   func = name of function
%   dfunc = name of derivative of function
%   xr = initial guess
%   es = desired relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by function
% output:
%   root = real root
%   ea = approximate relative error (%)
%   iter = number of iterations
if nargin<3,error('at least 3 input arguments required'),end
if nargin<4||isempty(es),es=0.0001;end
if nargin<5||isempty(maxit),maxit=50;end
iter = 0;
xi = xr; % Gemmer startgæt som xi.
while (1)
  xrold = xr;
  xr = xr - func(xr)/dfunc(xr);
  xi = [xi;xr]; % Udfylder nu vektoren med x-værdier (xr)
  iter = iter + 1;
  if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
```

Figur 6:Newton-Raphson

Først defineres funktion som er givet i opgaven og dens aflede. Dermed bruges Newton-Raphson metode med startgæt som 2 og 0 tolerance og 6 iterationer som er oplyst i opgaveteksten. Det laves dermed en tabel med iteratinersforløb.

xi	Ei	Ei+1/Ei^2
2	1	0,30303
1,303030303	0,30303	0,524642
1,048176519	0,048177	0,716151
1,001662171	0,001662	0,767269
1,00000212	2,12e-06	0,769215
1	3,46e-12	0
1	0	

Tael4 :iterationsforløb

Det ses at konvergensen $\frac{E(i+1)}{Ei^2}$ går mod den samme konstant og dermed har vi kvadratisk konvergens.

2.2

Ved at indføre 'iter' og 'display' i option1 og brug den rigtige tolerance fås den forekommende resultater i MATLAB.

```
Func-count                 x          f(x)
     2                     0          -0.3
     3      0.2307692307692307      -0.408284
     4       1.115384615384615       0.163314
     5      0.8626373626373627      -0.159703
     6      0.9875983861366492     -0.0159683
     7       1.000244517267474    0.000317932
     8      0.9999976453581985   -3.06103e-06
     9      0.9999999995571974   -5.75643e-10
    10      0.9999999999999999   -5.55112e-17
    11                       1   -5.55112e-17
```

```
option1 = optimset('TolX',1e-14,'display','iter');
x=fzero02601(f,[0 2],option1);
```

$$iterationer = 9$$

$$x = 1$$

Nulpunktet kan hermed findes vha. fzero02061 og et startgæt 0,2.

x	Ei	Ei+1/Ei^2
0,230769231	0,769231	0,195
1	0,115385	10,31746
0,862637363	0,137363	0,657266
0,987598386	0,012402	1,589839
1	0,000245	39,38273
0,999997645	2,35e-06	79,8658
1	4,43e-10	inf
1	0	

Tabel 5:iterationsforløb

Her ses at der ikke er kvadratisk konvergens, da $\frac{E(i+1)}{Ei^2}$ ikke nærmer sig konstant jo højere *i* bliver.

3. FUNKTION I FUNKTION

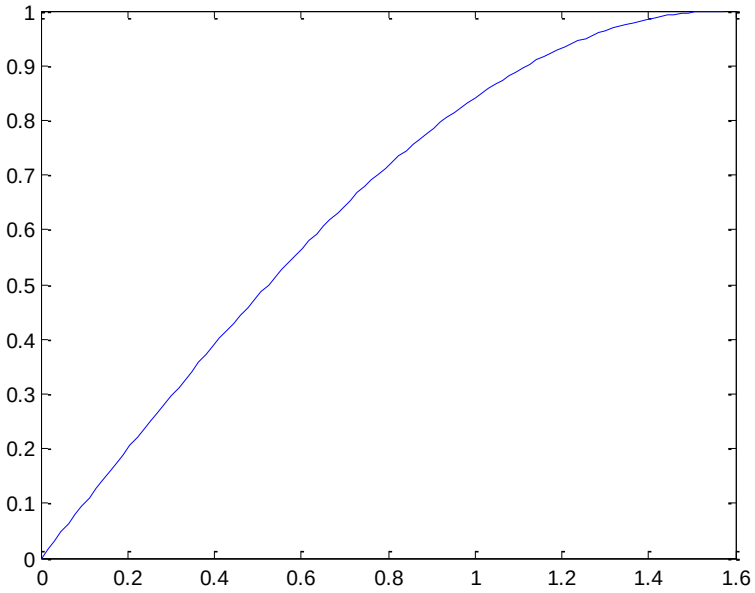
3.1

Fzero finder nulpunkt i funktion, dvs de to udtryk er samme. Ved at omskrive det til ligning fås:

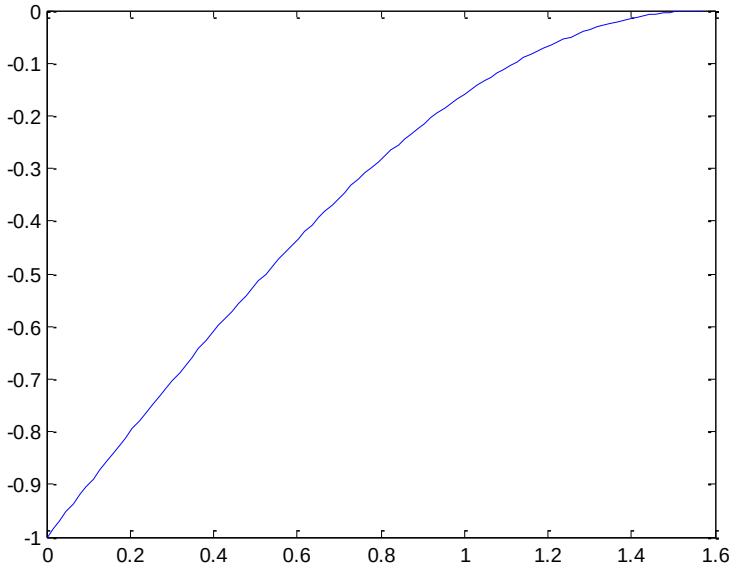
$$sin(y) - x = 0 \quad => \quad sin(y) = x \quad => \quad y = asin(x)$$

Hvor y svarer til t.

Dermed er det vist at udtrykket er det samme. Endvidere kan man sige at x skal antage værdier mellem 0.1 og 0.9, som beskrevet i funktion f. Da x flytter funktionen lodret. fzero kan dermed ikke finde nogle nulpunkter, hvis x antager værdier over 1, da funktionen dermed ligger under 1. aksen(figur 7). Omvendt kan x ikke antage værdier under 0, da funktionen dermed ikke skærer 1. aksen i det positive interval fra 0 til pi/2(figur 8).



Figur 7:



Figur 8

3.2

Ved at gøre brug af teknikken i 3.1 som viser (sin(y) - x) and asin(x) er ækvivalent, opstilles FUN som er en funktion i en anden funktion.

```
function [ fx ] = Fun( x )
% Functions navn er sat sum FUN med variabel: x and
% output parameter som fx. x er et tal mellem 0.1 and 0.9
roden = fzero( @ (y) (sin(y) - x), [0 pi/2]);
fx = roden*x^2+sqrt(x)-1;

end
```

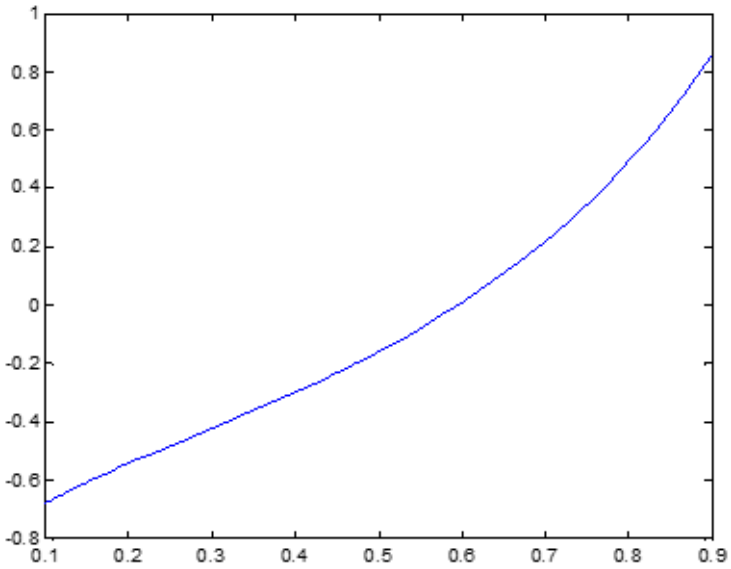
Figur 9: FUN funktion

3.3

Ved at bruge fzero findes nulpunkt.

```
nulpunkt = fzero(@Fun,0.5);
%rodden skal være tæt på 0.6
```

$$Rod = 0.596637275155296$$



Figur 10:fzero for at finde nulpunkt

4.OPTIMERING MED EN UBEKENDT

4.1

Formålet ved at bruge" golden section search "er at finde minimum for funktion f, samt antal iterationer. Først vælges den start gæt som er givet i opgaven [0,2] som repræsenter 0 som nedre grænser og øvre grænser. Samt es som er den ønskelige relativ fejl er sat til 0.1 som referer til 10^{-3}.

$$x = 0.979166867601577$$

$$iterationer = 14$$

4.2

Ved at isolere k i formel 5 i opgaveteksten vha. MAPLE fås:

```
> solve(tol = (2 − phi)·( phi − 1)^(k−1)·(xu − xl), k)
```

$$\frac{\ln(\phi - 1) + \ln\left(\frac{tol}{\phi xl - \phi xu - 2xl + 2xu}\right)}{\ln(\phi - 1)}$$

Denne ligning som definere k, indsættes i matlab og dermed findes k.

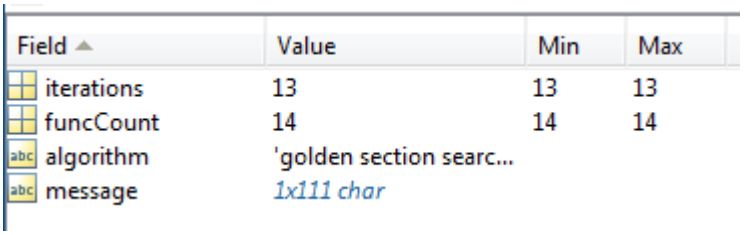
```
%4.2
tol=10^(-12);
%phi er definiret i side 187
phi=(1+sqrt(5))/2;
xl=0;
xu=2;
k=(log(phi-1)+log(tol/(phi*xl-phi*xu-2*xl+2*xu)))/log(phi-1);
```

$$k = 57.860083691792550$$

K rundes op til 58. Dvs der skal bruges 58 iterationer for at anskaf en tolerance svarende til $ek \leq tol$.

4.3

$$x_{min} = 0.978782315783540$$



Field	Value	Min	Max
iterations	13	13	13
funcCount	14	14	14
algorithm	'golden section searc...		
message	*1x111 char*		

Det ses at fmin bruges 13 iterationer i sammenligning med golden section search metode som brugt 58. Dette skyldes at fminbnd benytter sig af den langsomme robuste golden section search metode og derudover den hurtigere parabolske.

5. OPTIMERING MED FLERE UBEKENDTE

5.1

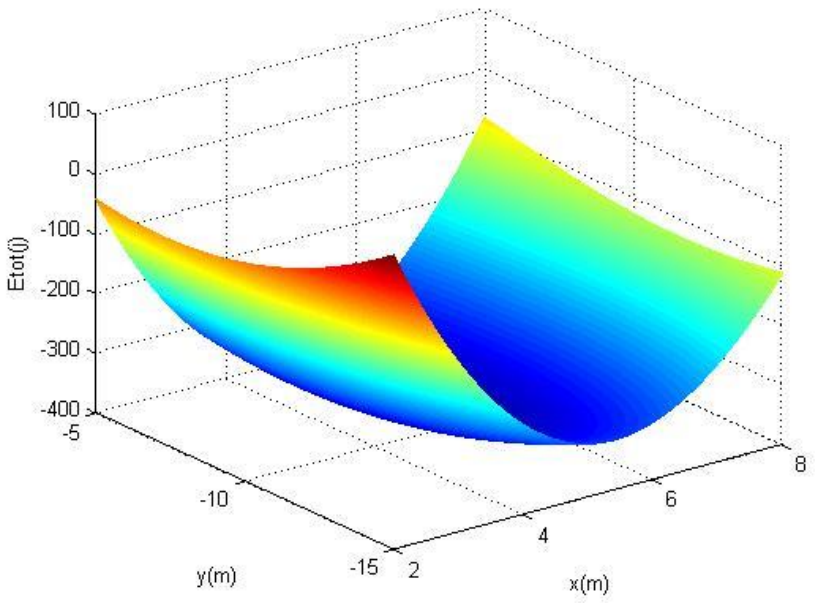
Konstanter mm. Er indsat i funktion Etot. Det skal bemærkes af g er varierende da det vil være nemmer at løse andre del opgaver som vil være afhængig af g.

```
function [ E ] = Etot( x1,y1,g )

% Input x1 og y1 er en skalar.

% Fjederkonstanter:
k1 = 10; % [N/m]
k2 = 20; % [N/m]
% Masse:
m = 4.5; % [kg]
%udstrakt længde
l=10;
% Position i loddet:
L1 = sqrt((x1^2)+(y1^2));
L2 =  sqrt((x1-10)^2+(y1-1)^2);

% Samlet funktion med Potentiel energi, fjeder og Potentiel energi,tyngdekraft: (vektor)
E = sum ( 0.5*(k1.*(abs(L1)-l).^2)+ 0.5*(k2.*(abs(L2)-l).^2)+ m*g*y1 );
end
```

Figur 11: Etot function opstilling

5.2

Ved at brug mesh, plottes Etot i 3 D. Plottet kan ses på figur 12.



Figur 12:3D plot

5.3

Bruger fminsearch til at finde minimum for funktion Etot med variabel x1, y1 og g. Hermed vælges et startgæt som er udtaget fra grafen i opgave 5.2 som er 6,-10.

```
result1 = fminsearch(@(L) Etot(L(1),L(2),g),[6,-10]);
```

Svaret:

$$x = 6.354497427384965[m] \qquad y = -9.939503542252844[m]$$

5.4

Da g er defineret som variabel i Etot funktion, er det en dele nemmer at sætte den=0 nu. Hermed fås nogle ny værdier hvilket repræsentere placering af kassen når tyngdeacceleration=0

```
%5.4
g=0;
result2 = fminsearch(@(L) Etot(L(1),L(2),g),[6,-10]);
```

Svaret:

$$x = 5.860311866023631[m] \qquad y = -8.102865685195283[m]$$

Det kan ses at kassen besværger sig op ad i y-aksen, hvilken giver fint mening når der ikke er påført tyngdekraft i y aksen ved at sætte g=0.

Et andet testtilfælde kan være at sætte masse= 0 og g=9.82 og kører programmet igen, hermed ses at det opnås det samme resultater som 5.3 hvilket viser at systemet er uafhængig af massen og fjederkraften virker som det skal.

5.5

For at finde den anden lokalt minimum, kan funktion differentieres i x og y og dermed solves for at finde x og y.

```
%5.5
%sættes g=9.82 igen
g=9.82;
syms L1 L2;
f= Etot(L1,L2,g);
%diff funktion Etot både L1 og L2.
fx=diff(f,L1);fy=diff(f,L2);
[a,b]=solve(fx,fy);
double([a,b])
```

$$x = 2.219089014580383487635[m] \qquad y = 2.400968711837420011804 0[m]$$

A. DET HÆGENDE KABEL

A.1

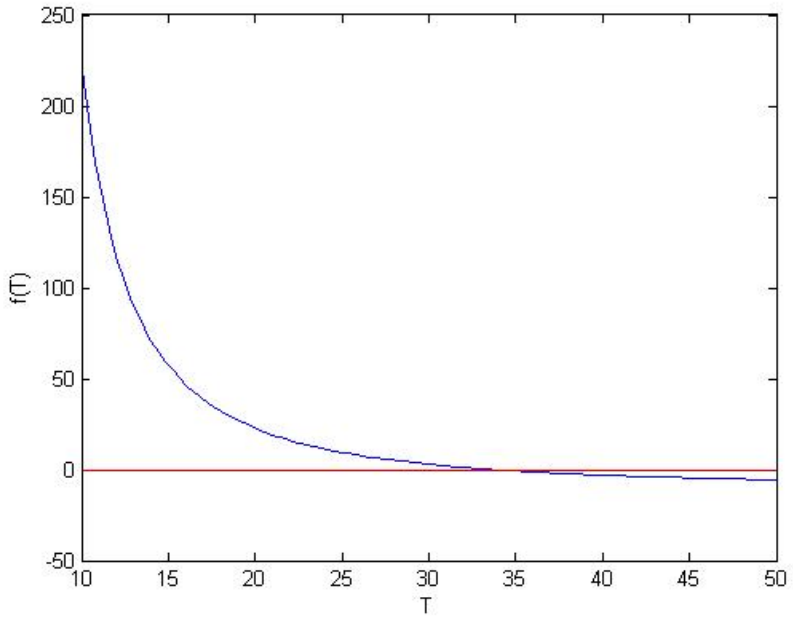
Den nemmeste måde at gøre på, er at samle leddene på højre siden i ligning 7 fra opgaveteksten. Derefter sættes ligning er lige med 0. Dermed er det muligt at anvende nulpunktproblemet.

```
% definerer vores funktion (nulpunktsproblem), samles ligning nr 7 på
% højreside og sættes =0 så har vi:
f=@(T)  T/w.*sinh((L.*w)./(2.*T))-(1./2).*S;
```

A.2

Ved at plotte den fundne funktion fra A 1, findes et passende startgæt interval.

```
%A2
% plotter for at finde startgæt
plot1 = linspace(10,50);
plot(plot1,f(plot1),plot1,zeros(size(plot1,2)))
%ved at plotte det ses at funktion skærer x-aksen i [32,36] .
xlabel('T');
ylabel('f(T)')
```



Figur 13:startgæt

Ud fra plottet ses at funktion skærer i et interval mellem [32,36]. Dette bruges til næste opgave.

A.3

I denne opgave bruges startgættet fra opgave A2. Først findes en rod med gæt på en tolerance 1e-3, som kan bruges til at bestemme tolerance. Dette sættes i den følgende ligning for at finde den korrekte tolerance.

```
Tolx= 0.01/(2*abs(T-1));
```

```
 Func-count     x          f(x)             Procedure
     2             36       -1.2463         initial
     3          34.012     -0.120252         interpolation
     4         33.8182    0.00123753         interpolation
     5         33.8182    0.00123753         interpolation

Zero found in the interval [32, 36]
```

Field	Value	Min	Max
intervaliterations	0	0	0
iterations	3	3	3
funcCount	5	5	5
algorithm	'bisection, interpolati...		
message	'Zero found in the int...		

Denne Tolx kommer i outputtet som sættes ind i funktion og køres igen. Resultater ses på nedenstående figur som er udtaget fra MATLAB. Det viser sig at fzero brugt 3 iterationer til at finde frem til **T=33.8182317697802** med den ønskede tolerance. (Tolx=1.523543387430190e-04)

Value	Name
@(T)T/w.*sinh((L.*w)...	f
26.5456	H_max
80	L
1x1 struct	optionA
1x1 struct	output
1x100 double	plot1
100	S
33.8182	T
1.5235e-04	Tolx
1	w
40	x
[32 36]	x0

A4

$$H_{max} = H\left(\frac{1}{2} * L\right) = 26.545636186788364$$

Hvor L=80 m fra opgaveteksten.

```
%A4
x=L/2;
H_max=(T/w)*(cosh(w*x/T)-1);
```

OPGAVE B. DESIGN OG PRIS FOR HÆNGEBRO

B.1

MATLAB-funktion for beregning af prisen for en given kabellængde S, afstand L, massetæthed ω og bropillehøjde B;

```
function [ P ] = pris( S, omega, B, L )
%startgættet
x0 = 30;
%Definition af funktionen
f = @(T)  T/omega.*sinh((L.*omega)./(2.*T))-(1./2).*S;
%Definering og et en startantagelse af tolerancen, TolX=1e-5
option1=optimset('TolX',1e-5);
%fzero funktionen bruges til at finde T
T = fzero(f,x0,option1);
%definerer nu x=L/2
x = L/2;
h_max = (T/omega)*(cosh(omega*x/T)-1);
clear 'x'
for i = 1:13
x(i) = (i-1)*L/12-L/2;
V (i) = B - h_max + (T/omega)*(cosh(omega*x(i)/T)-1);
end
P = 10*S*omega + 10000*(1-omega)^2 + sum(V);
end
```

Figur 14 - Funktionen for beregning af prisen for broen ud fra variablerne S,L, ω og B

Prisen for broen med et hængende kabel som fundet i opgave A.1-A.4 og med en bropillehøjde på $B = 30$

$$P = 1.174578375175209e + 03$$

Mindste prisen for broen fundet ved funktionen fminsearch

```
%Bruger funktionen fminsearch til at bestemme mindste prisen
[m1, Pmin1] = fminsearch( @ (x) pris(x(1),x(2),B,L),[100,1]);
```

Tabel 2 - funktionen fminsearch til bestemmelse af mindste prisen for broen

Prisen P(S, w, B) omskrives til en funktion af de to variable S og w (B antages en konstant). Den indbyggede funktion fminsearch bruges til at beregne den minimale pris Pmin for en bro med længde L = 80 og B = 30.

$$m1 = [86.272147171809500 \qquad 0.956871163431398]$$

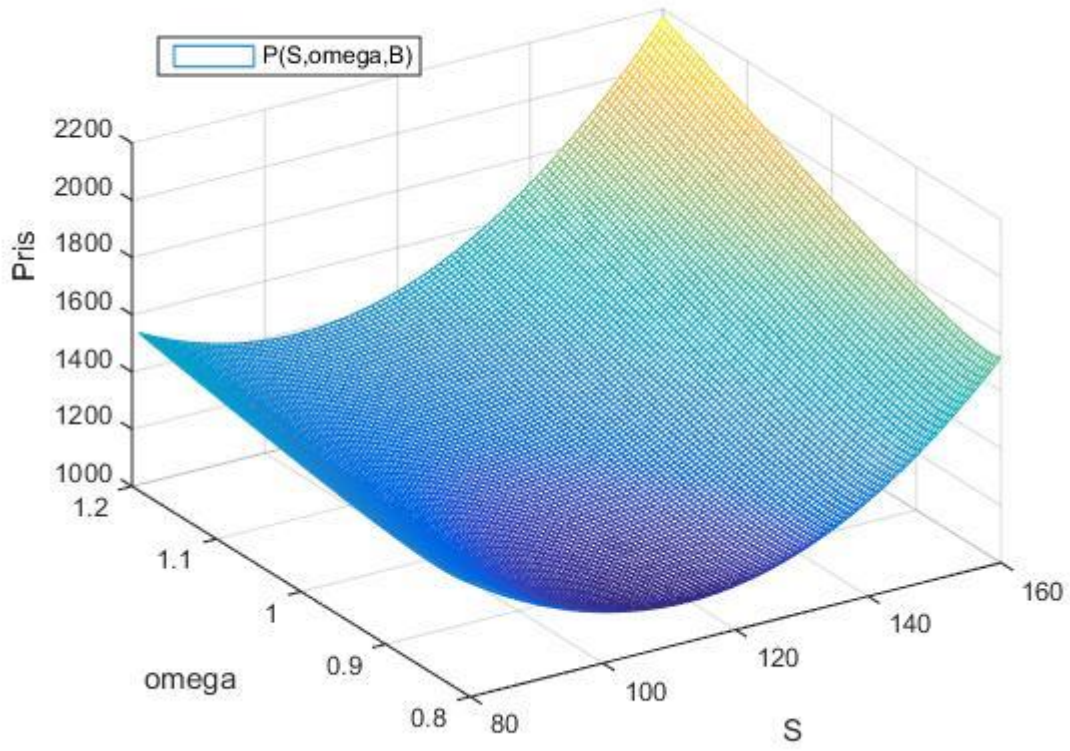
$$Pmin1 = 1.121372057821103e + 03$$

B.2

Prisfunktionen plottes i 3D, hvor værdier for S og ω skal være inden for intervallerne

$$L + 1 \leq S \leq 2L$$

$$0.8 \leq \omega \leq 1.2$$



Figur 15 - 3D plot af prisfunktionen

Fra plottet bestemmes startgættet til $x_0 = [86.2; 0.956]$

B.3

Prisen bliver beregnet vha. fminsearch. Pris, tolerance TolX, samt antallet af iterationer kan ses i ovenstående Figur. Startgættet x_0 er fundet i B2.

```
Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-03
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04

>> m2

m2 =

   86.3184    0.9568

>> Pmin2

Pmin2 =

   1.1214e+03

>> output

output =

    iterations: 30
     funcCount: 61
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.Tol...'
```

```
[m2, Pmin2, ~, output] = fminsearch( @ (x) pris(x(1),x(2),B,L),x0,optionB);
```

Tabel 3 - Miniums Prisen, Tolerance og iterationer bestemt ud fra funktionen fminsearch

B.4

```
%B4
%Ved at kigge på grafen ses at hmin ligger i x=0.
%Hvis det indsætte fås det hele til 0, derfor kan det ikke bruges ligesom
%i opgaveA. Dette skyldes at vi i opgave B nu kigger på et nyt
%koordinatsystem. I stedet kan h_min findes ud fra sammenhængen:
%h_min + h_max = B <=> h_min = B - h_max
%h_max beregnes på samme måde som før, men med S og omega sat til de
%værdier fundet for den billigste brokonstruktion.
%ved at kigge på m værdien, bestemmes S og omega og rundes op
S = 87;
omega = 0.96;
% B = 30; % samme som tidligere
% L = 80; % samme som tidligere
% startgæt
x0 = 30;
%Definering af funktion f(T)
f = @(T)  T/omega.*sinh((L.*omega)./(2.*T))-(1./2).*S;
%Definering af tolerance til fzero.
%starter med et gæt på tolerancen = 1e-3
optionB1=optimset('TolX',1e-3);
%fzero bruges til at bestemme T
T = fzero(f,x0,optionB1);
%Defination af x = L/2
x = L/2;
h_max = (T/omega)*(cosh(omega*x/T)-1);
h_min = B - h_max;
```

h_max og h_min er nu fundet

```
>> h_max

h_max =

   14.9178

>> h_min

h_min =

   15.0822
```

OPGAVE C. UNDERSØGELSE AF SEKANTMETODENS KONVERGENS

C.1

Til at implementer sekant metoden har vi lavet en funktion der laver beregningerne. Funktionen får kurvefunktionen, X_-1 og x_0 ind og så kan den udregne med sekant metoden. Det den ellers skal have information er det maksimale antal iterationer og den maksimale fejl.

```
function [ x ] = secantroot(f,Xa,Xb,err,imax )
%Denne funktion beregner nulpunkter for en ligning med sekant metoden.
 %f er ligningen
 %Xa er x_-1
 %Xb er x_0
 %err er den maksimale størrelse af fejlen.
 %imax er det maksimale antal interationer beregningen må lave
 x=[];
 for i=1:imax
     funxb=feval(f,Xb);
     x(i)=Xb-(funxb*(Xa-Xb))/(feval(f,Xa)-funxb);
     if abs((x(i)-Xb)/Xb)<err %Her tjekkes om fejlen er mindre en den satte værdi
        xs=x(i);
        break
     end
     Xa=Xb;
     Xb=x(i);
 end
 %outputtet er så en vektor med x værdierne for hver interation.
```

Figur 16:secant function opstilling

Det der så kommer som output er en vektor med x værdierne for hver udregning funktionen laver.

C.2

I denne opgave skal den lavet funktion testes på problemet fra opgave 2 med startgæt x_-1=0.3 og x_0=1.5 og der skal derefter finde afstanden fra roden E_i.

X_i	E_i
0,6818	0,3182
0,8926	0,1074
1,0391	0,0391
0,99659	0,0034
0,9999	9,9592
1	2,61e-7
1	2e-11
1	0

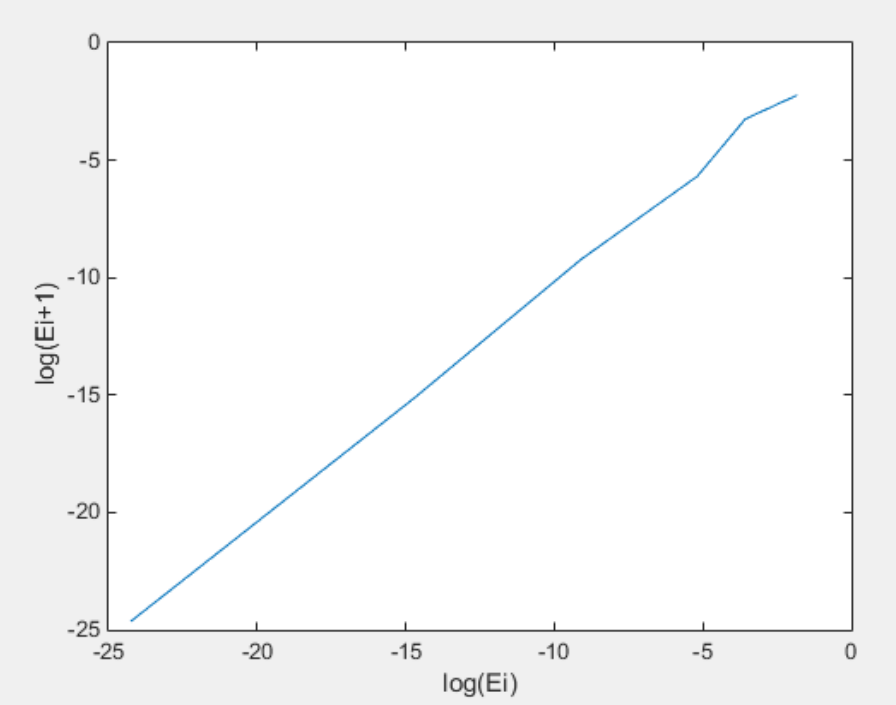
For funktionen med secant metoden skal den bruge 8 iterationer for at finde frem til en rod med en fejl på mindre end 10^-14.

C.3

For at finde relationen mellem E_i og E_i+1 skal der bestemmes en alpha i ligningen:

$$E_{i+1} = kE_i^{\alpha},$$

Denne alpha værdi bestemmes ved at plotte log(E_i) vs log(Ei+1) og det giver grafen:



Figur 17: plot af log(Ei+1) oglog(Ei)

For at hældningen på linjen er 1 skal alpha være 1.6.

C.4

Hvis der byttes om på x_-1 og x_0 vil der komme det samme resultat men der skal flere iterationer til at lave beregningen fordi den vil krydse skæringspunktet flere gange.

X_i
0,6818
1,7903
0,8581
0,9424
1,0074
0,9997
0,999998
1
1

Tabel 6

Det vil kræve 9 iterationer for at være inden for den maksimale fejl.

TEORI-SPØRGSMÅL – VALG AF METODE

TEORI-SPØRGSMÅL 1

Newton-Raphson metode

Fordele

- Konvergerer hurtigt (kvadratisk). Korrektheden af betydende cifre fordobles efter hver iteration der foretages.
- et startgæt er nødvendigt.

Ulemper

- Differentialkvotienten er krævet.
- Metoden er ikke (speciel) robust.
- Hvis tangenten er parallel eller næsten parallel med x-aksen, kan metoden ikke konvergere.
- Normalt forventes Newton-raphson metoden kun at konvergere når den er tæt på løsningen.

Sekant metode

Fordele

- Den kræver kun én funktionsevaluering per iteration, hvor Newton-Raphson kræver to.
- Den kræver ikke funktionens afledte for at virke. Endvidere behøves funktionen ej heller at være differentiabel

Ulemper

- Man har ikke kontrol over konvergensen og fejlen. Derfor kan man ikke være sikker på at roden nødvendigvis ligger indenfor intervallet. Det betyder at for ikke kontinuerlige funktioner begynder/kan metoden ikke nødvendigvis konvergere.
- Den konvergerer først når startgættet er tæt på roden.
- Langsommere end Newton-Raphson metoden.

Bisektion metode

Fordele

- Metoden konvergerer altid.
- Efter hver iteration bliver fejlen halveret.

Ulemper

- Den konvergerer langsomt.
- Kan kun finde én rod.
- Man skal starte med to startgæt.

Invers kvadratisk metode

Fordele

- hurtig
- robust
- Kræver ikke nogen afledede

Ulemper

- kræver tre punkter
- metoden er ineffektiv hvis startgættet er langt fra roden. (Derfor bruger man Brents metode hvor man først bruger den robuste bisektion og derefter invers kvadratisk interpolation.)

TEORI-SPØRGSMÅL 2 – KVADRATISK KONVERGENS

Kvadratisk konvergens betyder at metoden under speciel defineret antagelser konvergerer således at antallet af korrekte betydende cifre fordobles for hver iteration. Det vil sige at når forholdet E_{i+1}/E_i^2 nærmer sig en konstant nærmer fejlen sig 0. Hvis roden x* er bestemt kan konvergensen bestemmes med nedenstående formler.

$$E_{i+1} = |x_{i+1} - x^*|, \quad E_i = |x_i - x^*|$$

$$E_{i+1} = \frac{-f''(x^*)}{2f'(x^*)} E_i^2 = KE_i^2$$

TEORI-SPØRGSMÅL 3 – KONVERGENS AF "GOLDEN SECTION SEARCH"

Tre antagelser for funktionen f(x) gælder i det valgte interval [a,b]:

- Det antages at funktionen er kontinuert mellem startgættet $x_l \; og \; x_u$
- At den er unimodal. Det vil sige at den kun har et højeste maksimum.
- Kun har ét minimum

TEORI-SPØRGSMÅL 4 – DET GYLDNE SNIT

Faktoren ϕ indgår i placeringen af de to nye punkter ved "golden section search". Definitionerne af de nye punkter er følgende:

$$x_1 = x_l + d$$

$$x_2 = x_u + d$$

Hvor d er,

$$d = (\varphi - 1)(x_u - x_l)$$

Fordelen ved dette er at der spares en del funktionsberegninger, da funktionen betragtes på disse nye punkter og algoritmen smider intervallet uden minimumspunktet væk. Dette vil sige at algoritmen genbruger den gamle beregnede værdi af enten $x_1 \; eller \; x_2$.

TEORI-SPØRGSMÅL 5 - FEJLVURDERING

Den maksimale fejl er givet ved (2-phi) gange intervalbredden, som det også står formuleret ispørgsmål 4.2 Dvs. :

$$(2 - \phi)(x_u - x_l)$$

Idet xu beskriver den øvre grænse og xl beskriver den nedre grænse. Derudover reduceres fejlen med en faktor (phi-1) for hver iteration. Hvilket også er beskrevet i spørgsmål 4.2. Dvs. denne faktor skal ganges på ovenstående udtryk for hver iteration efter den første iteration. Vi ved at:

$$(\phi - 1)(\phi - 1) = (\phi - 1)^2$$

Vi får dermed alt i alt:

$$(2 - \phi)(x_u - x_l)(\phi - 1)^{k-1}$$

Der står som sagt k-1, da fejlen ikke bliver reduceret første gang, svarende til 1-1. Noget opløftet i 0 giver 1 og dermed passer formlen.

APPENDIX – MATLAB KODE

OPGAVE 1 – MATLAB KODE

```
clear all
close all
clc
%1.1 - Beregning af nulpunkt med fzero
%Definition af funktionen, samt plot af denne i graf i intervallet x=[3;4]
fx=@(x) cos(13*x)+1.4/cosh(x)+0.5;
figure()
fplot(fx,[3,4]); hold on;
%Startgætintervallerne bestemmes ud fra grafisk betragtning
%af den plottet funktion
%Tolerance 10^-6 indsættes vha. funktion optimset
option1 = optimset('TolX',1e-6);
%Rødderne defineres med fzero funktionen, startgætintervallet
%og tolerance TolX. - fzero(fun,startgæt,tolerance)
root1=fzero02601(fx,[3 3.1],option1);
root2=fzero02601(fx,[3.11 3.25],option1);
root3=fzero02601(fx,[3.5 3.6],option1);
root4=fzero02601(fx,[3.61 3.8],option1);
%Rødder plottes i samme graf som funktionen
plot(root1,0,'o');plot(root2,0,'o');plot(root3,0,'o');plot(root4,0,'o');
%Indsættelse af legends og akse-labels
legend('f(x)','root1','root2','root3','root4','Location','northwest')
xlabel('x')
ylabel('f(x)')
hold off;
%Samme operation gentages med tolerance TolX=10^-3 og nye rødder beregnes
figure()
fplot(fx,[3,4]); hold on;
option2 = optimset('TolX',1e-3);
root12=fzero02601(fx,[3 3.1],option2);
root22=fzero02601(fx,[3.11 3.25],option2);
root32=fzero02601(fx,[3.5 3.6],option2);
root42=fzero02601(fx,[3.61 3.8],option2);
plot(root12,0,'o');plot(root22,0,'o');plot(root32,0,'o');plot(root42,0,'o');
legend('f(x)','root12','root22','root32','root42','Location','northwest')
xlabel('x')
ylabel('f(x)')
hold off;
%De 8 rødder samles i forhold til stopkriteriet og en difference mellem
%de to stopkriterier kan nu bestemmes:
%Stopkriterierne TolX6 og TolX3
TolX6 = [root1, root2, root3, root4];
TolX3 = [root12, root22, root32, root42];
%Difference mellem de enkelte rødder i forhold til ændringen af
%stopkriteriet:
Diff = TolX6-TolX3
%%

%1.2
%root121 = fzero02601(fx,[3.3 3.8],option1);
%Det virker ikke da alle x værdier har positiv y værdier men de skal også
%have negativ værdier, -> ingen rødder
```

```
root122 = fzero02601(fx,[3.1 3.8],option1);
%Den virker da y værdien er henholdsvis har negativ og positiv værdier.
%Funktion fzero finder kun 1 af 3 rødder.
root123 = fzero02601(fx,[3.1 3.705],option1);
%Her virker det igen for de valgte x værdier, da der er
%både negativ og positiv y værdier. Der er en forskel på værdien af roden.
%%

%1.3
figure()
%Intervallet ændres til x=[0;2.5], da den mindste positive rod skal
%bestemmes. Dette gøres ud fra en grafisk betragtning.
fplot(fx,[0,2.5]);hold on;
fplot(@(x) 0,[0,2.5]);
legend('f(x)','f(x)=0','Location','northeast')
xlabel('x')
ylabel('f(x)')
option3 = optimset('TolX',1e-9,'display','iter');
root131=fzero02601(fx,[2 2.2],option3);
%Bisection formel som ses på s. 138 lign. 5.6 i bogen anvendes.
%Først sættes xu og xi:
x_u=2.2;
x_l=2;
%Delta x bestemmes
Deltax=x_u-x_l;
%Den maksimale fejl defineres;
E_a=10^(-9);
%Antal iterationer for bisection:
 n=log2(Deltax/E_a);
%Det ses at i root131 der er brug for 10 itteration, trods i bisection
%skal det være 27.5454...=28 iterationer.
```

OPGAVE 2 – MATLAB KODE

```
close all; clear all; clc;
%opgave 2.1
f = @(x) x^2 -0.7*x -0.3;
 df = @(x) 2*x -0.7;
 [root,ea,iter,xi] = newtraph(f,df,2,0,6);
 %root=1 og xi viser tabel under de 6 itterationer
 Ei=abs(xi-root);
 Ei1=[];
 for i=1:length(Ei)-1
 Ei1 = [Ei1;(Ei(1+i)/(Ei(i)^2))];
 end
 Ei1;

%opgave 2.2
option1 = optimset('TolX',1e-14,'display','iter');
root1=fzero02601(f,[0 2],option1);
%hvis Ei+1 går i mod konstant så er det kvadratisk konvegens ligesom
%2.1(Ei1 i programmet). det ses i 2.2 at den går som sinus/noget i den
%stil kurve derfor er det IKKE kvadratisk konvergens.
```

OPGAVE 3 – MATLAB KODE

```
close all; clear all; clc;
f = @(x) asin(x)*x^2+sqrt(x)-1; % 0.1<=x<=0.9
%3.1 de to funktioner beregnes med random x og ses at være ens
xrandom=0.45;
t1 = fzero(@(y) (sin(y)-xrandom) , [0 pi/2]);
t2=asin(xrandom);
%det ses funktion er defineret mellem 0.1 og 0.9 dvs fzero ikke kan finde,
%nogle værdier hvis x antages >1. Desuden kan x antages hellere ikke mindre
%end 0 da funktion ikke skær 1. aksen i det positiv interval mellem 0 pg pi/2.
xplot = linspace(0,pi/2);
f1 = sin(xplot)-1;
f2 = sin(xplot);
figure(1)
plot(xplot,f1)
figure(2)
plot(xplot,f2)
%solves i maple for y i den første ligning og sættes i den anden.
%%

%3.2
answer = Fun(0.45);

%3.3
nulpunkt = fzero(@Fun,0.5);
%rodden skal være tæt på 0.6
```

OPGAVE 4 – MATLAB KODE

```
close all; clear all; clc;
f = @(x) 1.5*x^6 + 2*x^2 - 12*x;
%4.1
[x,fx,ea,iter] = goldmin(f,0,2,0.1);
%det viser sig 14 itterationer og løsning som er minimum er x=0.9792
%4.2
tol=10^(-12);
%phi er definiret i side 187
phi=(1+sqrt(5))/2;
xl=0;
xu=2;
k=(log(phi-1)+log(tol/(phi*xl-phi*xu-2*xl+2*xu)))/log(phi-1);
%det laves solve fra ligning og k isoleres vha maple, der ved vælges lower
%og upper bund 0,2 og så findes k som 57.8601 = 58
%4.3% funktionen fminbnd er en indbygget matlab funktion, der finder location
% og value for min. xmin er location og fval er value.
option43= optimset('TolX',1e-12,'display','iter');
%det kan gøres på forskellige måder, x_new resultat og xmin resultat er det
%samme=0.9788
x_new = fminbnd(f,0,2,option43);
[xmin, fval, exitflag, output] = fminbnd(f,xl,xu,option43);
% som det ses via at brug matlab funktion, laves kun 11 iterationer til at
% beregne men goldmin bruges 58. Dette skyldes at
%fminbnd benytter sig af den langsomme robuste golden section search metode
%og derudover den hurtigere parabolske interpolation.
```

OPGAVE 5 – MATLAB KODE

```
close all; clear all; clc;
%5.1
x1 = 5;
y1 = 1;
g=9.82;
E1 = Etot(x1,y1,g);
%Det laves en Etot funktion for at definere vores samlet energi, denne
%funktion er laves udefra de ændring i koordinater vi havde.
%5.2
% Definerer mine 2 vektorer x1 og y1.
x1 = linspace(2,8,1000);
y1 = linspace(-15,-5,1000);

% Laver tom matrix
E = zeros(length(x1),length(y1));

% Laver matrix med værdierne for den samlede energi ved forskellige x1 og
% y1.
for i = 1:size(x1,2)
    for j = 1:size(y1,2)
% Samlet funktion med Potentiel energi, fjeder og Potentiel energi, masse:
(vektor)
E(i,j) = Etot(x1(i),y1(j),g);
    end
end

% Opretter figur:
figure(1);

% Laver mesh med værdierne for x1 og y1 ud af 1. og 2. aksen og energien op
% af 3. aksen.
mesh(x1,y1,E)

%5.3
result1 = fminsearch(@(L) Etot(L(1),L(2),g),[6,-10]);

%5.4
g=0;
result2 = fminsearch(@(L) Etot(L(1),L(2),g),[6,-10]);
%5.5
g=9.82;
syms L1 L2;
f= Etot(L1,L2,g);
fx=diff(f,L1);fy=diff(f,L2);
[a,b]=solve(fx,fy);
double([a,b])
```

OPGAVE A – MATLAB KODE

```
%% A .1
```

```
% definerer konstanter
L=80;
w=1;
S=100;

% definerer vores funktion (nulpunktsproblem), samles ligning nr 7 på
% højreside og sættes =0 så har vi:
f=@(T)  T/w.*sinh((L.*w)./(2.*T))-(1./2).*S;

%A2
% plotter for at finde startgæt
plot1 = linspace(10,40);
plot(plot1,f(plot1),plot1,zeros(size(plot1,2)))
%ved at plotte det ses at funktion skærer x-aksen i [32,36] .

%A3
optionA= optimset('TolX',1e-3,'display','iter');
%indføres start gæt
x0=[32,36];
%fzero defineres
T = fzero(f,x0,optionA);
%nu findes den rigtige tollerance udfra den given
Tolx= 0.01/(2*abs(T-1));
%det ses at der skal bruges 3 itterationer til at finde T=33.8182

%A4
x=L/2;
H_max=(T/w)*(cosh(w*x/T)-1);
```

OPGAVE B – MATLAB KODE

```
close all; clear all; clc;
S=100;
B=30;
L=80;
omega=1;
P = pris(S,omega,B,L);

%Bruger funktionen fminsearch til at bestemme mindste prisen
[m1, Pmin1] = fminsearch( @ (x) pris(x(1),x(2),B,L),[100,1]);

clear 'S'
clear 'omega'

%Der er lavede en pris funktion. Den beregnede pris er P=1.1746*1+^3
%for B=30. For at beregne mindste prisen, bruges fminsearch, hvor er L=80,

%B2.
%Definerer grænseværdierne for S og omega
S = linspace(81,160,100);
omega = linspace(0.8,1.2,100);
%det laves en tom vektor
P1 = zeros(length(S),length(omega));

for i = 1:size(S,2)
    for j = 1:size(omega,2)
P1(i,j) = pris(S(i),omega(j),B,L);
    end
end

% Opretter figur:
figure();

% Laver mesh med værdierne for S og omega.
mesh(S,omega,P1)
xlabel('S')
ylabel('omega')
zlabel('Pris')
legend('P(S,omega,B)','Location','northwest')
%B.3
%fminsearch bruges til at finde minimum for funktion med vektor
%Tolerance er sat til 10^-5
optionB= optimset('TolX',1e-3,'display','iter');
%startgættet ligger x= [S,omega]
x0=[86.2,0.956];
[m2, Pmin2, ~, output] = fminsearch( @ (x) pris(x(1),x(2),B,L),x0,optionB);
%det skal bruges 30 itteriationer til at opnå pmin2 som er det samme som
%pmin1=1.1214*10^3

%B4
%Ved at kigge på grafen ses at hmin ligger i x=0.
%Hvis det indsætte fås det hele til 0, derfor kan det ikke bruges ligesom
%i opgaveA. Dette skyldes at vi i opgave B nu kigger på et nyt
%koordinatsystem. I stedet kan h_min findes ud fra sammenhængen:
%h_min + h_max = B <=> h_min = B - h_max
```

```
%h_max beregnes på samme måde som før, men med S og omega sat til de
%værdier fundet for den billigste brokonstruktion.
%ved at kigge på m værdien, bestemmes S og omega og rundes op
S = 87;
omega = 0.96;
% B = 30; % samme som tidligere
% L = 80; % samme som tidligere
% startgæt
x0 = 30;
%Definering af funktion f(T)
f = @(T)  T/omega.*sinh((L.*omega)./(2.*T))-(1./2).*S;
%Definering af tolerance til fzero.
%starter med et gæt på tolerancen = 1e-3
optionB1=optimset('TolX',1e-3);
%fzero bruges til at bestemme T
T = fzero(f,x0,optionB1);
%Defination af x = L/2
x = L/2;
h_max = (T/omega)*(cosh(omega*x/T)-1);
h_min = B - h_max;
```

OPGAVE C – MATLAB KODE

```
clc; close all; clear all;
%C.2
%Start værdierne og ligningen
xn1=0.3;
x0=1.5;
fx=@(x) x^2-0.7*x-0.3;
%Funktionen benyttes på ligningen
x=secantroot(fx,xn1,x0,10^-14,15);
E1=abs(x-1);
%C.3
%Log funktionerne plottes og alpha(h) findes
h=1.6;
for i=1:length(E1)-1
    d(i)=E1(i);
    b(i)=E1(i+1);
end
figure()
plot(h*log(d),log(b)); hold on;
xlabel('log(Ei)')
ylabel('log(Ei+1)')
hold off;
%C.4
%Beregningen testes med ombyttet x-værdier
x2=secantroot(fx,x0,xn1,10^-14,15);
%fplot(fx,[-1,2])
E2=abs(x2-1);
```