Technical University of Denmark

3.Obligatoriske Aflevering

02601 – Introduktion til numeriske algoritmer

Forår 2015

Udearbejde af:	*Ansvarsområde*
Ran Wang – s111503	*Spørgsmål:1,2,A (English)*
Alireza Nedaei – s113681	*Resten af afleveringen:50%*
Mathias Thage Hansen – s113680	*Resten af afleveringen:50%*

1. OPTAG AF LUFTFORURENING

1.1

We see that the integral I from (1) can be estimated by the form described in (2), which gives a result:

$$I_{grov} = 11.3863$$

1.2

We are supposed to fit the function f(t) to the given data, which apparently we have to estimate several variables in the function, and it is a unconstrained nonlinear optimization problem. In order to do so, we start formulating the given function (3) with input t, y, and all the given initial values. The function returns quadratic error-residual that is minimized by "fminsearch" that estimates the best fitting variables corresponding to the initial values. The fitted data is plotted against the original in figure 1.

$$a1 = 0.9801$$
$$a2 = 0.3918$$
$$t1 = 1.5406$$
$$t2 = 9.3602$$
$$\sigma 1 = 4.4574$$
$$\sigma 2 = 5.3914$$
$$d1 = 2.3059$$
$$d2 = 1.6832$$

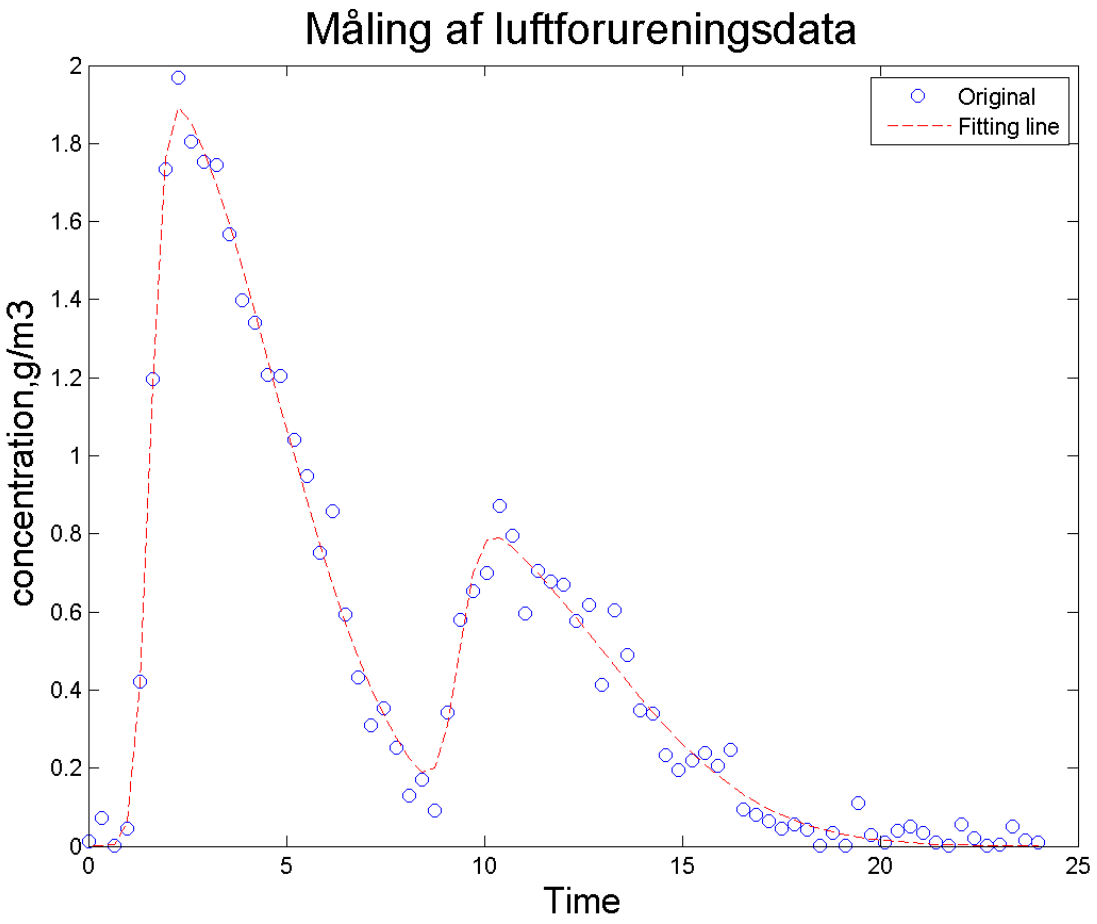


Figure 1 Måling af luftforureningsdata fit

1.3

We used Matlab function "quad" to find the area under the graph of the function, in other words, to compute the integral Numericall by using adaptive Simpson quadrature method, which yields:

$$I = 11.4837$$

2. KONDILØBEREN

2.1

We used Matlab function polyfit to find the coefficients at degree of 12, then we used another function polyfit to evaluate the coefficients, which yields the values of polynomial at time x. The fitted curve and position along with the measured data can be seen in figure 2.

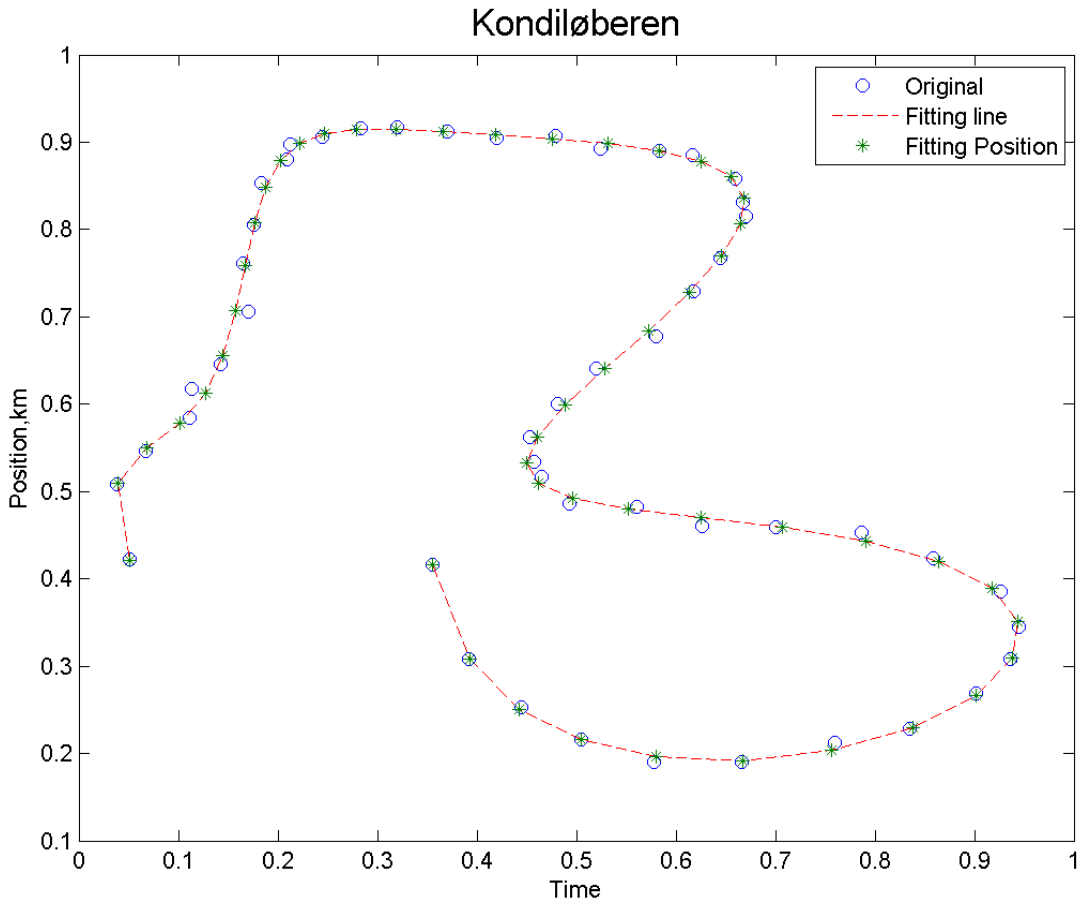


Figure 2 Kondiløberen fit

2.2

We used Matlab function polyder for computing derivatives $p_x'(t)$, $p_y'(t)$ of the two polynomials. It is noticed that the derivatives has one degree less than the original polynomials. in order to plot it with time t, we need to evaluate the derivatives using ployval. The derivative plot shows in figure 3.

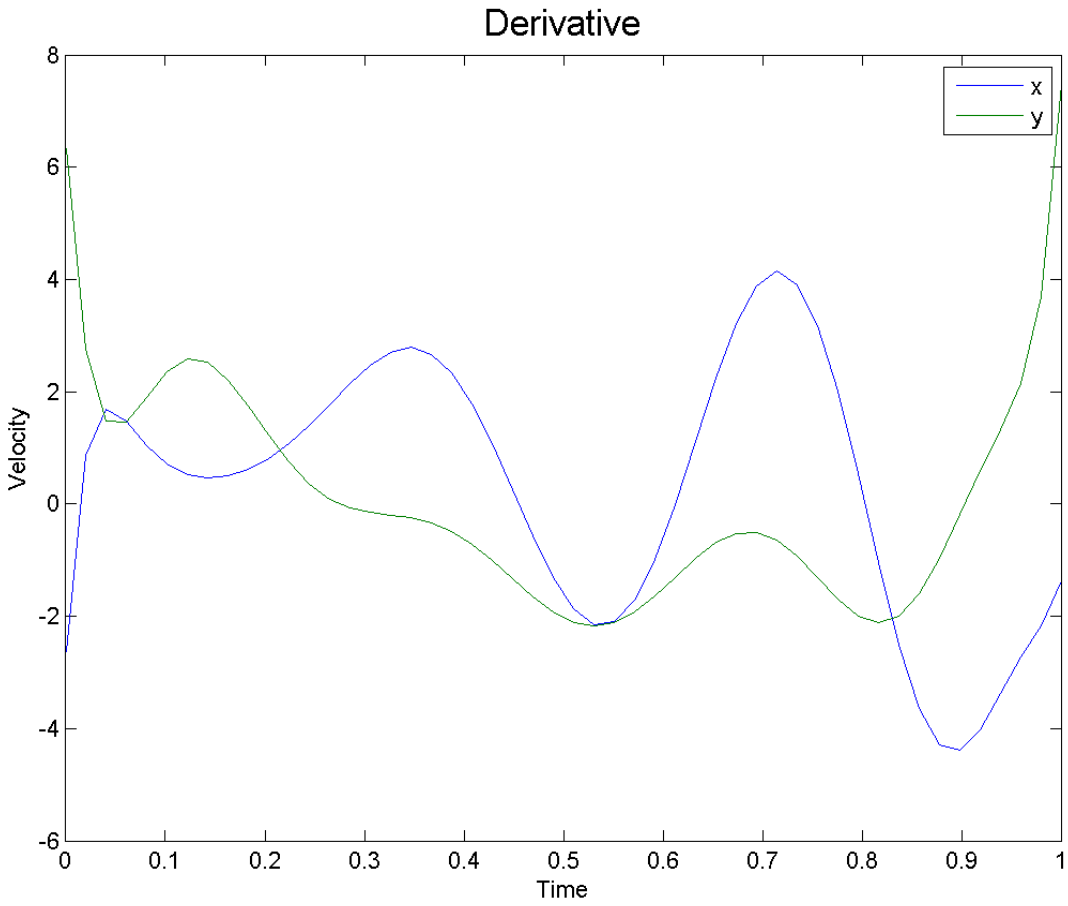


Figure 3 Derivative of polynomial

2.3

We are supposed to use Matlab function “quad” to calculate the distance function that is defined by D.

$$D = \int_0^T \sqrt{x'(t)^2 + y'(t)^2}\,dt$$

The integral includes lower bound 0 and upper bound T, and integrate with respect to t. the tolerance level is set to tol=1e-4.

$$D = 2.717$$

TEORISPØRGSMÅL 1. FEJLESTIMAT FOR TRAPEZ-METODEN

T1.1

Trapez-metoden er en metode som benytter to punkter på en kurve og derved er en ret linje som der bliver regnet på.

Til at beregne ligning 19.14 skal funktionen for linjen differenceres to gange hvilket gør at en førstegrads funktion giver 0 hvilket også gør at Et=0.

T1.2

G(x) funktionen beskriver forskellen på den oprindelige funktion f(x) og den funktionen som der er for trapez-metoden hvilket gør at formlen herunder beskriver sammenhængen.

$$g(x) = f(x) - trapez(x)$$

Funktionen for trapez-metode funktionen giver stadig 0 når den dobbelt differenceres så derfor skal f'' og g'' være lig med hinanden.

T1.3

Det beskrives at f''(x)=K og derfor må g''(x) også være lig med K og det integreres så to gange for at få f(x)

$$g''(x) = K$$

$$g'(x) = K * x + C_1$$

$$g(x) = \frac{1}{2} * K * x^2 + C_1 * x + C_2$$

Til at bestemme konstanterne C kan begyndelsesværdierne så benyttes så g(-r)=0 og g(r)=0 og løse den som to ligninger med to bekendte.

$$g(-r) = \frac{1}{2} * K * (-r)^2 + C_1 * (-r) + C_2 = 0 \; og \; g(r) = \frac{1}{2} * K * (r)^2 + C_1 * (r) + C_2 = 0$$

Udfra disse to ligninger findes:

$$C_1 = 0 \; og \; at \; C_2 = -\frac{1}{2} * K * r^2$$

Dette giver at g(x) er:

$$g(x) = \frac{1}{2} * K * x^2 - \frac{1}{2} * K * r^2 = \frac{1}{2} * K * (x + r) * (x - r)$$

T1.4

I denne opgave skal vises hvad der sker når g(x) bliver integreret.

$$\int_{-r}^{r} g(x)dx = \int_{-r}^{r} \frac{1}{2} * K * x^2 - \frac{1}{2} * K * r^2 \ dx$$

$$\int_{-r}^{r} \frac{1}{2} * K * x^2 - \frac{1}{2} * K * r^2 \ dx = \left[\frac{1}{6} * K * x^3 - \frac{1}{2} * K * r^2 * x\right]_{-r}^{r} \rightarrow$$

$$= -\frac{2}{3} * K * r^3$$

Så vides det at 2r = b-a

$$= -\frac{2}{3} * K * \left(\frac{b-a}{2}\right)^3 = -\frac{2}{3} * K * (b-a)^3 * \frac{1}{8} = -\frac{1}{12} * K * (b-a)^3$$

3.ODE METODER OG SKRIDTLÆNGE

3.1

I opgaven er begyndelsesværdiesproblem givet ved:

$$\frac{dy}{dt} = y * t^3 - 2 * y$$

Samt med begyndelsesværdie y(0)=1 over den givne interval t=[0,2], kan differentiel løses analytisk ved den given formel :

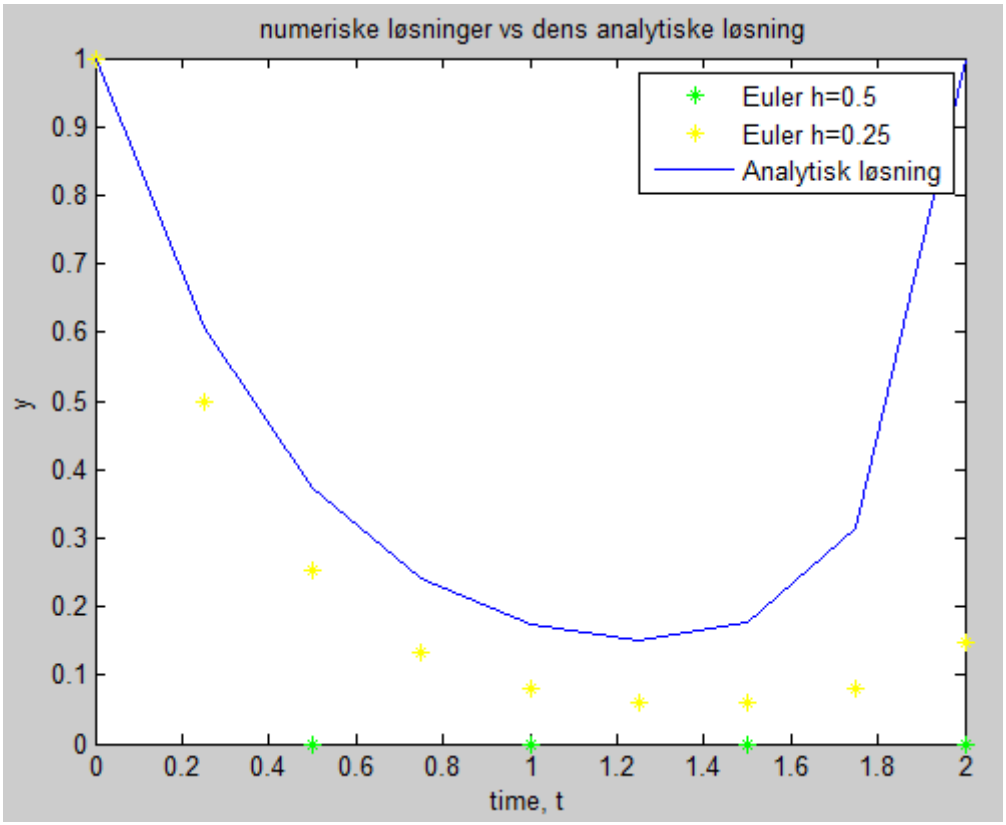
$$y(t) = y(0) * e^{0.25*t^4-2*t}$$

Ved at benytte Euler's numeriske metode bha. Den given koder i opgaven 'eulode.m', laves 2 tabeller for sammenligning af numerisk og analytiske beregninger i de givne skridtlængde, nemlig h=0.25 og h=0.5.

Sammenligning af løsninger (h=0.5)			
Tid	**Analytisk**	**Numerisk**	**Fejl**
t	**Ytrue**	**Yapprox**	**epsilon**
0	1	1	0
0.50	0.3737	0	100
1	0.1738	0	100
1.5	0.1765	0	100
2	1	0	**100**

Sammenligning af løsninger (h=0.25)			
Tid	**Analytisk**	**Numerisk**	**Fejl**
t	**Ytrue**	**Yapprox**	**epsilon**
0	1	1	0
0.2500	0.6071	0.5000	17.6444
0.5000	0.3737	0.2520	32.5738
0.7500	0.2415	0.1339	44.5748
1.0000	0.1738	0.0810	53.3635
1.2500	0.1511	0.0608	59.7806
1.5000	0.1765	0.0601	65.9684
1.7500	0.3150	0.0807	74.3729
2.000	1.000	0.1485	**85.1491**

Herunder ses der at relative fejl til tiden 2 for de to skridtlængde er 100% og 85.14%. Til sidste plottes de to numeriske løsninger samt dens analytisk.



Det kan konkluderes at den numeriske løsning i plottet bliver bedre, jo mindre skridtlængden bliver.

3.2

Ved at gøre præcis hvad der er blevet forklaret i opgaven , laves en ny fil som hedder heunode.m som kan ses på nedenstående figur.

```
% input:
%   dydt = name of the M-file that evaluates the ODE
%   tspan = [ti, tf] where ti and tf = initial and
%           final values of independent variable
%   y0 = initial value of dependent variable
%   h = step size
% output:
%   t = vector of independent variable
%   y = vector of solution for dependent variable

ti = tspan(1);
tf = tspan(2);
t = (ti:h:tf)';
n = length(t);
% if necessary, add an additional value of t
% so that range goes from t = ti to tf
if t(n)<tf
  t(n+1) = tf;
  n = n+1;
end
y = y0*ones(n,1); %preallocate y to improve efficiency
for i = 1:n-1  %implement Heun's method
    var = feval(dydt,t(i),y(i));
  yi0 = y(i) + var*(t(i+1)-t(i));
  y(i+1) = y(i) + (var+feval(dydt,t(i+1),yi0))*(t(i+1)-t(i))/2;
end
```

Den modificeret kode er afmærket med rød boks. Dvs linje de tilføjes linje i rød boks svarer til de to linje i opgave formel 9.

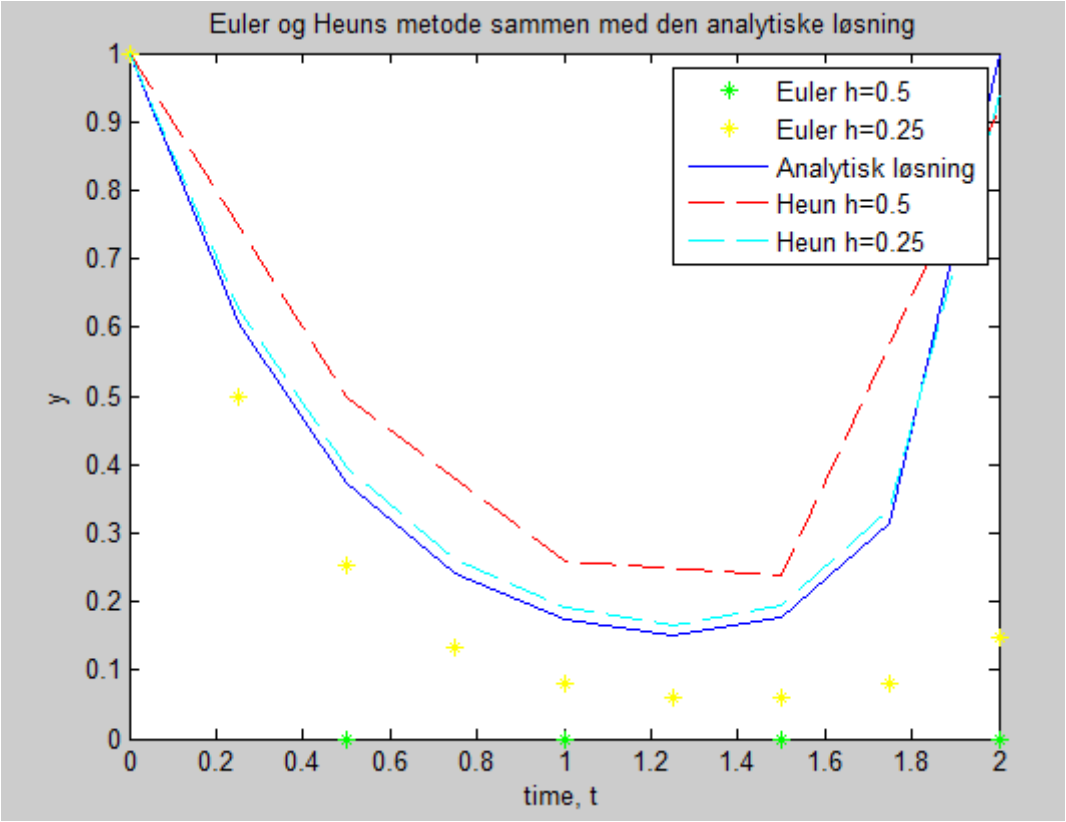
3.3

Ved hjælp af den nye heunode.m i matlab, benyttes Heun's numeriske metode. Derefter laves 2 tabeller ligesom opgave 3.1 der sammenligner numerisk og analytiske beregninger for skridtlængde h=0.25 og h=0.5. Det skal bemærkes at begge metoder, nemlig Euler's og Heuns's metode er inkluderet herunder.

Sammenligning af løsninger (h=0.5)					
Tid	**Analytisk**	**Numerisk(Eulers)**	**Fejl**	**Numerisk(Heuns)**	**Fejl**
t	**Ytrue**	**Yapprox**	**Epsilon1**	**Yapprox2**	**Epsilon2**
0	1.0000	1.0000	0	1.0000	0
0.5000	0.3737	0	100.0000	0.5000	33.8069
1.0000	0.1738	0	100.0000	0.2578	48.3609
1.5000	0.1765	0	100.0000	0.2377	34.6497
2.0000	1.0000	0	100.0000	0.9210	**7.9025**

Sammenligning af løsninger (h=0.5)					
Tid	**Analytisk**	**Numerisk(Eulers)**	**Fejl**	**Numerisk(Heuns)**	**Fejl**
t	**Ytrue**	**Yapprox**	**Epsilon1**	**Yapprox2**	**Epsilon2**
0	1.0000	1.0000	0	1.0000	0
0.2500	0.6071	0.5000	17.6444	0.6260	3.1053
0.5000	0.3737	0.2520	32.5738	0.3968	6.1826
0.7500	0.2415	0.1339	44.5748	0.2622	8.5728
1.0000	0.1738	0.0810	53.3635	0.1906	9.7016
1.2500	0.1511	0.0608	59.7806	0.1660	9.8204
1.5000	0.1765	0.0601	65.9684	0.1932	9.4465
1.7500	0.3150	0.0807	74.3729	0.3354	6.4847
2.0000	1.0000	0.1485	85.1491	0.9390	**6.0955**

Herunder kan det konstateres at relativ fejl for de to skridtlængder er 7.90% og 6.09%. Til sidst plottes de 4 numeriske løsninger samt analytisk.

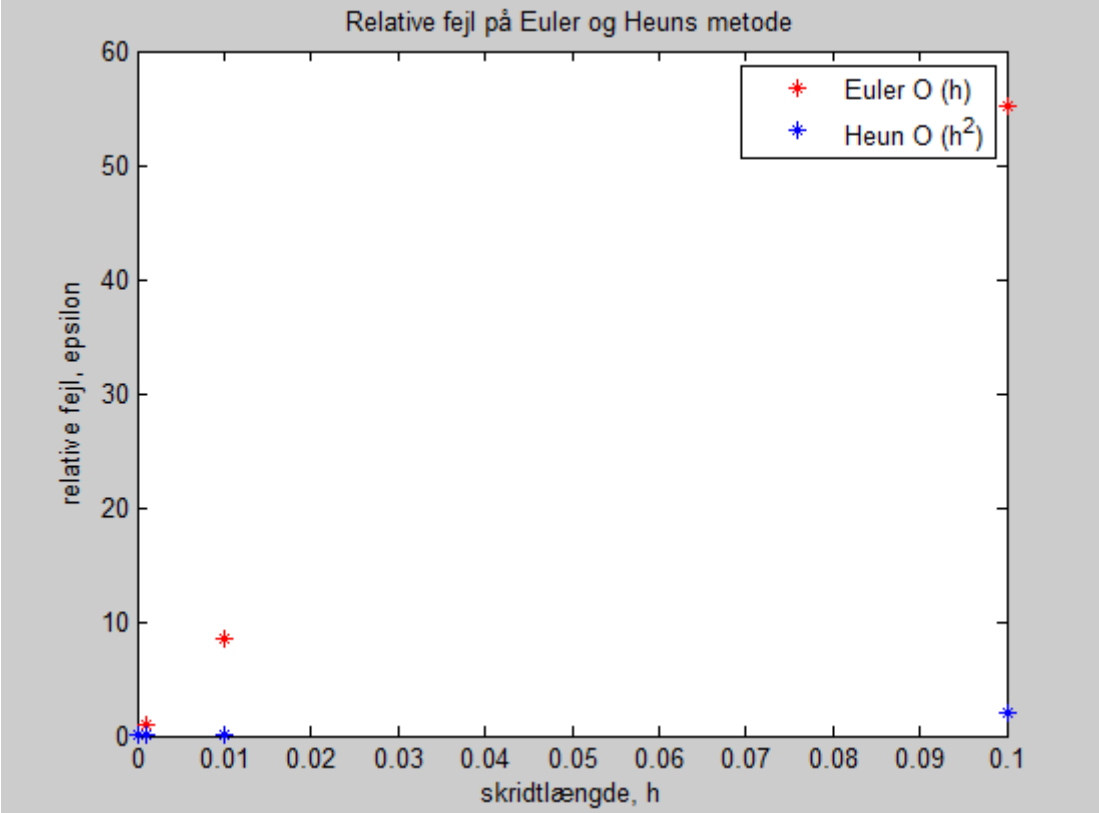


Ved at studere plottet, ses at den numeriske løsning er bedre for Heun's end Euler's metode.

3.4

Det opstilles et tabel for relativ fejl beregnet til tiden t=2 for små skridtlængder.

Relativ fejl for Eulers og Heuns		
Skridtlængde h	**Euler's metode (O(h))**	**Heun's metode (O(h^2))**
0.1	55.1263	2.0266
0.01	8.5872	0.0299
0.001	0.9084	0.003
0.0001	0.0914	0



Det konstateres hermed at Euler metoden er proportional mens Heun vokser kvadratisk.

4. EPIDEMI I EN BEFOLKNING

4.1

Systemet er givet ved

$$\frac{dy_1}{dt} = -cy_1y_2$$

$$\frac{dy_2}{dt} = cy_1y_2 - dy_2$$

$$\frac{dy_3}{dt} = -dy_2$$

Første skridt laves en funktion som navngives epidi:

```
function dy = epidi(t,y)
% UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
dy = zeros(3,1); % a column vector
dy(1) = -1*y(1)*y(2); % (10)
dy(2) = 1*y(1)*y(2)-5*y(2);
dy(3) = 5*y(2);
end
```

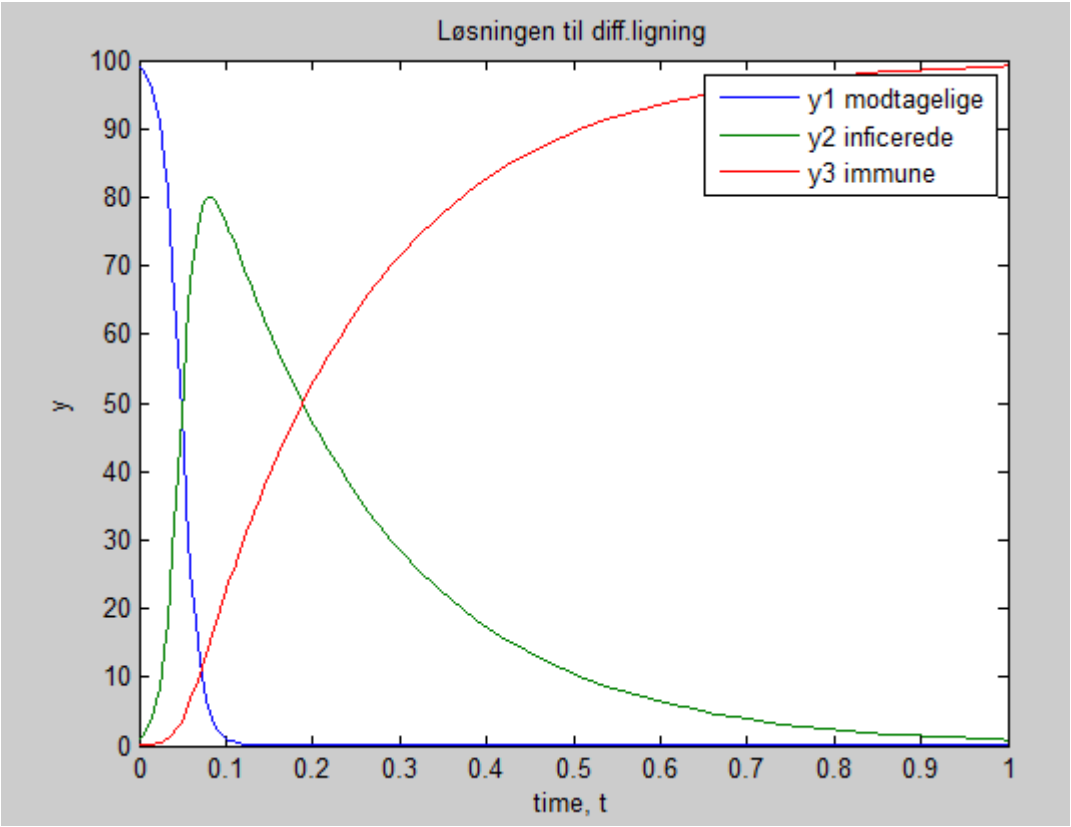
Derefter ved at brug ode45 indbygget løses differentialeligning. ODE løses med de givne tid i opgaver, [0 1] og begyndelser værdier i følgende:

$$y_1(0) = 99, \qquad y_2(0) = 1, \qquad y_3(0) = 0, \qquad c = 1, \qquad d = 5$$

Systemet i matlab kaldes ode45. Koden kan ses i bilag afsnit.

4.2

I opgaven er der 3 forskellige grupper, nemlig modtagelig, indificerede og immune. De plottes i samme plot:



Det ses at andelen af modtagelige falder, mens andelen af immune stiger. Dette bevirker at andelen af inficerede først stiger kraftigt, for derefter at aftage og gå mod 0. Det ses også at den maksimale procentdel inficerede er 80.07 % (Dette er bestemt numerisk ud fra vektoren.)

Plottet viser

- 2.delen af modtagelige falder
- 2.delen af immune stiger
- 2.delen af inficerede stiger kraftig og derefter aftager og gå mod 0

Den maksimale procentdel indificerede er også beregnede i matlab koden til 80.0731 %

4.3

I tidpunktet 0 og 1, er der lige så mange indificerede og immune personer, fra forrige opgave, konstateres også at ca ved t=0.2 og 50% de to funktioner skærer hinanden. Dette kan bruges som en nulpunkt problem hvor vi kan brug regne indificerede- imunne i fzero. Herefter findes en t som kan enten sættes i immune eller indificerede for at finde nulpunktet,

$$t = nulpunkt = 0.1881 \qquad y2 = y3 = 49.9978\%$$

TEORISPØRGSMÅL 2. OMSKRIVNING AF ANDEN-ORDENS LIGNINGER

T2.1

Ud fra ligning (13) skal der udledes en ligning for dy/dt og benyttes så vektorerne:

$$y = \begin{pmatrix} x \\ v \end{pmatrix} \; og \; \frac{dy}{dt} = \begin{pmatrix} dx/dt \\ dv/dt \end{pmatrix}$$

Så indsættes disse i ligningen (13) og derved fås at:

$$\frac{dy}{dt}(1) = \frac{dx}{dt} = v = y(2)$$

$$m * \frac{dv}{dt} + c * v + k * x = 0 \rightarrow m * \frac{dy}{dt}(2) + c * y(2) + k * y(1) = 0 \rightarrow \frac{dy}{dt}(2) = \frac{c * y(2) + k * y(1)}{m}$$

Dette er de to funktioner som beskriver dy/dt.

Funktionen odefun går bare ind og beregner vektoren dy/dt på sammen måde som gjort herover. Den får vektoren y og en tid t og så kan den så beregne vektoren dy/dt. Den definere også konstanterne m, c og k.

T2.2

Funtionen $\frac{d^2x}{dt^2} - (1 - x^2) * \frac{dx}{dt} + x = 0$ vil vi gerne opskrive på samme måde som i T1.2 og der benyttes de samme vektorer.

Til $\frac{d^2x}{dt^2}$ benyttes at $\frac{dx}{dt} = v$ så derfor er den lig med $\frac{dv}{dt}$

$$\frac{d^2x}{dt^2} - (1 - x^2) * \frac{dx}{dt} + x = 0 \rightarrow \frac{dv}{dt} - (1 - x^2) * \frac{dx}{dt} + x = 0 \rightarrow \frac{dy}{dt}(2) - (1 - y(1)^2 * y(2) + y(1) = 0$$

Dette giver at $\frac{dy}{dt}(1) = y(2) \; og \; at \; \frac{dy}{dt}(2) = (1 - y(1)^2) * y(2) - y(1)$

Så den funktion som så vil blive skrevet i matlab vil se ud på følgende måde.

```
1   function dydt = odefun(t,y)
2
3   dydt=[y(2);(1-y(1)^2)*y(2)-y(1)]
4   end
```

5. FINITE DIFFERENCE METODEN OG SKYDEMETODEN

5.1

Først skal konstanterne c,d og e bestemmes i ligningen $c * y_{i-1} + d * y_i + e * y_{i+1} + x_i = 0$ disse konstanter kan bestemmes ud fra ligningen:

$$7 * \frac{d^2x}{dx^2} + \frac{dy}{dx} - y + x = 0$$

$$\frac{d^2y}{dx^2} = \frac{y_{i+1} - 2 * y_i + y_{i-1}}{\Delta x^2} \ og \ \frac{dy}{dx} = \frac{y_{i+1} - y_{i-1}}{2\Delta x} \ og \ y = y_i \ og \ x = x_i$$

Så kan c,d og e bestemmes

$$7 * \frac{y_{i+1} - 2 * y_i + y_{i-1}}{\Delta x^2} + \frac{y_{i+1} - y_{i-1}}{2\Delta x} - y_i + x_i = 0 \rightarrow$$

$$7 * \frac{y_{i+1}}{\Delta x^2} - 7 * \frac{2 * y_i}{\Delta x^2} + 7 * \frac{y_{i-1}}{\Delta x^2} + \frac{y_{i+1}}{2\Delta x} - \frac{y_{i-1}}{2\Delta x} - y_i + x_i = 0 \rightarrow$$

$$7 * \frac{y_{i-1}}{\Delta x^2} - \frac{y_{i-1}}{2\Delta x} - 7 * \frac{2 * y_i}{\Delta x^2} - y_i + 7 * \frac{y_{i+1}}{\Delta x^2} + \frac{y_{i+1}}{2\Delta x} + x_i = 0 \rightarrow$$

Så kan y_i leddende sættes uden for en parentes og derved er c, d og e bestemt

$$\left(\frac{7}{\Delta x^2} - \frac{1}{2\Delta x}\right) y_{i-1} + \left(\frac{-14}{\Delta x^2} - 1\right) y_i + \left(\frac{7}{\Delta x^2} + \frac{1}{2\Delta x}\right) y_{i+1} + x_i = 0$$

Konstanterne er derfor følgende

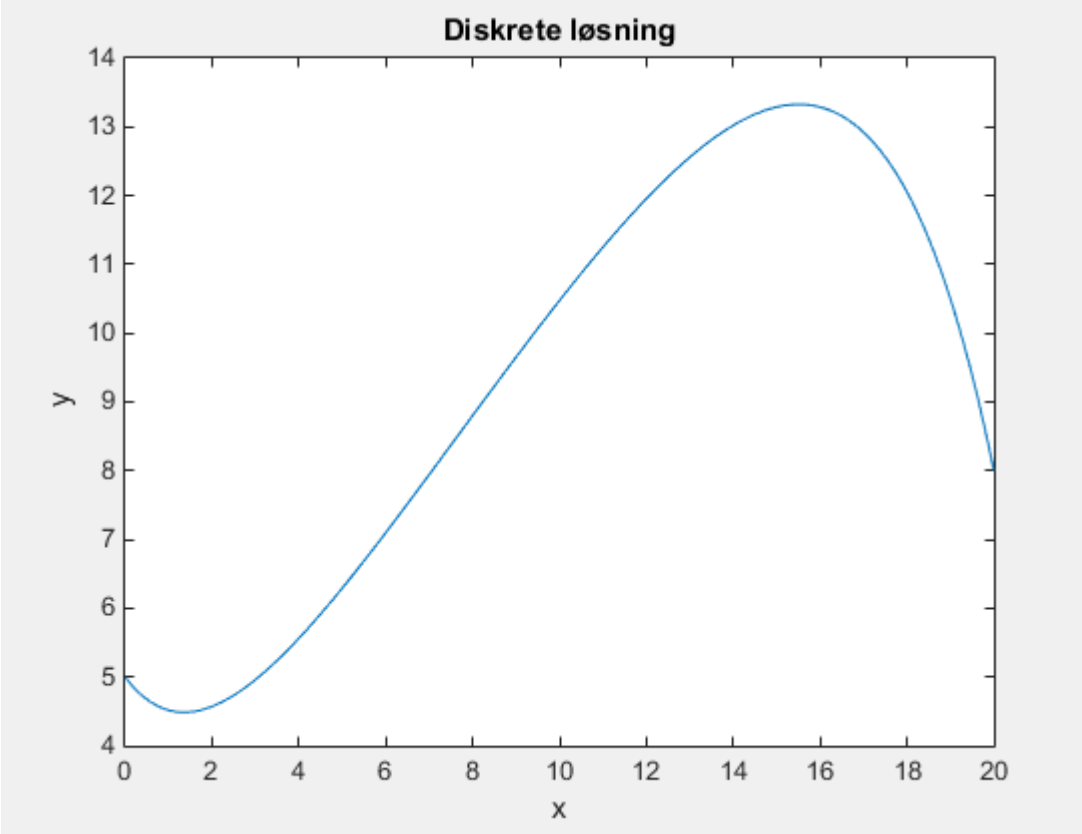
$$c = \left(\frac{7}{\Delta x^2} - \frac{1}{2\Delta x}\right) \ og \ d = \left(\frac{-14}{\Delta x^2} - 1\right) \ og \ e = \left(\frac{7}{\Delta x^2} + \frac{1}{2\Delta x}\right)$$

5.2

Den diskrete løsning findes ved at benytte backslash funktionen på de to matricer i formel (21).

$$y = A \backslash b$$

Herunder kan man så se et plot af den fundende løsning.



5.3

For at lave skyde metoden skal der opskrives en funktionen som beregner den afledet og den anden afledet.

```
1    function [ dydx ] = odefun( x,y )
2
3 -      dydx=[y(2);(-y(2)+y(1)-x)/7];
4 -  end
```

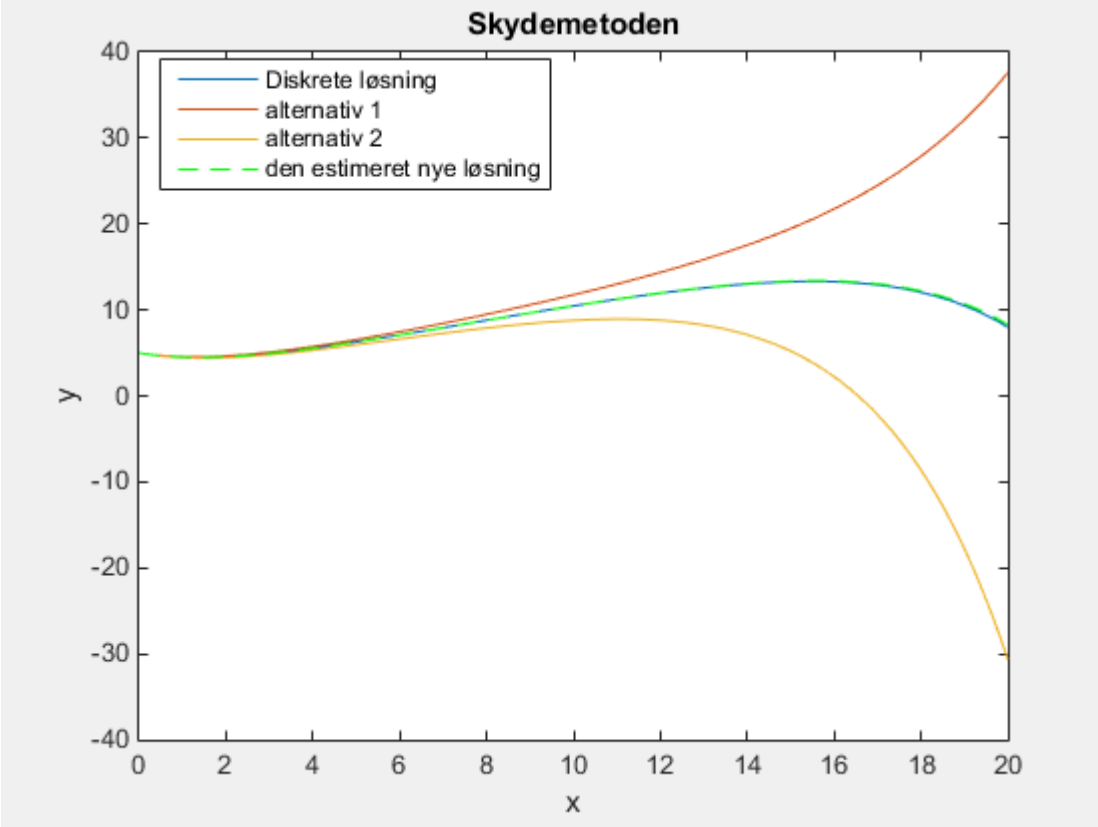
Herefter benyttes ode45 funktionen som så beregner funktions værdierne ud fra start gættene.

Da disse to gæt ikke rammer rigtigt i forhold til den diskrete løsning skal kan den rigtige hældning til problemet estimeres ved at lave følgende beregning.

$$z_a = z_{a1} + \frac{z_{a2} - z_{a1}}{T_{b2} - T_{b1}}(T_b - T_{b1})$$

Hvor z-værdierne er starthældningerne og T værdierne er den estimeret slutværdi.

Den hældning som denne beregning kommer frem til er så -0.8429 og det kan ses på plottet at den er den rigtige løsning.


Skydemetoden
Diskrete løsning
alternativ 1
alternativ 2
den estimeret nye løsning
y
x
40
30
20
10
0
-10
-20
-30
-40
0
2
4
6
8
10
12
14
16
18
20

OPGAVE A. SLINKY DER HOLDES FAST I DEN ENE ENDE

A.1

Firstly, We calculate the derivatives of [22] and [23], and then I plug in all the given values to the derivatives, so that I get the stretched lengths of the slinky under both conditions.

$$y_{uden}(l) = l$$
$$y_{uden}'(l) = 1$$

$$y_{med}(l) = l + a \cdot g \cdot \left(\frac{2l}{L_0} - \frac{l^2}{L_0^2}\right)$$
$$y_{med}'(l) = 1 + a \cdot g \cdot \left(\frac{2}{L_0} - \frac{2l}{L_0^2}\right)$$

$$y_{uden}'(0) - 1 = 1 - 1 = 0cm$$
$$y_{med}'(0) - 1 = 1 + \frac{1}{9.82} \cdot 9.82 \cdot \left(\frac{2}{0.1} - \frac{2 \cdot 0}{0.1^2}\right) - 1 = 20cm$$

A.2

We make two functions describing the final energy with and without gravity, which the slinky remains when the potential energy from the slinky minus the potential energy from gravity. Once the functions are formulated, I use Matlab function quad to compute the integrals.

$$I_{uden} = \int_0^{L_0} \left[-\frac{1}{L_0} M \cdot g \cdot y_{uden}(l) + \frac{1}{2} \cdot k \cdot L_0 \cdot (y_{uden}'(l) - 1)^2\right] dl$$
$$\rightarrow I_{uden} = \int_0^{0.1m} \left[-\frac{1}{0.1m} 0.2kg \cdot 9.82\, m/s^2 \cdot l + \frac{1}{2} \cdot 1\, kg/s \cdot 0.1m \cdot (1-1)^2\right] dl = \underline{-0.093N}$$
$$\dots$$
$$I_{med} = \int_0^{L_0} \left[-\frac{1}{L_0} M \cdot g \cdot y_{med}(l) + \frac{1}{2} \cdot k \cdot L_0 \cdot (y_{med}'(l) - 1)^2\right] dl$$
$$\rightarrow I_{med} = \int_0^{0.1m} \left[\begin{array}{l} -\frac{1}{0.1m} 0.2kg \cdot 9.82\, m/s^2 \cdot l + \frac{1}{9.82\, m/s^2} \cdot 9.82\, m/s^2 \cdot \left(\frac{2l}{0.1m} - \frac{l^2}{0.1m^2}\right) \\ +\frac{1}{2} \cdot 1\, kg/s \cdot 0.1m \cdot \left(1 + \frac{1}{9.82\, m/s^2} \cdot 9.82\, m/s^2 \cdot \left(\frac{2}{0.1m} - \frac{2l}{0.1m^2}\right) - 1\right)^2 \end{array}\right] dl = \underline{-0.7409N}$$

A.3

We use Matlab function *fminbnd* to find the minimum a, which is a variable in the integral I_{med}. The function *fminbnd* retures a value that is exactly the same as M/2k, which is very close to 1/g. Therefore, I_{med} with the estimated *a* is also quite close to the I_{med} calculated in A.2.

$$a_{opti} = 0.10$$

$$a = \frac{M}{2k} = \frac{0.2}{2 \cdot 1} = 0.10$$

$$I_{med,opti} = -0.741N$$

A.4

Firstly, we formulated a function Y(I,s) according to the given information. Secondly, in order to find the s, the condition I(s) = I_{ustr} must be satisfied. We use Matlab function *fzero* to find s under the condition that I(s) – I_{ustr}=0, which gives:

$$s = 0.0681$$

Finally, we plug s and other values into L_{max}, so I get:

$$L_{max} = s\left[L_0 + a \cdot g \cdot \left(\frac{2L_0}{L_0} - \frac{{L_0}^2}{{L_0}^2}\right)\right] = \underline{0.073m}$$

B. SKYDETRÆNING

B.1

$$\frac{d^2x}{dt^2} = -\frac{C_d A\rho}{2m}\left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right)^{\frac{1}{2}}\frac{dx}{dt} \quad \rightarrow \quad \frac{dx}{dt} = v_x \quad \rightarrow \frac{dv_x}{dt} = -\frac{C_d A\rho}{2m}\left((v_x)^2 + \left(v_y\right)^2\right)^{\frac{1}{2}} v_x$$

$$\frac{d^2y}{dt^2} = -\frac{g}{m} - \frac{C_d A\rho}{2m}\left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right)^{\frac{1}{2}}\frac{dy}{dt} \quad \rightarrow \quad \frac{dy}{dt} = v_y \rightarrow \quad \frac{dv_y}{dt} = -\frac{g}{m} - \frac{C_d A\rho}{2m}\left((v_x)^2 + \left(v_y\right)^2\right)^{\frac{1}{2}} v_y$$

B.2

$$z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} \rightarrow z_3 = \frac{dx}{dt}\ og\ z_4 = \frac{dy}{dt} \text{ -> } z = \begin{pmatrix} x \\ y \\ \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix}$$

Den afledede dzdt kan derved defineres som en vektor med 4 variable ved:

$$dzdt = \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{d^2x}{dt^2} \\ \frac{d^2y}{dt^2} \end{pmatrix}$$

Matlab funktion er I følgende:

```
function dzdt = odefunb( t,z )
g = 9.81;
% Masse  [kg]
m = 0.55;
% Luftens massetæthed [kg/m^3]
rho = 1.2041;
% Drag coefficient
C_d = 0.5;
% Diameter [m]
d = 0.07;
% Tværsnitsareal  [m^2]
A = pi*d^2/4;
% Output:
dzdt = [ z(3) ; z(4) ; -C_d*A*rho/(2*m)*((z(3))^2 + (z(4))^2)^(1/2)*z(3) ;
-g/m-C_d*A*rho/(2*m)*((z(3))^2 + (z(4))^2)^(1/2)*z(4) ];
end
```

B.3

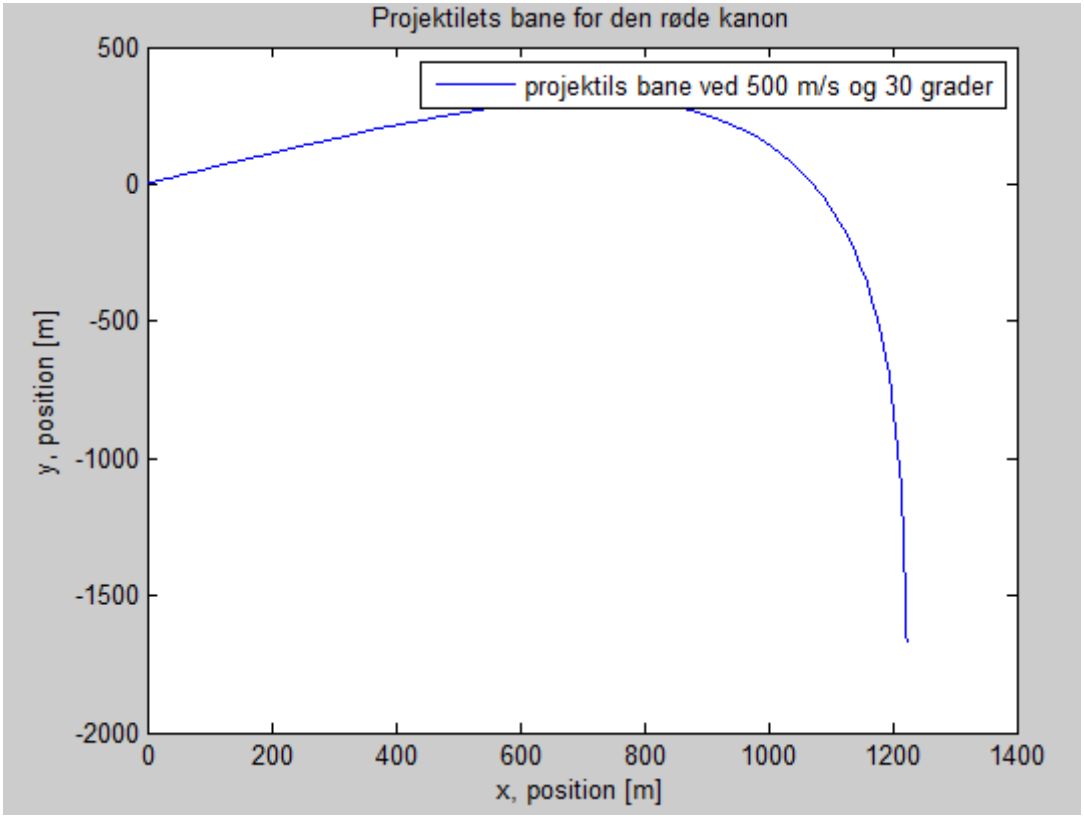
De følgende er punkter er indgivet i ode45 indbygget funktion

- Vektoren $[0\ 30]$ -> angiver at der skal simuleres i tidsrummet fra 0 til 30 sek.
- Vektoren [$0\ 2\ cos(pi/180 * 30) * 500\ sin(pi/180 * 30) * 500$] -> angiver start betingelserne. Startposition x_0 = 0 og $y_0 = 2$ Starthastighed i x og y
- retning -> $dx/dt = cos(pi/180 * 30) * 500\ og\ dy/dt = sin(pi/180 * 30) * 500$

```
[T,Y] = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*30)*500 sin(pi/180*30)*500 ]);
```

Starthastighed i x- og y-retning er fundet vha. simpel trekantsberegning, som værende hhv. cosinus til 30 grader gange de 500 m/s og sinus til 30 gange de 500 m/s.

Til sidste vises banen i nedstående figur:



B.4

Ved at bruge deval, fås adgang til løsningen y og x som er funktion af t. Vi kan hermed udnytte y=0 og beregner hvornår projektilet slår ned. Derefter vha fzero finder vi nulpunktesproblem. Den fundne tid bruges til at sættes i x(t). Matlab filen kan ses i Bilag B4 (den funden tid). Den funden tid og dens x koordinat er dermed beregnet til:

$$t = 11.2591\ [sek] \qquad x = 1.071885e + 03[m] \qquad y = 0\ [m]$$

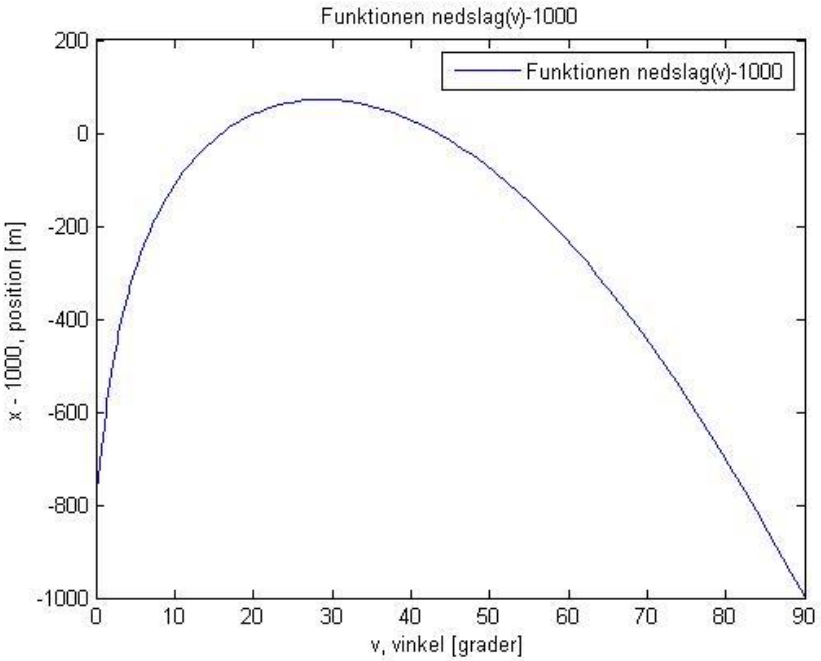
For at finde nøjagtigheden af løsning, og ændrer tolerancen på de numeriske beregninger i løsning af differentialeligningen, sættes RelTol i ode45. Matlab koden er i Bilag afsnit B4. Det skal dog bemærkes at kun den første kode for tolerancen er i bilag og derved gentages beregninger ved nye tolerancer. Det opstilles til sidst en tabel der viser hvordan løsningen afhænger af tolerancen.

RelTol	**1e-1**	**1e-2**	**1e-3(standard)**	**1e-4**	**1e-5**
X-kordinat[m]	1.0823e+03	1.07316e+03	1.07188e+03	1.07177e+03	1.07176e+03

Det kan konstateres fra overstående tabel at standardopløsning er passende når der er ønsket i opgaven at ramme $\pm 0.5\ [m]$

B.5

I første skridt laves en funktion som kaldes nedslag som er x koordinat, funktion af vinkel v. Funcktionen kan ses i MATLAB BILAG B5. Da vi ved at x=1000 m så skal nedslag(v)-1000=0. Denne ligning plottes for at finde et brugbare startgæt.



Udefra grafen, vælges hermed to startgæt på 12 og 38.

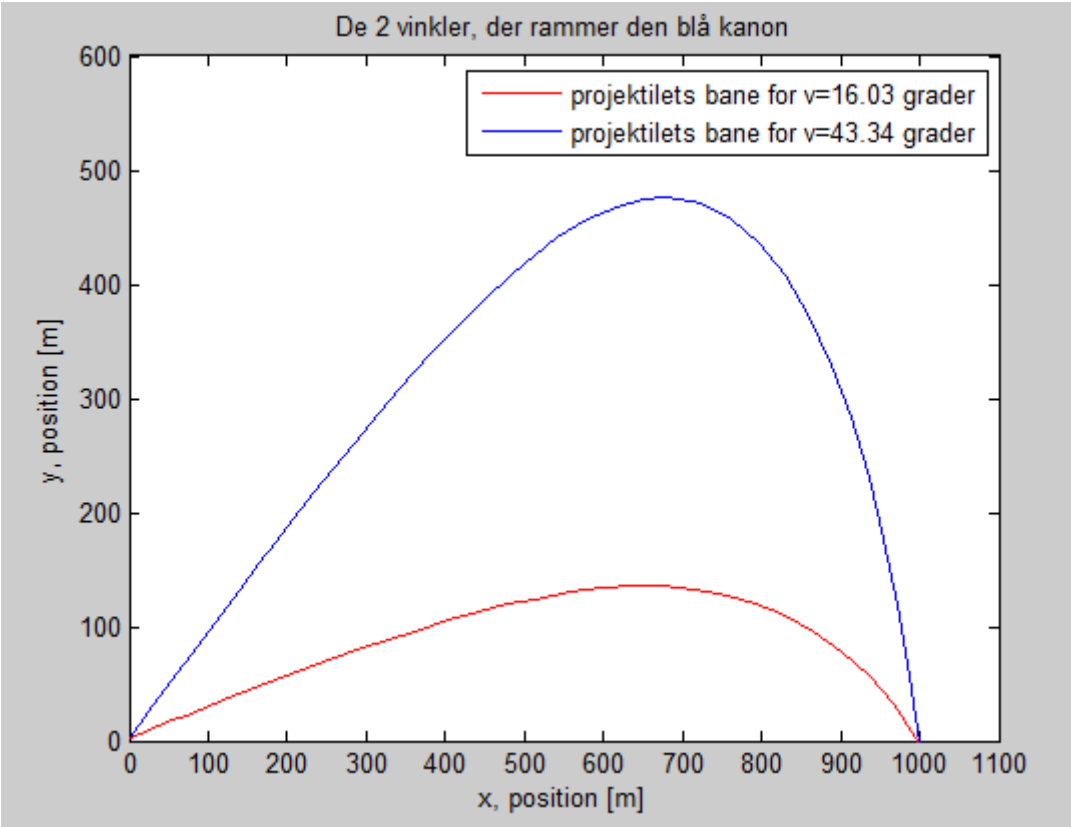
Løses hermed med fzero og mine startgæt:

```
v1 = fzero(@(v) nedslag(v)-1000,12)
v2 = fzero(@(v) nedslag(v)-1000,38)
```

De to vinkler findes hermed til:

$$v1 = 16.03 \ \ grader \qquad v2 = 43.34 \ grader$$

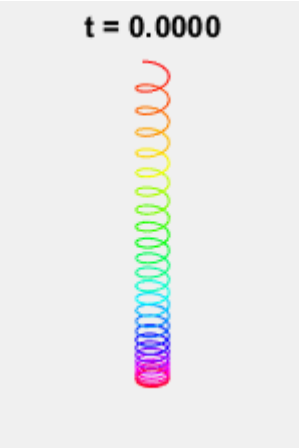
For at tjekke om vinkerne rigtige, indsættes disse og konstateres at løsningen er rigtig da den rammer x=1000 inden for $\pm 0.5\ [m]$. To løsninger plottes hermed :



C. DEN FALDENDE SLINKY

C.1

Til at tegne slinky'en skal y og l bestemmes. Y er bestemt ud fra formel (23) og startværdierne i opgave a og l er bestemt ud fra (28) og startværdierne i opgave a. Det giver følgende plot.



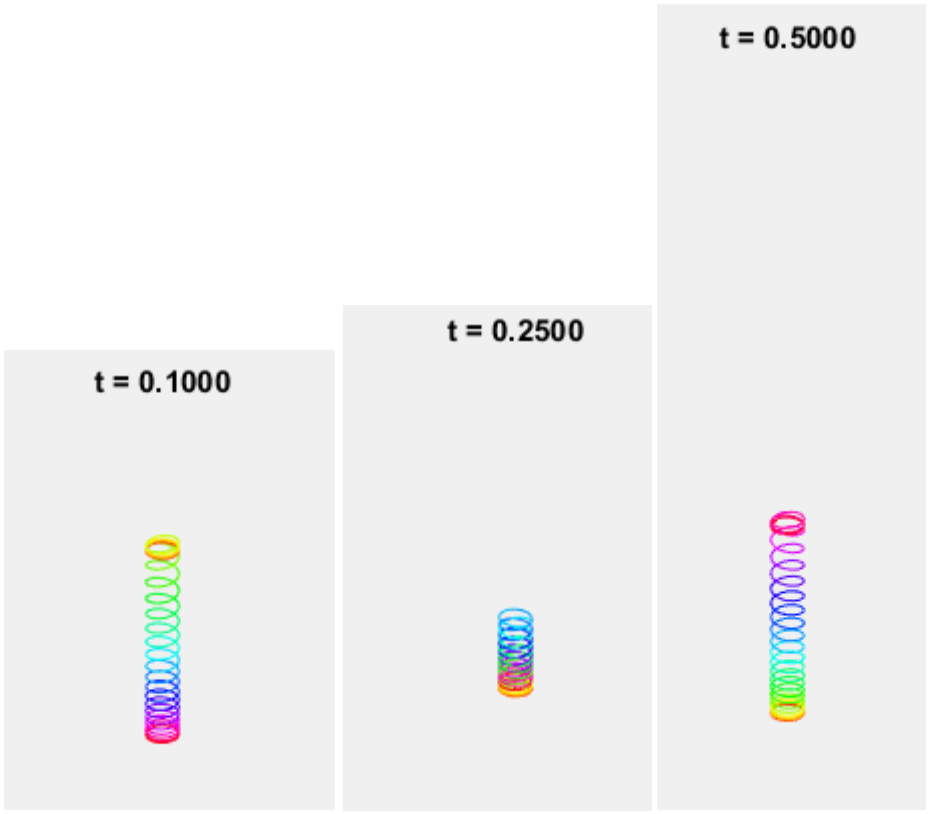
C.2

Til at simulere hvordan slinky'en falder benyttes ode45 funktionen men for at lave beregningen opskrives den funktion som giver outputtet den afledet og den anden afledet.

```
1     function [ dzdt] = odeC2(t,z0,l,dl,M,k,L_o,g)
2
3 -     y1=z0(1:400);
4 -     y2=z0(401:end);
5 -     dzdt=[y2;ffun(l,y1,dl,M,k,L_o,g)];
6 -     end
7
```

Da ligningen for y ikke har t i sig giver det derfor at den afledet af den er 0 og den anden afledet findes ved at bruge ffun funktionen.

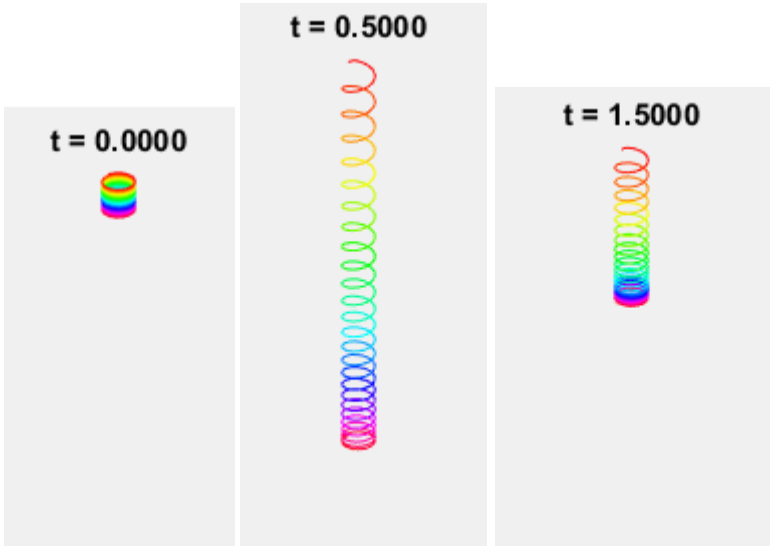
Derefter kan slinkyen tegnes.



C.3

I denne opgave skal der simuleres hvis slinkyen starter I ustrakt længde og derefter bliver sluppen men stadig holdes fast I toppen.

Dette beregnes på samme måde som i opgave C2 de eneste forskelle er at y(0)=l og der bruges true i ffun funktionen for at holde fast i toppen.



APPENDIX – MATLAB KODE

OPGAVE 1

```
load('airpollutiondata.mat')
```

```
%% 1.1
t = 24;
n = 75;
delta = t/n
I = delta* sum(y)
```

```
%% 1.2
load('airpollutiondata.mat')
a1=1;a2=1;t1=2;t2=8;d1=1;d2=1;sigma1=4;sigma2=4;
p=[a1,a2,t1,t2,sigma1,sigma2,d1,d2];
p = fminsearch(@(p)fitfun3(t,y,p),p)

a1=p(1);a2=p(2);t1=p(3);t2=p(4);sigma1=p(5);sigma2=p(6);d1=p(7);d2=p(8);

f= a1.*exp(-((t-t1)/sigma1).^2).*(1+erf(d1.*(t-t1)))+...
   (a2.*exp(-((t-t2)./sigma2).^2).*(1+erf(d2.*(t-t2))));

plot(t,y,'o',t,f,'r--');
xlabel('Time','FontSize',16);
ylabel('concentration,g/m3','FontSize',16);
legend('Original','Fitting line','FontSize',16);
title('Måling af luftforureningsdata','FontSize',20);
saveas(gcf,'fitting.png')
```

```
%% 1.3

f=@(t) a1.*exp(-((t-t1)./sigma1).^2).*(1+erf(d1.*(t-t1)))+...
   (a2.*exp(-((t-t2)./sigma2).^2).*(1+erf(d2.*(t-t2))));

I=quad(f,0,24) %% adaptive Simpson quadrature
```

OPGAVE 2

```
%% 2.1
load('kondi.mat')
n=12;
px=polyfit(t,x,n);
py=polyfit(t,y,n);
fx = polyval(px,t);
fy = polyval(py,t);
figure(1)
plot(x,y,'o',fx,fy,'r--',fx,fy,'*')
xlabel('Time');
ylabel('Position,km');
legend('Original','Fitting line','Fitting Position');
title('Kondiløberen','FontSize',16);
saveas(gcf,'Kondiløberen.png')
% axis([0  1  0  1])
```

```
%% 2.2
dx = polyder(px)
vdx=polyval(dx,t);
dy = polyder(py)
vdy=polyval(dy,t);
figure(2)
plot(t,vdx,t,vdy)
xlabel('Time');
ylabel('Velocity');
legend('x','y');
title('Derivative','FontSize',16);
saveas(gcf,'Derivative.png')
```

```
%% 2.3
f =@(t) sqrt(polyval(dx,t).^2 + polyval(dy,t).^2);
tol=1e-4;
Q = quad(f,0,t(end),tol)
```

OPGAVE 3.1

```
%% 3.1)
% Differentialligning:
dydt = @(t,y) y*t^3 -2*y; % (6)
% Interval: t1, t2
tspan = [0, 2];
% Begyndelsesværdi: y(0)=1
y0 = 1;
% Skridtlængde: h1, (grov)
h1 = 0.5;
% Skridtlængde: h2, (fin)
h2 = 0.25;
% Kalder m.fil med Euler's metode (grove værdier)
[t1,y1] = eulode(dydt, tspan, y0, h1);
% Kalder m.fil med Euler's metode (fine værdier)
[t2,y2] = eulode(dydt, tspan, y0, h2);
% Analytisk løsning til differentialligningen (6)
y = @(t) y0*exp(0.25*t.^4-2*t);
y1a = y(t1);
y2a = y(t2);
% Procentvis relative fejl:
epsilon1 = zeros(length(y1a),1);
for i = 1:length(y1a)
epsilon1(i) = abs((y1a(i)-y1(i))/y1a(i))*100;
epsilon1(i) = abs((y1a(i)-y1(i))/y1a(i))*100;
end
epsilon2 = zeros(length(y2a),1);
for j = 1:length(y2a)
epsilon2(j) = abs((y2a(j)-y2(j))/y2a(j))*100;
end
% Tabel af numeriske løsning (grov) (t1,y1) viser tiden og den approksimerede
% værdi. Sammenholdt med den analytiske løsning.
disp([t1,y1a,y1,epsilon1])
% Tabel af numeriske løsning (fin) (t2,y2) viser tiden og den approksimerede
% værdi. Sammenholdt med den analytiske løsning.
disp([t2,y2a,y2,epsilon2])
% Det ses ud fra de 2 tabeller, at den relative fejl er hhv. 100% og 85.15%
% til tiden t=2.
% Plotter de 3 grafer sammen:
figure(1)
plot(t1,y1,'*g',t2,y2,'*y',t2,y2a,'-b');
legend('Euler h=0.5','Euler h=0.25','Analytisk løsning');
title(' numeriske løsninger vs dens analytiske løsning')
xlabel('time, t')
ylabel('y')
```

OPGAVE 3.3

```
% Genbruger koden fra opgave 3.1) og kopierer det sidste ind, da det er
% fuldstændig samme fremgangsmåde.
% Kalder m.fil med Heun's metode (grove værdier)
[t1h,y1h] = heunode(dydt, tspan, y0, h1);
% Kalder m.fil med Heun's metode (fine værdier)
[t2h,y2h] = heunode(dydt, tspan, y0, h2);
% Procentvis relative fejl:
epsilon1h = zeros(length(y1a),1);
for i = 1:length(y1a)
epsilon1h(i) = abs((y1a(i)-y1h(i))/y1a(i))*100;
end
epsilon2h = zeros(length(y2a),1);
for j = 1:length(y2a)
epsilon2h(j) = abs((y2a(j)-y2h(j))/y2a(j))*100;
end
% Tabel af numeriske løsning (grov) (t1,y1) viser tiden og den approksimerede
% værdi. Sammenholdt med den analytiske løsning. Nu også med Heun's metode.
disp([t1,y1a,y1,epsilon1,y1h,epsilon1h])
% Tabel af numeriske løsning (fin) (t2,y2) viser tiden og den approksimerede
% værdi. Sammenholdt med den analytiske løsning. Nu også med Heun's metode.
disp([t2,y2a,y2,epsilon2,y2h,epsilon2h])
```

OPGAVE 3.4

```
%3.4
for i = 3:6
    hi = 0.1*10^(3-i);

    [ti,yi] = eulode(dydt, tspan, y0, hi);

    [tih,yih] = heunode(dydt, tspan, y0, hi);
    yia = y(ti);

    epsiloni = zeros(length(yia),1);
    epsilonih = zeros(length(yia),1);
    for j = 1:length(yia)
        epsiloni(j) = abs((yia(j)-yi(j))/yia(j))*100;
        epsilonih(j) = abs((yia(j)-yih(j))/yia(j))*100;
    end
    euler(i-2)= epsiloni(length(epsiloni));
    heun(i-2) = epsilonih(length(epsilonih));
end
euler
heun
h = [0.1 0.01 0.001 0.0001];
figure(3)
plot(h,euler,'*r',h,heun,'*b');
```

OPGAVE 4.1

```
clear all; close all; clc
%4.1
%først defineres de begyndelse værdier , de tre første er kondition vektor
%og tiden er give [0 1]
 %y1(0) = 99;
 %y2(0) = 1;
 %y3(0) = 0;
 %c = 1;
 %d = 5;
 [T,Y] = ode45(@epidi,[0 1],[99 1 0]);
```

OPGAVE 4.2

```
 %4.2
figure(1)
plot(T,Y(:,1),'-',T,Y(:,2),'-',T,Y(:,3),'-')
legend('y1 modtagelige','y2 inficerede','y3 immune');
title('Løsningen til diff.ligning')
xlabel('time, t')
ylabel('y')
% Angiver hvor stor en procentdel der er inficerede, når denne procentdel
% er maksimal:
maks_inf = max(Y(:,2))
```

OPGAVE 4.3

```
% 4.3)
% Finder det tidspunkt tau_ 0 mellem t=0 og t=1 hvor der er lige så mange
% inficerede personer som der er immune personer.
sol = ode45(@epidi,[0 1],[99 1 0]);
% Det kan ses rent grafisk at vi ønsker at finder intersection mellem y2 og
% y3. Svaret er ca. t=0.2 og 50% Laver en ny vektor, der er
% mere præcis i dette interval.
x = linspace(0,1,10^5);
y = deval(sol,x);
figure(2)
plot(x,y)
% Som man kan se er dette lidt mere præcist.
maks_inf_pre = max(y(2,:)) %
```

```
% Finder nulpunkt i matlab:
% Inficerede:
y2 = @(t) deval(sol,t,2);
% Immune:
y3 = @(t) deval(sol,t,3);
% Nulpunkt via fzero:
nulp = fzero(@(t) y2(t)-y3(t),[0.1 0.3])
% Procentsatsen er altså:
y2(nulp)
y3(nulp)
```

OPGAVE 5

```
%% Opgave 5.1
dx=0.1;
%De fundende konstanter c,d og e opskrives.
c=7/dx^2-1/(2*dx);
d=-14/dx^2-1;
e=7/dx^2+1/(2*dx);

%% Opgave 5.2
N=20/dx+1;
%Matricerne opskrives
A=full(gallery('tridiag',N-2,c,d,e));
x=(0:dx:20)';
b=-x(2:N-1);b(1)=b(1)-c*5;b(end)=b(end)-e*8;
%Den diskrete løsning findes
y=A\b;
figure();
plot(x,[5;y;8]);
title('Diskrete løsning')
xlabel('x')
ylabel('y')

%% Opgave 5.3
%Startgættene
start1=[5 -0.8];
start2=[5 -0.9];
tspan=linspace(0,20,200);
%beregningen af de to funktioner ud fra startgættene
[x1 z1]=ode45(@odefun,tspan,start1);
[x2 z2]=ode45(@odefun,tspan,start2);
%den rigtige hælding bestemmes
z=start1(2)+(start2(2)-start1(2))/(z2(end,1)-z1(end,1))*(y(end)-z1(end,1));
%den nye funktionen bestemmes
nyt=[5 z];
[x3 z3]=ode45(@odefun,tspan,nyt);
figure();
plot(x,[5;y;8],x1,z1(:,1),x2,z2(:,1),x3,z3(:,1),'g--');
legend('Diskrete løsning','alternativ 1','alternativ 2','den estimeret nye løsning')
title('Skydemetoden');
xlabel('x')
ylabel('y')
```

```
function [ dydx ] = odefun( x,y )
dydx=[y(2);(-y(2)+y(1)-x)/7];
end
```

OPGAVE A – MATLAB KODE

```
%% sporgsmaal A
clear all; close all; clc;

%% A.2
M=0.2; k=1; L0=0.1;D=0.1;vik=25;g=9.82;
p=[M k L0 D vik g];
%opskriver funktionerne for potentiel energi med og uden tyngdekraft
%finder integralet af de to med quad
% y_ud = l;
dy_ud= 1;
Iuden =@(l)-1/L0*M*g*l+(1/2)*k*L0*(dy_ud-1).^2;
Iu=quad(Iuden,0,L0)
Im=quad(@(l)Imed(l,1/g),0,L0)
%% A.3
%find optimal a.
aopt=fminbnd( @(a) quad(@(l)Imed(l,a),0,L0),0,1)
%finder værdien af integralet med et nye a
Imed_opt=quad(@(l)Imed(l,aopt),0,L0)
%% A.4
I_ustr=Iu;
s=fzero(@(s) I_ustr-quad(@(l)ipot_s(l,s),0,L0),0,1)
%koerer nu med det fundne s
Lmax= s*(L0+aopt*g*((2*L0)/L0 - L0^2/L0^2))
```

B.4

Den funden tid:

```
% kaldet sol.
sol = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*30)*500 sin(pi/180*30)*500 ]);
% Finder nulpunkt i matlab
y = @(t) deval(sol,t,2);
% Nulpunkt vhp  fzero: [sek]
t_0 = fzero(@(t) y(t),[0 30])
% Projektil, x position (t):
x = @(t) deval(sol,t,1);
% indsættelse af den fundne tid [m]
x_value = x(t_0)
```

```
% Varierer nu nøjagtigheden af løsningen vha. "RelTol", så jeg kan ramme
% nedslagets x koordinat indenfor plus minus 0.5 m.
format 'long'
% Sætter tolerancen til min numeriske løsning af differentialligningen:
options1 = odeset('RelTol',1e-1);
sol1 = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*30)*500 sin(pi/180*30)*500 ], options1 );
y1 = @(t) deval(sol1,t,2);
t1 = fzero(@(t) y1(t),[0 30])
x1 = @(t) deval(sol1,t,1);
x_value1 = x(t1)
options2 = odeset('RelTol',1e-2);
sol2 = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*30)*500 sin(pi/180*30)*500 ], options2 );
```

B.5

Nedslag funtkion:

```
function x_nedslag = nedslag(v)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
% Kører først ode45 endnu en gang, hvor jeg gemmer resultatet i en struc
% kaldet sol.
sol = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*v)*500 sin(pi/180*v)*500 ]).
% Finder nulpunkt i matlab:
% Projektil, y position (t):
y = @(t) deval(sol,t,2);
% Nulpunkt via fzero: (finder tidspunktet, hvor y=0, første gang) [sek]
t_0 = fzero(@(t) y(t),[0 30]);
% Projektil, x position (t):
x = @(t) deval(sol,t,1);
% x position hvor projektilet slår ned (indsætter den fundne tid) [m]
x_nedslag = x(t_0);
end
```

```
sol_tjek1 = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*16.03)*500 sin(pi/180*16.03)*500 ]);
y_tjek1 = @(t) deval(sol_tjek1,t,2);
% Nulpunkt via fzero: (finder tidspunktet, hvor y=0, første gang) [sek]
t_tjek1 = fzero(@(t) y_tjek1(t),[0 30]);
x_tjek1 = @(t) deval(sol_tjek1,t,1);
% x position hvor projektilet slår ned (indsætter den fundne tid) [m]
x_value_tjek1 = x_tjek1(t_tjek1)
sol_tjek2 = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*43.34)*500 sin(pi/180*43.34)*500 ]);
y_tjek2 = @(t) deval(sol_tjek2,t,2);
% Nulpunkt via fzero: (finder tidspunktet, hvor y=0, første gang) [sek]
t_tjek2 = fzero(@(t) y_tjek2(t),[0 30]);
x_tjek2 = @(t) deval(sol_tjek2,t,1);
% x position hvor projektilet slår ned (indsætter den fundne tid) [m]
x_value_tjek2 = x_tjek2(t_tjek2)
% Plotter til sidst de fundne løsninger:
[T1,Y1] = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*16.03)*500 sin(pi/180*16.03)*500 ]);
[T2,Y2] = ode45(@odefunb,[0 30],[ 0 2 cos(pi/180*43.34)*500 sin(pi/180*43.34)*500 ]);
figure(3)
plot(Y1(:,1),Y1(:,2),'r',Y2(:,1),Y2(:,2),'b')
legend('projektilets bane for v=16.03 grader','projektilets bane for v=43.34 grader');
title('De 2 vinkler, der rammer den blå kanon');
xlabel('x, position [m]');
ylabel('y, position [m]');
axis([0 1100 0 600]);
```

OPGAVE C

```
%% Opgave C.1
%Konstanter og startværdier opskrives
n=400;
t=0;
M=0.2;
k=1;
L_o=0.1;
g=9.82;
a=M/(2*k);
dl=L_o/n;
%y og længderne bestemmes
j=[1:n];
l=j*dl;
y=l+a*g*(2*l./L_o-l.^2/L_o^2);
%slinky'en tegnes
draw_slinky(t,y,l,0.1,25);
%% Opgave C.2
l=l';
y=y';
t=[0:0.05:0.5];
z0=[y,0*y];
[t z]=ode45(@(t,z0) odeC2(t,z0,l,dl,M,k,L_o,g),t,z0);
yi=z(:,1:401);
figure();
draw_slinky(0.1,yi(t==0.1,:),l,0.1,25);
figure();
draw_slinky(0.25,yi(t==0.25,:),l,0.1,25);
figure();
draw_slinky(0.5,yi(t==0.5,:),l,0.1,25);
%% Opgave C.3
y0=l;
t=[0:0.1:1.5];
z0=[y0,0*y0];
[t z]=ode45(@(t,z0) odeC3(t,z0,l,dl,M,k,L_o,g),t,z0);
yi=z(:,1:401);
figure();
draw_slinky(0,yi(t==0,:),l,0.1,25);
figure();
draw_slinky(0.5,yi(t==0.5,:),l,0.1,25);
figure();
draw_slinky(1.5,yi(t==1.5,:),l,0.1,25);
```

```
function [ dzdt] = odeC2(t,z0,l,dl,M,k,L_o,g)

y1=z0(1:400);
y2=z0(401:end);
dzdt=[y2;ffun(l,y1,dl,M,k,L_o,g)];
end

function [ dzdt] = odeC3(t,z0,l,dl,M,k,L_o,g)

y1=z0(1:400);
y2=z0(401:end);
dzdt=[y2;ffun(l,y1,dl,M,k,L_o,g,true)];
end
```