Projekt 1B: Fjädringssystem

Simon Sjöling, Ayana Musaev

October 18, 2021

Hos hjuldrivna fordon spelar fjädringssystemet en viktig roll. Projektets syfte är att undersöka hur en förenklad modell av fjädringssystemet, quater car-modellen, varierar beroende på olika numeriska metoder.

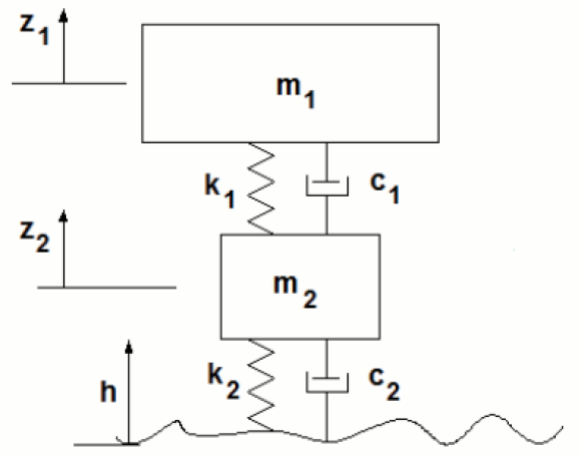

Figure 1: Fjäderingssystem i en quater car modell.

Parametrar som tillkommer med laborationslydelsen är:

m_1	460 [kg]
m_2	60 [kg]
$k_{1,ref}$	5500 [kg/s^2]
$k_{2,ref}$	130000 [kg/s^2]
c_1	300 [kg/s]
c_2	1300 [kg/s]
v	60/3.6 [m/s]
H	0.2 [m]
L	1 [m]

U1.

I denna uppgift skall det rörelseekvationerna för fjäderingsystemet skrivas om så att de går att lösa som ett system av differentialekvationer. På matrisform tecknas följande samband för rörelsen:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{F} \tag{1}$$

med matrisalgebra kan vi lösa ut $\ddot{\boldsymbol{q}}$

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{F} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{K}\boldsymbol{q}) \tag{2}$$

vilket utskrivet tar formen

$$\ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{z_1} \\ \ddot{z_2} \end{bmatrix} = \begin{bmatrix} -\frac{c_1\,\dot{z_1} - c_1\,\dot{z_2} + k_1\,z_1 - k_1\,z_2}{m_1} \\ \frac{c_2\,\dot{h} + c_1\,\dot{z_1} - c_1\,\dot{z_2} - c_2\,\dot{z_2} + k_2\,h + k_1\,z_1 - k_1\,z_2 - k_2\,z_2}{m_2} \end{bmatrix} \tag{3}$$

Den formen vi är intresserade av att skriva (1) är:

$$\frac{d\boldsymbol{v}}{dt} = \boldsymbol{A}\,\boldsymbol{v} + \boldsymbol{g}(t) \tag{4}$$

För att uppnå detta införs nu variabler så att systemet blir av första ordningen. Vi kan då även teckna $\boldsymbol{v}$ och $\frac{d\boldsymbol{v}}{dt}$ i termer av substitutionen:

$$\begin{cases} u_1 = z_1 \\ u_2 = \dot{z}_1 \\ \omega_1 = z_2 \\ \omega_2 = \dot{z}_2 \end{cases} \Rightarrow \boldsymbol{v} = \begin{bmatrix} u_1 \\ \omega_1 \\ u_2 \\ \omega_2 \end{bmatrix} \Rightarrow \frac{d\boldsymbol{v}}{dt} = \begin{bmatrix} u_2 \\ \omega_2 \\ \dot{u}_2 \\ \dot{\omega}_2 \end{bmatrix} \tag{5}$$

med substitutionen och $\ddot{\boldsymbol{q}}$ kan vi teckna $\frac{d\boldsymbol{v}}{dt}$

$$\frac{d\boldsymbol{v}}{dt} = \begin{bmatrix} u_2 \\ w_2 \\ -\frac{c_1\,\dot{z_1} - c_1\,\dot{z_2} + k_1\,z_1 - k_1\,z_2}{m_1} \\ \frac{c_2\,\dot{h} + c_1\,\dot{z_1} - c_1\,\dot{z_2} - c_2\,\dot{z_2} + k_2\,h + k_1\,z_1 - k_1\,z_2 - k_2\,z_2}{m_2} \end{bmatrix} \tag{6}$$

För att få systemet på formen (4) behöver de termer innehållande element i $\boldsymbol{v}$ skrivas på matrisform och de övriga tidsberoende termerna bildar $\boldsymbol{g}(t)$. Resultatet blir följande:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{m_1} & \frac{k_1}{m_1} & -\frac{c_1}{m_1} & \frac{c_1}{m_1} \\ \frac{k_1}{m_2} & -\frac{k_1+k_2}{m_2} & \frac{c_1}{m_2} & -\frac{c_1+c_2}{m_2} \end{bmatrix},\ \boldsymbol{g}(t) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{c_2\,\dot{h} + k_2\,h}{m_2} \end{bmatrix} \tag{7}$$

U2. a)

För denna uppgift har vi skrivit hjälpfunktioner för kod i senare delen av projektet. Dessa funktioner är $roadprofile$ som ger profilen hos guppet vid en tid t och $quartercar$ som returnerar systemets (1) tillstånd vid tiden t, alltså en lösning v. Se kod i appendix 1.1 och 1.2.

U2. b) och c)

Uppgiften diskuteras vid kamratåterkoppling. Kod och plottar för b.) återfinns i appendix 1.4 och för c.) i 1.5.

i.)

Lösningen ses i figur 2. Det maximala utslaget för z_1 och z_2 tillsammans är $0.186\,m$ och för passagerarna innebär detta att som maximum kommer de att guppa upp ca $19\,cm$ gentemot vägen. Maximala utslaget är $0.02\,m$ och $0.183\,m$ för z_1, respektive z_2. Uppgiften diskuteras kamratåterkoppling. Kod återfinns i appendix 1.5.

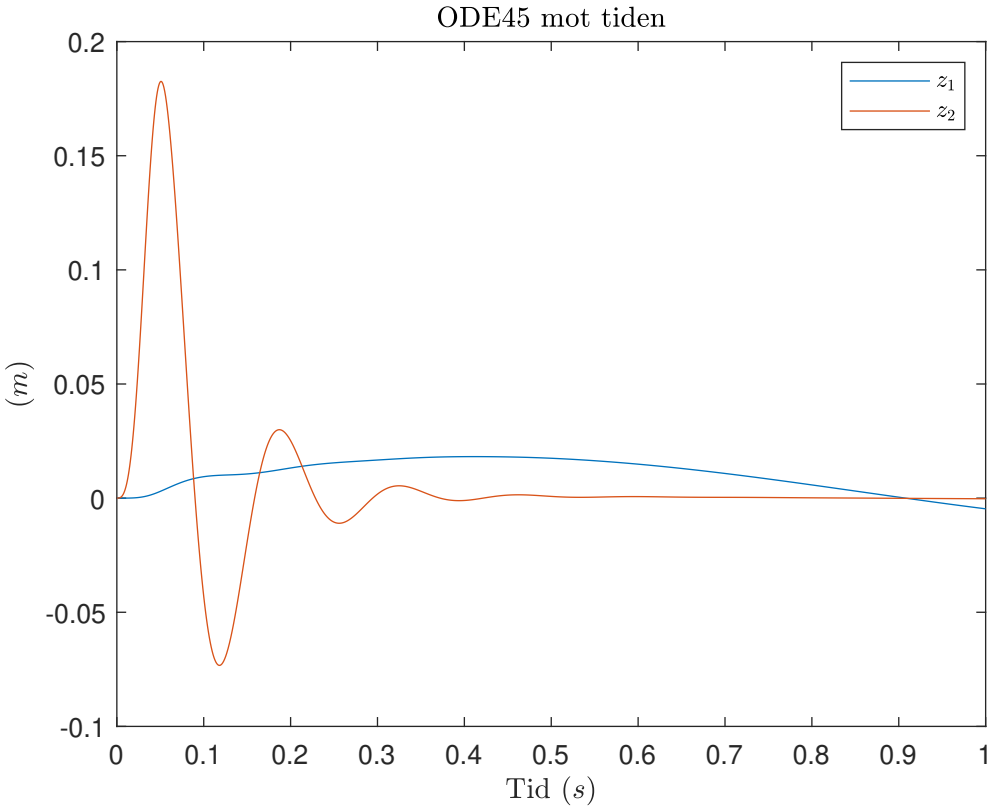

Figure 2: Numerisk lösning för z_1 och z_2 med ODE45 mellan $t = 0$ och $t = 1$. RelTol är satt till 10^{-6}.

ii.)

Vi tolkar nu mängden tidssteg som ODE45-metoden använder vid olika tidpunkter. Det observeras att då funktionen har svängningar med större amplitud, som då $0 \leq t \leq 0.5$ så används fler steg för att fånga dessa stora rörelser. Vid $0.5 \leq t \leq 1$ ser man att det används färre steg då funktionen beter sig mer förutsägbart. Se figur 3.

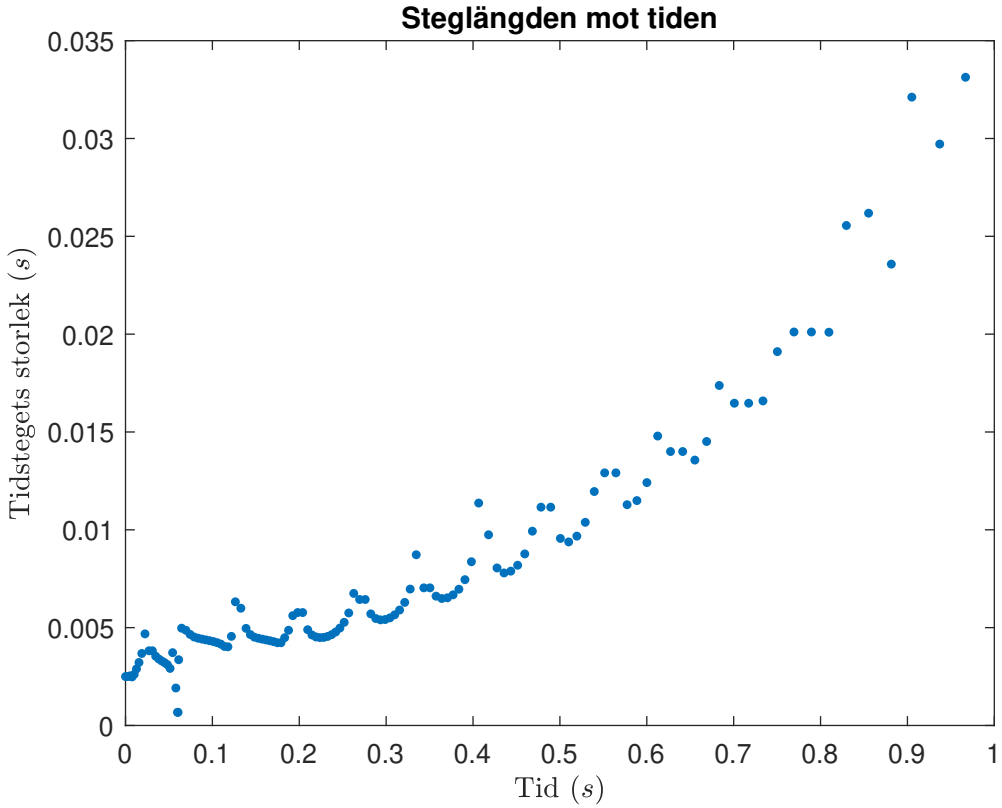

Figure 3: Steglängden till ODE45 mot tiden.

iii.)

Det framkommer genom analys av figur 4 att metoden där tidslängden är mindre ($\Delta t = 5 \cdot 10^{-4}$) följer lösningen för ODE45 bättre och därmed ger en bättre approximativ lösning. Det observeras även att det större tidssteget($\Delta t = 5 \cdot 10^{-3}$) ger en lösning med sämre stabilitet.

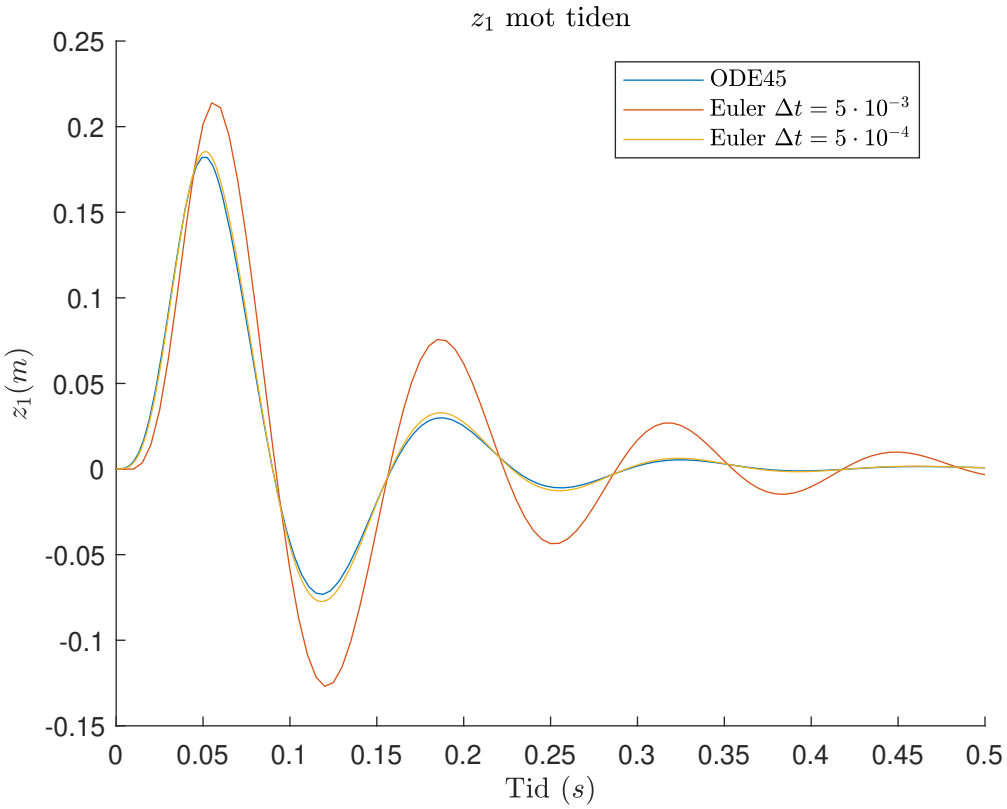

Figure 4: Två lösningar för z_1 med Euler Framåt.

U3.

För att uppnå största komfort för bilföraren och för passagerarna krävs det att fjädringskonstanterna har olika egenskaper: k_1 är liten, som ger mjuk fjädring, och att k_2 är stor, ger hård fjädring. Detta leder till att ekvationssystemet blir ett styvt system. Styva system fungerar bara upptill ett stabilitetsvillkor för varje numerisk metod, därefter blir lösningen instabil. Ett ännu styvare system används i beräkningen i slutet av uppgiften.

i.)

Systemmatrisen $\boldsymbol{A}$ antas vara diagonaliserbar med egenvärden λ_1, λ_2, λ_3, λ_4. Här härleds stabilitetsvillkoret för Euler Framåt på formen

$$\Delta t \leq \Delta t_{max} = \min_{1 \leq k \leq 4} F(\lambda_k) \tag{8}$$

Vi inleder med att teckna stabilitetsvillkoret (som kommer från den generella ekvationen för ordinära differentiella ekvationer):

$$|1 + h\lambda| < 1 \tag{9}$$

Då λ är ett komplext tal på formen $a + b\,i$ så kan vi försöka få ett villkor för h:

$$\begin{aligned} &|1 + h\,(a + b\,i)| < 1 \\ \Leftrightarrow\ &|1 + h\,(a + b\,i)|^2 < 1^2 \\ \Leftrightarrow\ &a^2\,h^2 + 2\,a\,h + a^2\,h^2 + 1 < 1 \\ \Leftrightarrow\ &h^2\,(a^2 + b^2) + 2\,a\,h < 0 \\ \Leftrightarrow\ &h\,\left(h\,(a^2 + b^2) + 2\,a\right) < 0 \end{aligned}$$

När vi ställt upp villkoret på detta sätt så blir det tydligt att det finns två olika villkor till h:

$$\begin{cases} h < \quad 0 \\ h < -\frac{2\,a}{a^2+b^2} \end{cases}$$

Dock är $h < 0$ inte relevant då tidssteget i detta fall alltid skulle vara negativt. Vi kan även se att $a < 0$ då systemet inte är stabilt annars. Därför har vi nu endast kvar ett villkor, $F\,(\lambda)$, som man kan uttrycka på följande sätt:

$$h < \frac{2\,|\,\mathrm{Re}(\lambda)|}{|\lambda|^2} = F\,(\lambda) \tag{10}$$

ii.)

I denna uppgift tas stabilitetsvillkoret fram för Euler Framåt med de givna numeriska parametrarna. Dessutom tas det fram vad det maximala tidssteget blir om den samtidigt uppfyller villkoret.
För egenvärdena $\lambda_1, \lambda_2, \lambda_3$ och λ_4 av systemmatrisen $\boldsymbol{A}$ beräknas $F(\lambda)$. Det minsta värdet på funktionen som ger stabilitetsvillkoret (10), ger även det största värdet som tidssteget h kan anta för att vara stabil.

Maximala tidssteget t_{max} beräknades till värdet $0,0118\,s$. Se koden i appendix 1.6. Värdet jämföres även med tidssteget som användes i U2.iii: $\Delta t = 5 \cdot 10^{-3}$. Detta tidssteg uppfyller stabilitetsvillkoret då den är mindre än t_{max} och därmed är lösningen stabil.

iii.)

I denna deluppgift används samma kod som i U2 ii men tidsteget är istället $\Delta t = \alpha \Delta t_{max}$, där α = 0.9, 1, 1.1, 1.5. Detta för att se hur mycket hjulet förflyttar sig i vertikalled, beroende på hur steglängderna varierar runt Δt_{max}.

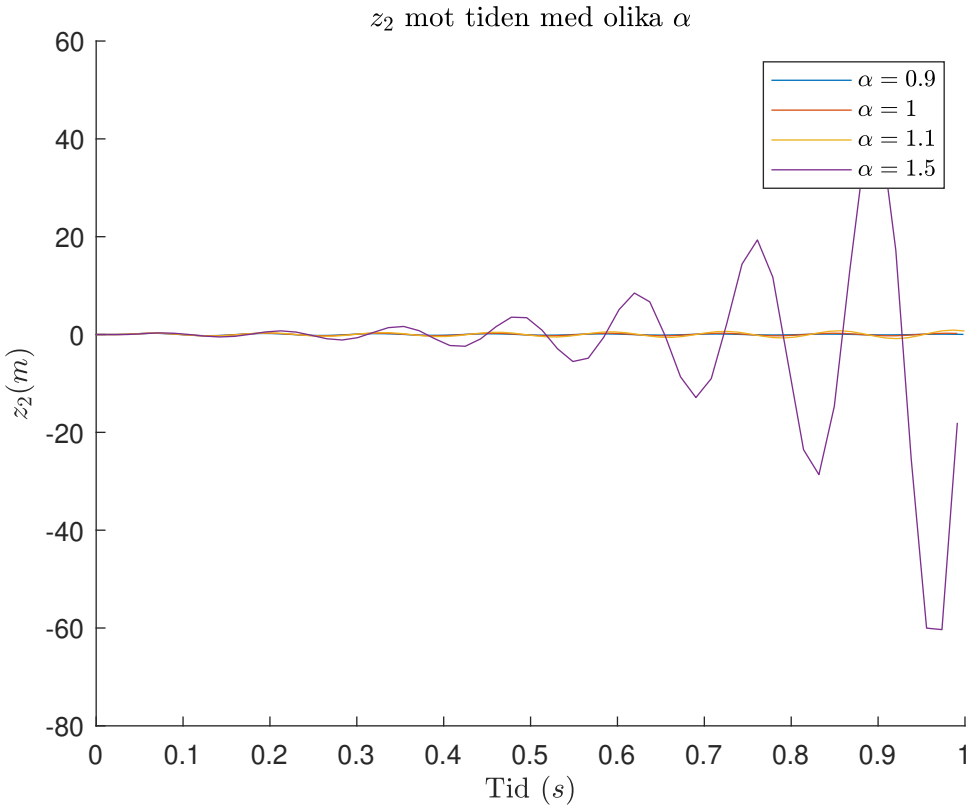

Figure 5: z_2 beräknat för olika α. Det noteras att är särskilt instabil.

Här framkommer det att så länge tidssteget håller sig runt, eller helst under tidssteget som stabilitetsvillkoret begär så kommer hjulet att vara stabil mot marken. Om det inte följer den (som vid $\alpha = 1.5$) då fungerar det inte längre att använda Euler Framåt som metod för quater car-modellen. Se kod 1.7 i appendix.

iv.)

Här införes ett nytt k_2 som ger ett styvare system, nämligen $k_2 = 100 \cdot k_{2,ref}$. För ett styvare system behövs det räknas ut ett nytt maximalt tidssteg Δt_{max}.
Koden för den beräkningen följer här nedan:

```
% Stabilitetsvillkoret
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

% Ber knar egenv rden
eigs = eig(A);
dts = zeros(length(eigs), 1);

% Ber knar villkor f r varje egenv rde och anv nder sedan det minsta.
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end
t_max = min(dts);
```

Resultatet av denna beräkning blev för det styva systemet $\Delta t_{max} = 1.230 \cdot 10^{-4}$.

U4.b)

Styvare system används som i uppgift U3.iv men nu används den implicita trapetsmetoden(ITM) som lämpar sig bättre för styva system. Teorin säger generellt att implicita metoder är absolutstabila oavsett värde på tidssteget. ITM har noggrannhetsordning 2 och därmed behövs det inte lika små tidssteg för att uppnå samma noggrannhet som för Euler Framåt (som har noggrannhetsordning 1). Lösningen från ITM ges på formen

$$\boldsymbol{U}_{i+1} = \boldsymbol{B}^{-1} \left(\boldsymbol{C}\boldsymbol{U}_{i+1} + \boldsymbol{f} \right) \tag{11}$$

där $\boldsymbol{B} = \boldsymbol{I} - \frac{h}{2}\boldsymbol{A}$, $\boldsymbol{C} = \boldsymbol{I} + \frac{h}{2}\boldsymbol{A}$ och $\boldsymbol{f} = \frac{h}{2}\left(\boldsymbol{g}\left(t_i\right) + \boldsymbol{g}\left(t_{i+1}\right) \right)$.

Det definieras en egen funktionfil G.m som ger $\boldsymbol{g}(t)$ för ett givet t för att kunna använda ITM-funktionen för godtyckliga $\boldsymbol{g}(t)$ och $\boldsymbol{A}$.

```
function res=G(H,L,v,k2,c2,m2,t)
% Retunerar G(t)  f  r  ett t samt givna parametrar,  p   vektorform.
    [h,hdot] = roadprofile(H,L,v,t);
    res = [0;0;0;(k2*h + c2*hdot)/m2];
end
```

ITM.m ser nu ut på följande sätt:

```
function [tv, solm]=ITM(A, tv, gfunc, y_0)
    % Implementering av ITM.

    % Antar konstant tidssteg.
    h = tv(2) - tv(1);
    N = length(tv)-1;
    solm = [y_0 zeros(length(y_0), N)]; % S tter yv0 som initalv rden.

    % Ber knar matriser
    I = speye(4);
    B = (I - (h/2) * A);
    C = (I + (h/2) * A);
    % Ber knar U_i+1 f r varje tidssteg.
    for i = 1:N
        f = (h/2)*(gfunc(tv(i)) + gfunc(tv(i+1)));
        solm(:,i+1) = B\(C*solm(:,i) + f);
    end
end
```

i.)

Med det styvare systemet blir stabiliteten hos ITM inte påverkad av steglängden. Om $\Delta t = \alpha t_{max}$ då $\alpha = 1, 10, 100$ ses ingen förändring hos stabiliteten. Detta stämmer överens med teorin om att implicita numeriska metoder inte har något stabilitetsvillkor till skillnad från explicita metod som Euler Framåt. Se figur 6 samt koden i appendix 1.9.

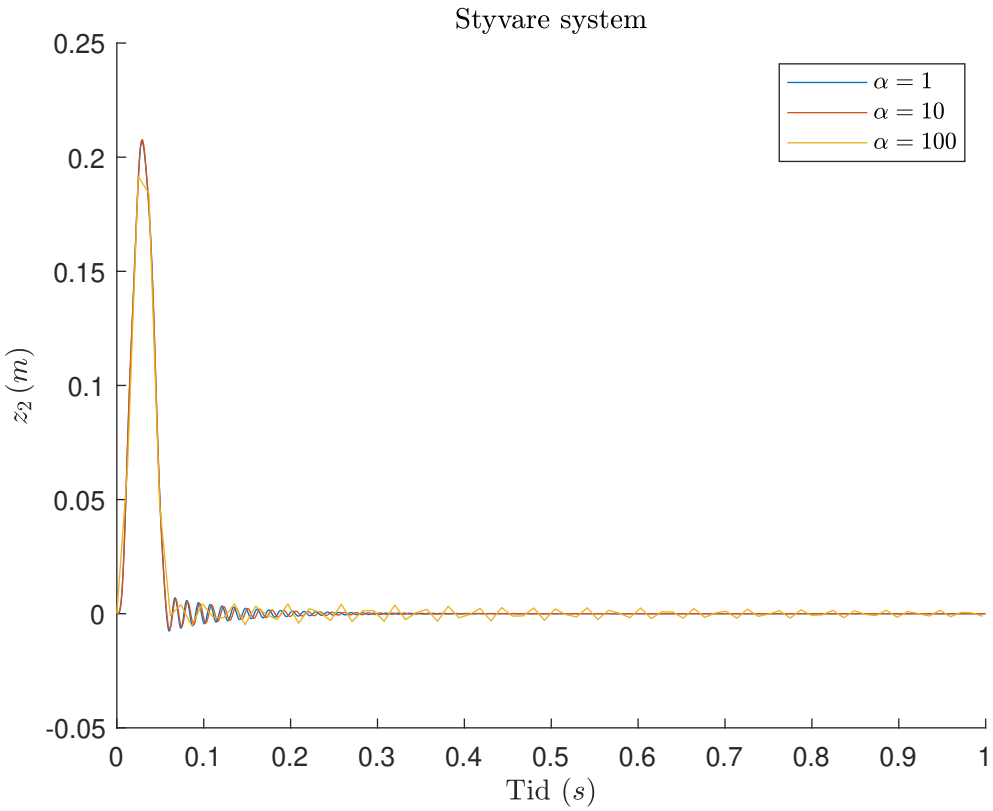

Figure 6: z_2 som representerar hjulet mot tiden.

Maximala tidsteget som det styvare systemet ger är $t_{max} = 1.23 \cdot 10^{-4}$.

ii.)

Det gjordes en konvergensstudie för ITM för det styvare systemet. Vi använder ODE45 med en lägre feltolerans (`AbsTol` och `RelTol` satta till 10^{-9}) som referenslösning. Vi beräknar normen av felet genom att subtrahera referenslösningen med ITM-lösningen och ta fram det största absolutbeloppet av värdena i den resulterade felvektorn.

Normen av felet till komponenten z_2 över tidsintervallet $0 \leq t \leq 0.05$ blev $1.70 \cdot 10^{-7}$. Detta beräknas enligt koden 1.10 i appendix.

Sist tas det fram ett $\Delta t = \alpha \Delta t_0$ där $\alpha = 1,\ 1/2,\ 1/4$ och $1/8$. Felen och noggrannhetsordningen för varje tidssteg beräknas även. Här var det viktigt att Δt_0 är tillräckligt liten så att steglängden ryms i tidsintervallet.
Resultatet för felen på varje α:

α	Fel
$1/8$	$1.826 \cdot 10^{-10}$
$1/4$	$7.57 \cdot 10^{-10}$
$1/2$	$3.063 \cdot 10^{-9}$
1	$1.229 \cdot 10^{-8}$

Svaret hittades numeriskt genom att beräknades först en vektor `tv` innehållande det största antalet tidssteg, d.v.s då $\alpha = 1/8$. Då $\alpha = 1/4$, $1/2$, 1 så används en delmängd av värdena i tidsvektorn, t.ex. då $\alpha = 1/4$ används varannan komponent, då $\alpha = 1/2$ används var fjärde, o.s.v. För varje α beräknas felet av med referenslösningen. Därefter beräknas noggrannhetsordningen. Implementeringen ser ut på följande sätt:

```
% Ber knar tidsvektor
tend = 0.05;
h = (1/8) * t_0; % F rst  f  r det minsta v rdet  p   alpha
N = floor(tend/h);
tv = h*(0:N); % Ber knar alla tider i tidsspannet.

% Ber knar referensl sning med ODE45.
opts = odeset('RelTol', 1e-9, 'AbsTol', 1e-9);
[tvref, yref] = ode45(vfunc, tv, v_0, opts);

pvec = zeros(1, 4);
for n=0:3
    tvn = tv(1:2^n:end);
    [~, solm] = ITM(A, tvn, gfunc, v_0);
    errv = solm(2,:)' - yref(1:2^n:end,2); % 2^n ger hur m nga steg som ska tas i
        tidsvektorn.
    err = max(abs(errv));
    pvec(4 - n) = err; % Fyller backl nges.
end
% Ber knar noggrannhetsordning
pe10 = log(pvec(1) / pvec(2))/log(2);
pe20 = log(pvec(2) / pvec(3))/log(2);
pe40 = log(pvec(3) / pvec(4))/log(2);
```

Resultatet för noggrannhetsordningen blir `pe10` = 1.999, `pe20` = 1.997 och `pe40` = 1.986, vilket stämmer överens med det teoretiska värdet som är 2. Se full kod 1.11 i appendix.

1 Appendix

1.1 roadprofile.m

```
function [h,hdot] = roadprofile(H,L,v,t)
% Returnerar profilen av v gen  f r givna parametrars v rde.
if(t <= L/v)
    h = (H/2)*(1-cos((2*pi*v*t)/L));
    hdot = (H/2)*((((2*pi*v)/L)*sin((2*pi*v*t)/L)));
else
    h = 0;
    hdot = 0;
end
```

1.2 quartercar.m

```
function dy = quartercar(t,y,A,k2,c2,m2,H,L,v)
% Funktionen definierar och returnerar h gerledet i systemet dy/dt = A*y(t) + g(t)
[h,hdot] = roadprofile(H,L,v,t);
g = [0;0;0;(k2*h + c2*hdot)/m2];
dy = A*y + g;

end
```

1.3 EulerF.m

```
function [tv, solm]=EulerF(func, h, tspan, yv0)
    % Implementation av EulerF.
    N = floor((tspan(2) - tspan(1))/h);
    solm = [yv0 zeros(length(yv0), N)]; % S tter yv0 som initalv rden.
    tv = tspan(1)+h*(0:N); % Ber knar alla tider i tidsspannet.
    % Utf r loopen och ber knar  f  r varje t i intervallet.
    for i = 1:N
        solm(:,i+1) = solm(:,i) + h*func(tv(i), solm(:,i));
    end
end
```

1.4 U2_b.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];
% Initialvillkor
v_0 = [0;0;0;0];
% Tidesteget
dt = 5*10^-4;

% En anonym funktion som bara  r  beroende av t. Detta  r  n dv ndigt
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);
[tv, solm] = EulerF(vfunc, dt,[0, 1], v_0); % Se EulerF.m  f  r  l sningsmetod.

% Plottar nu  l sning av z_1 och z_2  fr n  l sningsmatrisen.
plot(tv, solm(1:2,:))
legend(['$z_1$', '$z_2$', '$\dot{z_1}$', '$\dot{z_2}$'],'Interpreter','latex')

title('EulerF med $\Delta t = $' + num2str(dt) + ' mot tiden','Interpreter','latex');
legend(['$z_1$', '$z_2$'],'Interpreter','latex');
xlabel('Tid $(s)$','Interpreter','latex');
ylabel('$(m)$','Interpreter','latex');
saveas(gcf,'plot_U2_c','epsc');
```

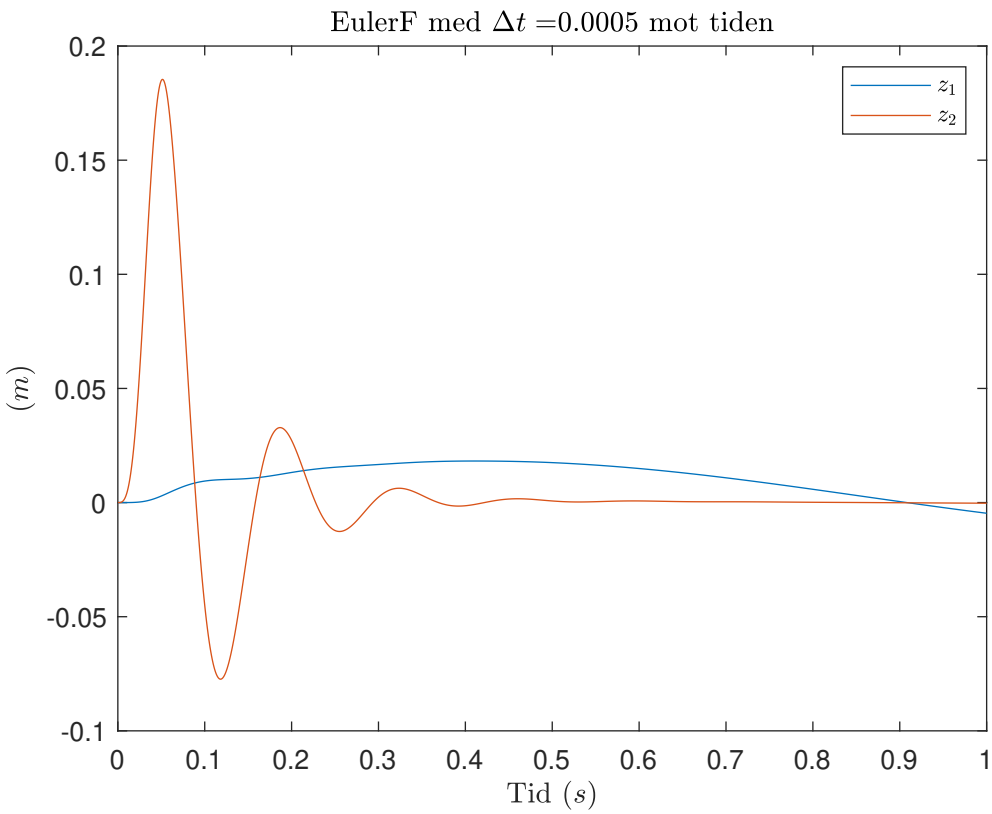

Figure 7: Numerisk lösning för z_1 och z_2 med Euler framåt mellan $t = 0$ och $t = 1$.

1.5 U2 c.) och U3 c.) i.)

```
clc, clear
% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];
% Initialvillkor
v_0 = [0;0;0;0];
% Tidesteget
dt = 5*10^-4;

% En anonym funktion som bara  r  beroende av t. Detta  r  n dv ndigt
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);

opts = odeset('RelTol', 1e-6);
[tv, solm] = ode45(vfunc, [0, 1], v_0, opts);
plot(tv, solm(:,1:2));

% Printar maxv rdet  f  r  z_1 och z_2
fprintf('Maxv rde  f  r  z_1: %d \n Maxv rde  f  r  z_2: %d', max(solm(:,1)), max(solm
    (:,2)));

title('ODE45 mot tiden','Interpreter','latex');
legend(['$z_1$', '$z_2$'],'Interpreter','latex');
xlabel('Tid $(s)$','Interpreter','latex');
ylabel('$(m)$','Interpreter','latex');
```

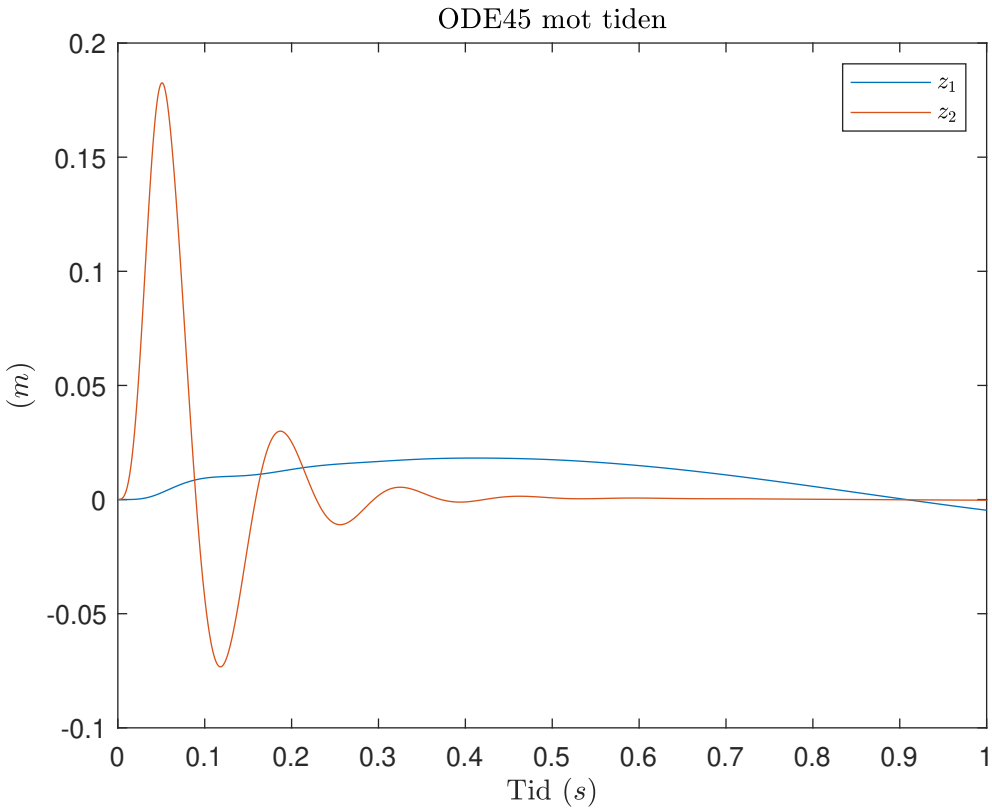

Figure 8: Numerisk lösning för z_1 och z_2 med ODE45 framåt mellan $t = 0$ och $t = 1$.

1.6 U3_ii.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
```

```
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];
% Framtaget stabilitetsvillkor.
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1); % Array av v rden med F(lambda) f r olika egenv rden.
for i=1:length(eigs) % V rderar F f r varje egenv rde.
   dts(i) = F(eigs(i));
end

% S tter de minsta v rdet till t_max.
t_max = min(dts);

fprintf('T_max = %d', t_max);
```

1.7 U3_iii.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];

% Ber knar t_max
% Framtaget stabilitetsvillkor.
```

```
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1);
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end

t_max = min(dts);
dt = t_max * 0.1;

% Initialvilkor
v_0 = [0;0;0;0];

tspan = [0, 1];
% Anonym funktion som kan anv ndas av Euler-metoden.
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);
%
[tv, solm] = EulerF(vfunc, dt, tspan, v_0);
plot(tv, solm(1:2,:));
title_str = 'Styvare system med $\Delta t = 0.1 \cdot t_{max} = $' + num2str(dt) + '
    $\,s$';
legend(['$z_1$', '$z_2$'],'Interpreter','latex');
title(title_str, 'Interpreter','latex');
xlabel('Tid $(s)$','Interpreter','latex');
ylabel('$z_1 / z_2 \, (m)$','Interpreter','latex');
```

1.8 U3_iv.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];

% Ber knar t_max
% Framtaget stabilitetsvillkor.
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1);
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end

t_max = min(dts);
dt = t_max * 0.1;

% Initialvilkor
v_0 = [0;0;0;0];

tspan = [0, 1];
% Anonym funktion som kan anv ndas av Euler-metoden.
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);
```

```
%
[tv, solm] = EulerF(vfunc, dt, tspan, v_0);
plot(tv, solm(1:2,:));
title_str = 'Styvare system med $\Delta t = 0.1 \cdot t_{max} = $' + num2str(dt) + '
    $\,s$';
legend(['$z_1$', '$z_2$'],'Interpreter','latex');
title(title_str, 'Interpreter','latex');
xlabel('Tid $(s)$','Interpreter','latex');
ylabel('$z_1 / z_2 \, (m)$','Interpreter','latex');
```

1.9 U4_b_i.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];

% Initialvilkor
v_0 = [0;0;0;0];

gfunc = @(t) G(H,L,v,k2,c2,m2,t);

% Functionen f r ODE45
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);

```

```
% Ber knar t_max
% Framtaget stabilitetsvillkor.
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1);
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end

t_max = min(dts);
fprintf('T_max: %d', t_max);
% Implementation av implicita trapetsmetoden.
tspan = [0, 1];

alpha_vals = [1, 10, 100];

hold on;
for n=1:length(alpha_vals)
    h = alpha_vals(n) * t_max;
    N = floor((tspan(2) - tspan(1))/h);
    tv = tspan(1)+h*(0:N); % Ber knar alla tider i tidsspannet.

    % Anv nder ITM-funktionen.
    [tv, solm] = ITM(A,tv,gfunc,v_0);

    plot(tv, solm(1,:));
end

legend(['$\alpha = 1$', '$\alpha = 10$', '$\alpha = 100$', 'ODE45'],'Interpreter','
    latex');
title_str = 'Styvare system med $t_{max} = $' + num2str(t_max) + '$\,s$';
title(title_str, 'Interpreter','latex');
xlabel('Tid $(s)$','Interpreter','latex');
ylabel('$z_2 \, (m)$','Interpreter','latex');
```

1.10 U4_b_ii.m

```
clc, clear

% Givna systemparametrar.
m1 = 460; m2 = 60;
k1_ref = 5500; k2_ref = 130000;
k1 = k1_ref; k2 = k2_ref;
c1 = 300; c2 = 1300;
v  = (60/3.6);
H = 0.2;
L = 1;

% Systemmatrisen
A = [0, 0, 1, 0;
    0, 0, 0, 1;
    -k1/m1, k1/m1, -c1/m1, c1/m1;
    k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
    ];

% Initialvilkor
v_0 = [0;0;0;0];

gfunc = @(t) G(H,L,v,k2,c2,m2,t);

% Functionen f r ODE45
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);

% Ber knar t_max
% Framtaget stabilitetsvillkor.
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1);
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end

t_0 = min(dts);
```

```
38
39 % Ber knar tidsvektor
40 tend = 0.05;
41 h = 0.1 * t_0;
42 N = floor(tend/h);
43 tv = h*(0:N); % Ber knar alla tider i tidsspannet.
44
45 % Ber knar referenslsning med ODE45.
46 opts = odeset('RelTol', 1e-9, 'AbsTol', 1e-9);
47 [tvref, yref] = ode45(vfunc, tv, v_0, opts);
48
49 % Ber knar med ITM.
50 [tv, solm] = ITM(A,tv,gfunc,v_0);
51
52 % Normen av felet tas nu fram nu.
53 errvec = solm(2,:)' - yref(:,2);
54 err = max(abs(errvec));
55 disp(err);
```

1.11 U4_b_ii_2.m

```
1  % Givna systemparametrar.
2  m1 = 460; m2 = 60;
3  k1_ref = 5500; k2_ref = 130000;
4  k1 = k1_ref; k2 = k2_ref;
5  c1 = 300; c2 = 1300;
6  v  = (60/3.6);
7  H = 0.2;
8  L = 1;
9
10 % Systemmatrisen
11 A = [0, 0, 1, 0;
12     0, 0, 0, 1;
13     -k1/m1, k1/m1, -c1/m1, c1/m1;
14     k1/m2, -(k1 + k2)/m2, c1/m2, -(c1 + c2)/m2
15     ];
16
17 % Initialvilkor
```

```
v_0 = [0;0;0;0];

% Anonym funktion som retunerar g(t) f r ett t.
% G r s att inte samtliga parametrar m ste skickas med manuellt vid varje anrop.
gfunc = @(t) G(H,L,v,k2,c2,m2,t);

% Functionen f r ODE45
vfunc = @(t, y) quartercar(t,y,A,k2,c2,m2,H,L,v);

% Ber knar t_max
% Framtaget stabilitetsvillkor.
F = @(lambda) 2*abs(real(lambda)) / abs(lambda)^2;

eigs = eig(A);
dts = zeros(length(eigs), 1);
for i=1:length(eigs)
   dts(i) = F(eigs(i));
end

t_0 = 10^-5;

% Ber knar tidsvektor
tend = 0.05;
h = (1/8) * t_0; % F rst f r det minsta v rdet p alpha
N = floor(tend/h);
tv = h*(0:N); % Ber knar alla tider i tidsspannet.

% Ber knar referensl sning med ODE45.
opts = odeset('RelTol', 1e-9, 'AbsTol', 1e-9);
[tvref, yref] = ode45(vfunc, tv, v_0, opts);

pvec = zeros(1, 4);
for n=0:3
    tvn = tv(1:2^n:end);
    [~, solm] = ITM(A, tvn, gfunc, v_0);
    errv = solm(2,:)' - yref(1:2^n:end,2); % 2^n ger hur m nga steg som ska tas i
        tidsvektorn.
    err = max(abs(errv));
```

```
    pvec(4 - n) = err; % Fyller backl nges.
end

% Ber knar noggrannhetsordning
pe10 = log(pvec(1) / pvec(2))/log(2);
pe20 = log(pvec(2) / pvec(3))/log(2);
pe40 = log(pvec(3) / pvec(4))/log(2);

fprintf('alpha=1/8: %d \n alpha=1/4: %d \n alpha=1/2: %d \n alpha=1: %d \n', pvec(1),
    pvec(2), pvec(3));
fprintf('pe10: %d \n pe20: %d \n pe40: %d \n', pe10, pe20, pe40);
```