Laboration 1

Laborationen görs i grupper om två, men examineras individuellt genom datorprov. Vid datorprovet kommer ni att ha tillgång till den kod ni skrivit för uppgifterna nedan. Provet kommer även att innehålla teorifrågor om de koncept som arbetas med i laborationen. För mer information, se kurshemsidan.

Utnyttja gärna alla schemalagda handledningstillfällen för att ställa frågor. Häftet "Matlab 8 i korthet", liksom den interaktiva "Matlab Onramp", ger en stegvis introduktion till Matlab *som kan vara användbar (se Canvas). Flitigt utnyttjande av* Matlabs `help`*-kommando rekommenderas också.*

Syftet med denna laboration är att:

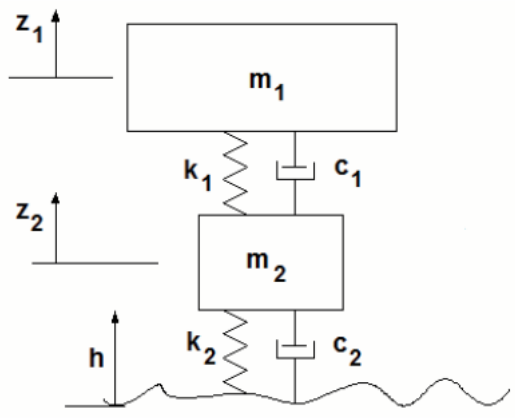
- *lösa ett system av begynnelsevärdesproblem med både explicita metoder från kursen (Euler framåt och* Matlabs *inbyggda* `ode45`*) och en implicit metod (implicita mittpunktsmetoden).*
- *illustrera de centrala begreppen stabilitet och noggrannhetsordning.*
- *implementera Newtons metod och verifiera dess konvergensordning.*

Bakgrund

Laborationen behandlar en viktig komponent hos hjuldrivna fordon, nämligen fjädringssystemet. Ett fjädringssystem är ett system som innehåller fjädrar och stötdämpare som kopplar samman ett fordon med dess hjul. Fjädringssystemet tillåter att hjulen och bilchassit rör sig relativt varandra och har som uppgift att bidra till både god väghållning och till att göra färden bekväm. För att designa ett fjädringssystem som hanterar båda dessa uppgifter på ett tillfredställande sätt behöver man hitta en kompromiss då uppgifterna till stor del är motstridiga. Förenklat uttryckt ger hård fjädring goda väghållningsegenskaper medan mjuk fjädring ger hög komfort.

Matematisk beskrivning av ett fjädringssystem

En enkel modell av ett fjädringssystem för en bil visas i Figur 1. Modellen kallas för en quarter car-modell eftersom det är ett fjädringssystemet för ett hjul kopplat till en fjärdedel av bilens chassi. Modellen kan göras mer komplex genom att ta med flera hjul och en större del av chassit i så kallade half car-(två hjul) eller full car-(fyra hjul) modeller.



Figur 1: Fjädringssystem i en quarter car modell.

I Figur 1 motsvarar m_1 massan för chassi-delen och m_2 massan för hjulet, k_1 och k_2 är fjäderkonstanter och c_1 och c_2 är dämpningskonstanter. Vidare är h vägunderlagets vertikala position och z_1 och z_2 motsvarar förflyttning av de två fjädrarna. På en bil tillhör fjäderkonstanten k_2 den fjäder som har kontakt med vägen via hjulet och fjäderkonstanten k_1 den fjäder som är kopplad till bilkupén.

Rörelseekvationer för quarter car-systemet ges av de ordinära differentialekvationerna (ODE) för z_1 och z_2,

$$\begin{aligned} &m_1\ddot{z_1} + c_1(\dot{z}_1 - \dot{z}_2) + k_1(z_1 - z_2) = 0, \\ &m_2\ddot{z_2} + c_1(\dot{z}_2 - \dot{z}_1) + c_2(\dot{z}_2 - \dot{h}) + k_1(z_2 - z_1) + k_2(z_2 - h) = 0, \end{aligned}$$

där vi använt notationen $\frac{d}{dt}z_1 = \dot{z}_1$, $\frac{d^2}{dt^2}z_1 = \ddot{z}_1$. Rörelseekvationerna kan också skrivas på matris-vektor-form,

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

där koordinaterna och deras tidsderivator är

$$\boldsymbol{q} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \quad \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix}, \quad \ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{bmatrix},$$

och systemmatriserna är

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} c_1 & -c_1 \\ -c_1 & c_1 + c_2 \end{bmatrix}, \quad \boldsymbol{K} = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 + k_2 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} 0 \\ k_2 h + c_2 \dot{h} \end{bmatrix}.$$

$\boldsymbol{M}$ kallas massmatrisen, $\boldsymbol{C}$ dämpningsmatrisen, $\boldsymbol{K}$ styvhetsmatrisen och $\boldsymbol{F}$ kraftvektorn. Notera att vägunderlagets påverkan i systemet kommer in genom kraftvektorn $\boldsymbol{F}$.

Låt vägunderlaget beskrivas av funktionen

$$h(t) = \begin{cases} \frac{H}{2}\left(1 - \cos\left(\frac{2\pi u t}{L}\right)\right), & t \le L/u, \\ 0, & t > L/u, \end{cases} \tag{2}$$

vilket beskriver ett gupp med höjd H och längd L.

Parametrar till ekvation (1) återfinns i Tabell 1.

parameter	värde [enhet]
m_1	475 [kg]
m_2	53 [kg]
$k_{1,ref}$	5400 [kg/s^2]
$k_{2,ref}$	135000 [kg/s^2]
c_1	310 [kg/s]
c_2	1200 [kg/s]
u	65/3.6 [m/s]
H	0.24 [m]
L	1 [m]

Tabell 1: Parametrar till rörelseekvationerna (1).

1. Numerisk lösning av ODE: explicit tidsintegrering

Uppgift 1. Omskrivning

Skriv om rörelseekvationerna (1) som ett första ordningens system i tiden på formen

$$\frac{d\boldsymbol{v}}{dt} = \boldsymbol{A}\boldsymbol{v} + \boldsymbol{g}(t), \tag{3}$$

där $\boldsymbol{v} = [z_1, z_2, \dot{z}_1, \dot{z}_2]^T$ är vektorn som innehåller de okända variablerna.

Uppgift 2. Tidsintegrering med explicit metod

Ni ska nu lösa problemet numeriskt genom att integrera i tiden. Skriv en funktion `quartercar.m` som innehåller systemet från Uppgift 1. Skriv en kod som löser problemet numeriskt med

- Framåt Euler.
- Matlabs inbyggda funktion `ode45`.

Antag att de vertikala hastigheterna och positionerna är noll initialt. Använd parametrarna i tabellen, med $k_1 = k_{1,ref}$ och $k_2 = k_{2,ref}$. Ni kan själva välja sluttid för simuleringarna, men tänk på att sluttiden bör vara betydligt längre än tiden det tar att köra över guppet för att effekten på fjädringssystemet ska synas ordentligt.

a) Plotta den numeriska lösningen för z_1 och z_2 som funktion av tiden då du använt `ode45` med relativ tolerens 10^{-6}. Vad blir det maximala utslaget för z_1 och z_2 och vad innebär det för passagerarna?

b) Visualisera storleken på tidstegen för `ode45`. Hur förändras tidsstegsstorleken med tiden? Förklara förändringarna i tidsstegen över tid genom att studera plottarna för den numeriska lösningen och tidsstegen tillsammans. (Kom ihåg parametern `refine` för `ode45` som du vill ändra beroende på om du vill ha en

snygg plot eller se hur stora tidssteg som använts). Du finner fler tips om `ode45` i slutet av detta dokument.

c) Jämför lösningarna för Eulers metod med $\Delta t = 5 \cdot 10^{-3}$ och $\Delta t = 5 \cdot 10^{-4}$ med lösningen med `ode45` (plotta lösningarna för z_2 i samma figur) och kommentera.

2. Styva problem: stabilitet, noggrannhet och implicita metoder

För att bibehålla ett fast väggrepp som samtidigt inte leder till en alltför skakig färd för passagerarna vill man typiskt ha hård fjädring (dvs en stor fjäderkonstant) vid hjulen och mjuk fjädring (liten fjäderkonstant) vid bilkupén. Detta kan leda till en god kompromiss för de egenskaper fjädringssystemet bör ha. För de matematiska ekvationerna leder stor variation i fjäderkonstanterna till ett så kallat styvt system, som är mer utfordrande att lösa numeriskt jämfört med ett icke-styvt problem. Styva system kännetecknas av att olika komponenter i systemet förändras på olika tidsskalor. En komponent som varierar långsamt kanske behöver lösas för en lång tid medan en annan innehåller snabba variationer under en kortare tid och att det sedan inte händer så mycket. Här ska ni studera hur olika numeriska metoder för tidsintegrering beter sig beroende på problemets egenskaper samt val av tidssteg.

Uppgift 3. Stabilitet

a) Antag att systemmatrisen i systemet från Uppgift 1 är diagonaliserbar och att egenvärdena är $\lambda_1, \lambda_2, \lambda_3, \lambda_4$. Härled stabilitetsvillkoret för framåt Euler, och uttryck det på formen

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

b) Beräkna stabilitetsvillkoret för Euler framåt för systemet när parametrar från Tabell 1 används. Hur stort blir det maximala tidsteget Δt_{max} för vilket stabilitetsvillkoret uppfylls? Uppfyller $\Delta t = 5 \cdot 10^{-3}$ som användes i Uppgift 2 c) stabilitetsvillkoret?

c) Kör koden ni skrev för framåt Euler i Uppgift 2 med $\Delta t = \alpha \Delta t_{max}$ för $\alpha = 0.9, 1, 1.1, 1.5$. Vad händer?

d) Låt oss nu betrakta ett styvare system, med hårdare fjädring vid hjulet. Välj $k_2 = 100 k_{2,ref}$ och räkna ut det nya maximala tidsteget Δt_{max}. Kör koden med $\Delta t = 0.1 \Delta t_{max}$ och visualisera z_1 och z_2 som funktion av tiden.

För explicita metoder och styva system kan stabilitetsvillkoret leda till en skarp begränsning på tidssteget. Ett alternativ är att istället använda en implicit tidsintegreringsmetod. Implicita metoder har generellt mycket goda stabilitetsegenskaper. En nackdel är att ett linjärt ekvationssystem behöver lösas i varje tidssteg. För många styva problem blir det

ändå mer effektivt att använda en implicit metod då det går att ta ett mycket större tidssteg jämför med för en explicit metod.

Med en parameter θ och ett tidssteg Δt kan vi skriva en diskretisering av systemet i (3) som

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \Delta t \left((1-\theta)(\boldsymbol{A}\boldsymbol{v}^n + \boldsymbol{g}^n) + \theta(\boldsymbol{A}\boldsymbol{v}^{n+1} + \boldsymbol{g}^{n+1})\right), \quad n = 0, 1, 2, \ldots.$$

där index n indikerar en lösning vid tiden $n\Delta t$. Med $\theta = 0$ ger detta diskretiseringen med framåt Euler, för $\theta = 1$ bakåt Euler, och för $\theta = 1/2$ kallas diskretiseringen den implicita trapetsmetoden.

Att en metod är stabil betyder inte att den är noggrann. Både stabilitet och noggrannhet är viktiga egenskaper när man väljer en numerisk metod för tidsintegrering. Ett lämpligt alternativ till att använda en första ordningens implicit metod (som Euler bakåt) med ett litet tidssteg (nödvändigt för tillräcklig noggrannhet) är att använda en implicit metod med högre noggrannhetsordning. På så sätt går det att ta relativt stora tidssteg även för styva problem utan att kompromissa med noggrannheten.

Uppgift 4. Implicita trapetsmetoden

Gör en implementation av den implicita trapetsmetoden ($\theta = 1/2$).

a) Använd parametervärden och initialvillkor för det styvare problemet i Uppgift 3 d), så att $k_2 = 100k_{2,ref}$. Kör koden med tidssteget $\Delta t = \alpha \Delta t_{max}$ med det Δt_{max} som beräknats för Euler framåt, för $\alpha = 1, 10, 100$. Vad händer? Stämmer det med teorin?

b) Vi vill nu göra en konveregensstudie för metoden över intervallet fram till $t = 0.05$ s med $k_2 = 100k_{2,ref}$. Mät normen av felet som det maximala absolutbeloppet av felet i komponenten z_2 över tidsintervallet.

   En mycket noggrann numerisk lösning kan användas istället för en exakt (analytisk) lösning för att mäta felen. En sådan *referenslösning* kan skapas med `ode45` med en strikt feltolerans (t.ex. relativa och absoluta feltoleransen satt till 10^{-9}). Ändra inparametern `tspan` för att få lösningen i de tidpunkter du vill.

   Välj ett Δt_0 för trapetsmetoden. Kör koden med $\Delta t = \alpha \Delta t_0$ för $\alpha = 1, 1/2, 1/4, 1/8$ och beräkna felen och noggrannhetsordningen. Om du inte ser den noggrannhetsordning du förväntar dig: prova att ta ett mindre Δt_0.

3. Optimering av fjäderkonstanter: Newtons metod för icke-linjära system

Som tidigare nämnts behöver man hitta en bra kompromiss mellan god komfort och god säkerhet (väghållning) vid design av bilens fjädringssystem. Här ska ni bestämma fjäderkonstanter så att bekvämligheten ökar utan att äventyra säkerheten. Här behöver ni applicera Newtons metod på ett system med två variabler, fjäderkonstanterna k_1 och k_2.

En överföringsfunktion (eller signalfunktion) beskriver sambandet mellan insignal och utsignal i frekvensplanet för ett linjärt tidsoberoende system. Överföringsfunktionen tas fram genom *Laplacetransform* av rörelseekvationerna (1) med avseende på tiden. Detta ger en tidsoberoende överföringsfunktion som beror på Laplacevariabeln s istället för tiden. I den här laborationen nöjer vi oss med att se på den generella formen av överföringsfunktionerna (ingen specifik kunskap om Laplacetransformer efterfrågas).

Överföringsfunktionerna för komfort respektive säkerhet ges av $T_k(s)$ och $T_s(s)$, som beskrivs med sambanden

$$\begin{aligned} Z_1(s) &= T_k(s)H(s), \\ Z_2(s) - H(s) &= T_s(s)H(s), \end{aligned}$$

där $Z_1(s), Z_2(s), H(s)$ är de Laplacetransformerade motsvarigheterna till $z_1(t), z_2(t), h(t)$.

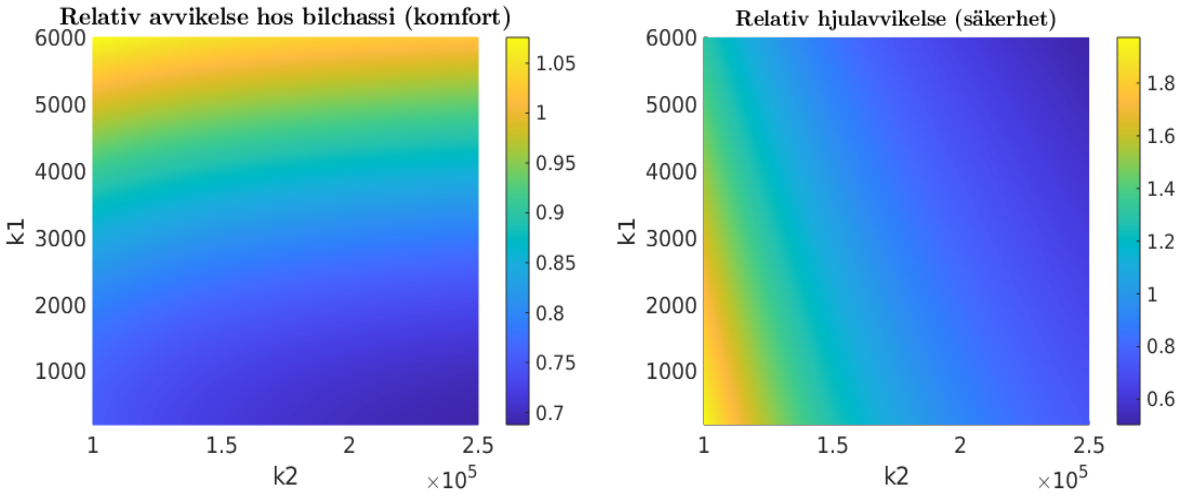
Notera: $T_k(s)$ beskriver sambandet mellan vägunderlaget och hur mycket passagerarna förflyttar sig i vertikalled, medan $T_s(s)$ beskriver sambandet mellan vägunderlaget och skillnaden mellan vägunderlag och hjul. För att maximera bekvämlighet vill man minimera $|T_k(s)|$, och motsvarande för säkerhet och $|T_s(s)|$.

Ofta är man intresserad av att undersöka hur överföringsfunktioner beror på olika frekvenser, och därför beskrivs T_k och T_s som funktioner av s. Här är vi intresserade av att undersöka hur överföringsfunktionerna beror på fjäderkonstanterna för en specifik frekvens (ett fixt värde på s) och beskriver dem istället som funktioner av k_1, k_2, dvs $T_k(k_1, k_2)$, $T_s(k_1, k_2)$. Frekvensen fås från $h(t)$ i (2), där $s = i\omega$, $\omega = \frac{2\pi u}{L}$.

Notera: Då s är ett imaginärt tal kommer utvärdering av överföringsfunktionerna ge komplexa tal. Därför studerar vi deras normerade absolutbelopp, definierade som

$$\tilde{T}_k(k_1, k_2) = \frac{|T_k(k_1, k_2)|}{|T_k(k_{1,ref}, k_{2,ref})|} \quad \text{och} \quad \tilde{T}_s(k_1, k_2) = \frac{|T_s(k_1, k_2)|}{|T_s(k_{1,ref}, k_{2,ref})|}, \tag{4}$$

där $k_{1,ref}$, $k_{2,ref}$ är parametervärden enligt Tabell 1. Vidare kommer vi att referera till $\tilde{T}_k(k_1, k_2)$ och $\tilde{T}_s(k_1, k_2)$ som överföringsfunktionerna. Eftersom att överföringsfunktionerna är normerade har vi att $\tilde{T}_k(k_{1,ref}, k_{2,ref}) = 1$, $\tilde{T}_s(k_{1,ref}, k_{2,ref}) = 1$.



Figur 2: Överföringsfunktioner för komfort, $\tilde{T}_k(k_1, k_2)$, respektive säkerhet, $\tilde{T}_s(k_1, k_2)$, som funktioner av k_2 och k_1. Vi har att $\tilde{T}_k(k_{1,ref}, k_{2,ref}) = 1$, $\tilde{T}_s(k_{1,ref}, k_{2,ref}) = 1$.

De icke-linjära funktionerna $\tilde{T}_k(k_1, k_2)$ och $\tilde{T}_s(k_1, k_2)$ med avseende på fjäderkonstanterna k_1 och k_2 illustreras i Figur 2. Notera från Figur 2 att för att öka bekvämligheten behöver värdet på k_1 minska, men om man samtidigt inte vill att väghållningsegenskaperna ska försämras ser man att k_2 samtidigt behöver öka.

Uppgift 5. Newtons metod

Ni vill bestämma fjäderkonstanter så att färden blir mer bekväm, utan att äventyra säkerheten. Överföringsfunktionen för komfort, $\tilde{T}_k(k_1, k_2)$, ska vara 71% av det tidigare värdet medan överföringsfunktionen för säkerhet, $\tilde{T}_s(k_1, k_2)$, inte ska förändras. Skriv ett program som numeriskt beräknar fjäderkonstanterna k_1 och k_2 med Newtons metod för system.

a) Vad blir de nya optimerade värdena på k_1 och k_2? Ni kan använda $k_{1,ref}$, $k_{2,ref}$ som initialgissning och toleransen 10^{-6}.

b) Verifiera att din implementering av Newtons metod konvergerar med förväntad konvergensordning. Testa med lite olika startgissningar. Ger alla samma lösning, eller konvergerar metoden mot en annan lösning för någon startgissning?

c) Kör koden ni skrev i Uppgift 2 med de nya värdena på k_1 och k_2 och plotta z_1 som en funktion av tiden. För jämförelse, plotta även z_1 som en funktion av tiden för $k_{1,ref}$ och $k_{2,ref}$. Hur stämmer resultatet överens med uppgiften att öka komforten?

Till hjälp finns funktionsfilen `transfer_functions.m` som innehåller en funktion som tar fjäderkonstanterna k_1, k_2 samt konstanterna c_k, c_s som inparametrar. Som utdata ges en kolonnvektor som innehåller $[F_k(k_1, k_2, c_k) \;\; F_s(k_1, k_2, c_s)]^T$, där

$$F_k(k_1, k_2, c_k) = \tilde{T}_k(k_1, k_2) - c_k, \tag{5}$$
$$F_s(k_1, k_2, c_s) = \tilde{T}_s(k_1, k_2) - c_s. \tag{6}$$

Dessutom finns funktionsfilen `Jacobian_transfer_functions.m,` som innehåller en funktion som tar tar fjäderkonstanterna k_1, k_2 som inparametrar och ger Jakobianen $J(k_1, k_2)$ som utdata. Jakobianen ges av

$$J(k_1, k_2) = \begin{bmatrix} \frac{\partial F_k}{\partial k_1}(k_1, k_2) & \frac{\partial F_k}{\partial k_2}(k_1, k_2) \\ \frac{\partial F_s}{\partial k_1}(k_1, k_2) & \frac{\partial F_s}{\partial k_2}(k_1, k_2) \end{bmatrix}. \tag{7}$$

Funktionsanropet kan alltså se ut så här:

```
J = Jacobian_transfer_functions(k1,k2);
```

där `k1`, `k2` är skalärer och `J` är en matris av storlek 2×2.

Tips: Notera att c_k och c_s är konstanter vars värde du behöver sätta innan du applicerar Newtons metod på problemet i (5)-(6). Uppgiftstexten till Uppgift 5 bestämmer vad de skall vara i det här fallet.

Om ode45

Ode45 är en så kallad inbäddad Runge Kutta-metod som använder två Runge Kutta-metoder av olika noggrannhetsordning. I varje tidssteg uppskattas det lokala trunkeringsfelet som skillnaden mellan lösningarna som beräknats från de två metoderna. Till skillnad från Euler framåt och Runge Kutta 4 som båda använder ett fixt tidssteg så bestäms steglängden på tidsteget i en inbäddad metod adaptivt för att uppfylla en viss feltolerens. Anropet till ode45 i Matlab ser ut så här:

```
[t,y] = ode45(odefun,tspan,y0,options);
```

utparametrarna t och y är vektorn med sparade tidsvärden och lösningsvärden vid motsvarande tider. Inparametrarna odefun är ett funktionshandtag, tspan en vektor med start- och sluttid, y0 en vektor med initialpositionerna, och options innehåller specifika önskemål för simuleringen. Till exempel går det att ange en relativ feltolerans på 10^{-6} och en absolut feltolerens på 10^{-7} genom att specifiera

```
options = odeset('RelTol',1e-6,'AbsTol',1e-7);
```

innan anropet till ode45. Det finns också en parameter 'Refine' för alla Matlabs ODE lösare. Om den har värdet 1 så returnerar lösaren de tidsvärden t som använts i beräkningen, och motsvarande y-värden. Med anropet ovan så är då t(2)-t(1) storleken på h i första steget, t(3)-t(2) storleken i andra etc. ode45 har dock defaultvärdet på 'Refine' satt till 4, vilket gör att den returnerar extra värden inom varje beräkningssteg. Sätt denna parameter på samma sätt som feltoleranserna:

```
options = odeset('RelTol',1e-6,'AbsTol',1e-7, 'Refine',1);
```

Det går också att utelämna options i anropet till ode45, då sätts alla parametrar till sina sk defaultvärden: den relativa feltoleransen till 10^{-3}, den absoluta feltolerensen till 10^{-6} och parametern 'Refine' till 4.

Tips: När funktionen odefun tar inparametrar behöver anropet till ode45 modifieras. Antag att vi har en funktion odefun.m med extra inparametrar A och B, då görs anropet på följande sätt:

```
[t,y] = ode45(@(t,y) odefun(t,y,A,B),tspan,y0,options);
```