Ordinära differentialekvationer

1 Inledning

Vi skall se på *begynnelsevärdesproblem* för första ordningens differentialekvation

$$\begin{cases} u' = f(t,u),\ a \le t \le b \\ u(a) = u_a \end{cases}$$

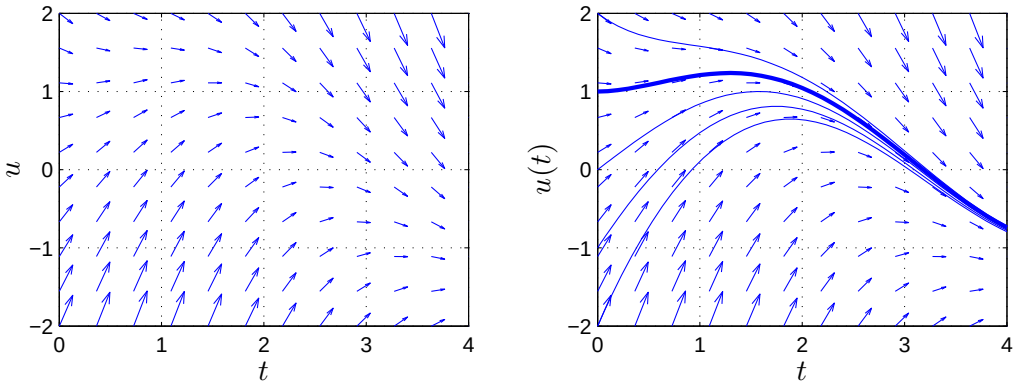
där f en given funktion och u_a en given konstant.

Som exempel tar vi problemet

$$\begin{cases} u' = -u(t) + \sin(t) + \cos(t),\ 0 \le t \le 4 \\ u(0) = u_0 \end{cases}$$

med analytisk (exakt) lösning $u(t) = \sin(t) + u_0\,\mathrm{e}^{-t}$.

I den vänstra figuren nedan har vi ritat *riktningsfältet* och i den högra lösningskurvorna för några olika värden på u_0. Vi ser hur lösningskurvorna följer riktningsfältet.



Metoder för att beräkna numeriska (approximativa) lösningar till differentialekvationer bygger på idén att försöka följa riktningsfältet så *noggrant* och *effektivt* som möjligt.

2 Differensmetoder

Vi skall approximera lösningen $u(t)$ till differentialekvationen på ett nät $t_n = a + nh$ för $n = 0, 1, \cdots, N$, med steglängden $h = \frac{b-a}{N}$.

Låter vi u_n beteckna approximationen av $u(t_n)$ och ersätter $u'(t_n)$ med en *framåtdifferenskvot* $D_+u(t_n)$ så gäller

$$D_+u(t_n) = \frac{u(t_{n+1}) - u(t_n)}{h} \approx u'(t_n) = f(t_n, u(t_n)) \Rightarrow$$
$$u(t_{n+1}) \approx u(t_n) + hf(t_n, u(t_n))$$

Detta ger **Eulers framåtmetod**

$$u_{n+1} = u_n + hf(t_n, u_n)$$

Utgående från begynnelsevärdet försöker metoden följa riktningsfältet med korta steg.

Metoden är *explicit* eftersom alla värden i högerledet är kända.

Ersätter vi $u(t_{n+1})$ med en *bakåtdifferenskvot* $D_-u(t_{n+1})$ får vi

$$D_-u(t_{n+1}) = \frac{u(t_{n+1}) - u(t_n)}{h} \approx u'(t_{n+1}) = f(t_{n+1}, u(t_{n+1})) \Rightarrow$$

$$u(t_{n+1}) \approx u(t_n) + hf(t_{n+1}, u(t_{n+1}))$$

Detta ger **Eulers bakåtmetod**

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$$

Metoden är *implicit* eftersom u_{n+1}, som är obekant, finns med även i f. För att ta ett steg måste man normalt lösa en icke-linjär ekvation

$$g(z) = z - u_n - hf(t_{n+1}, z) = 0$$

med t.ex. Newtons metod.

Vi kan också *integrera* differentialekvationen från t_n till $t_{n+1} = t_n + h$ och får

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t, u(t))\, dt$$

och kan sedan approximera integralen på olika sätt.

Med *vänster* och *höger rektangelregel* får vi Eulers framåt- respektive bakåtmetod och med *trapetsregeln* får vi den implicita **trapetsmetoden**

$$u_{n+1} = u_n + \frac{h}{2}(f(t_n, u_n) + f(t_{n+1}, u_{n+1}))$$

Om vi i denna metod ersätter u_{n+1} i högerledet med en Euler framåt approximationen får vi **Heuns metod**

$$u_{n+1} = u_n + \frac{h}{2}(f(t_n, u_n) + f(t_n + h, u_n + hf(t_n, u_n)))$$

som är en *explicit* metod.

Metoderna vi sett på är alla *konvergenta*, dvs. tar vi tillräckligt liten steglängd kan vi få godtyckligt bra approximation på ett ändligt intervall. För Euler metoderna gäller att om vi halverar steglängden så (ungefär) halveras felet i approximationen. För trapetsmetoden och Heuns metod gäller att om vi halverar steglängden så delas felet i approximationen med (ungefär) fyra.

Ett begynnelsevärdesproblem som beskriver förlopp eller processer vilka utspelas under tidsintervall av mycket olika storleksordning kallas för *styvt*. T.ex. inom kemisk reaktionsteknik är styva problem vanliga. För styva problem måste implicita metoder används, dvs. av typen Euler bakåtmetoden eller trapetsmetoden. Explicita metoder, som Euler framåtmetoden eller Heuns metod, blir mycket ineffektiva.

3 Rita riktningsfält

Vi skall se hur man kan rita riktningsfält till differentialekvationen

$$\begin{cases} u' = f(t,u),\ a \le t \le b \\ u(a) = u_a \end{cases}$$

Ett riktningsfält består i en samling punkter (t_i, u_j) i tu-planet (ett gitter) där vi i varje punkt ritar en liten pil i den riktning som en lösningskurva $t \mapsto (t, u(t))$ genom punkten har precis i punkten, dvs. en pil i riktningen $(1, u'(t_i)) = (1, f(t_i, u_j))$. Vi skalar pilarna så att vi får en tydlig bild. (För korta pilar och inget syns, för långa pilar och bilden blir grötig.)

Som exempel ritar vi riktningsfältet till det inledande begynnelsevärdesproblemet

$$\begin{cases} u' = -u(t) + \sin(t) + \cos(t),\ 0 \le t \le 4 \\ u(0) = u_0 \end{cases}$$

Först bildar vi ett gitter (grid) med `meshgrid`. I matriserna `T` och `U` kommer vi ha gittrets koordinater. Sedan bildar vi en matris `DT` med ettor, som är första koordinaterna i pilarna, och en matris `DU`, som är andra koordinaterna i pilarna. Slutligen ritar vi ut pilarna med `quiver`, där talet 0.9 är en skalfaktor (lagom långa pilar).

```
>> f=@(t,u)-u+sin(t)+cos(t);
>> a=0; b=4;
>> t=linspace(a,b,12); u=linspace(-2,2,12);
>> [T,U]=meshgrid(t,u);
>> DT=ones(size(T)); DU=f(T,U);
>> quiver(T,U,DT,DU,0.9)
```

Som resultat får vi vänstra figuren på första sidan.

Uppgift 1. Rita riktningsfältet till följande begynnelsevärdesproblem

$$\begin{cases} u' = -u(t) + \sin(5t) + \cos(2t),\ 0 \le t \le 5 \\ u(0) = u_0 \end{cases}$$

4 Eget program i MATLAB

Vi skall nu beskriva hur man kan beräkna en numerisk lösning till begynnelsevärdesproblemet

$$\begin{cases} u' = f(t,u),\ a \le t \le b \\ u(a) = u_a \end{cases}$$

De metoder vi sett på går alla att använda men vi nöjer vi oss med Euler framåtmetoden, som är den enklaste av dessa metoder.

Vi bildar först ett nät med nodpunkterna $t_n = a + nh,\ n = 0, 1, \cdots, N$, och steglängden $h = \frac{b-a}{N}$. Detta ger en uppdelning av intervallet $a \le t \le b$ i N stycken lika långa delintervall

$$a = t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} < \cdots < t_{N-1} < t_N = b$$

Vi beräknar sedan en approximativ lösning enligt

$$U(t_0) = u_a$$
$$U(t_{n+1}) = U(t_n) + hf(t_n, U(t_n)).$$

Genom att förbinda punkterna $(t_n, U(t_n))$ med räta linjer får vi en graf och funktionen $U(t)$ blir definierad också mellan beräkningsnoderna t_n.

I MATLAB måste $U(t_n)$ representeras av en vektor `U` med N komponenter. Med andra ord, `U(n)` skall innehålla approximationen av $u(t_n)$ för beräkningsnoden (tidpunkten) t_n.

Vi ser på vårt inledande exempel och tar $u(0) = 1$. Så här enkel blir MATLAB-koden

```
>> f=@(t,u)-u+sin(t)+cos(t);
>> a=0; b=4; ua=1;
>> N=100; h=(b-a)/N;
>> t=a+(0:N)*h; U=zeros(size(t));
>> U(1)=ua;
>> for n=1:N
       U(n+1)=U(n)+h*f(t(n),U(n));
   end
>> plot(t,U)
```

Resultatet ser vi i högra figuren på första sidan, där vi även ritat upp lösningar för några andra begynnelsevärden.

Uppgift 2. Vi ser återigen på begynnelsevärdesproblemet

$$\begin{cases} u' = -u(t) + \sin(5t) + \cos(2t),\ 0 \le t \le 5 \\ u(0) = 2 \end{cases}$$

Lös problemet med Euler framåtmetoden. Rita en graf av lösningen i en figur som dessutom innehåller riktningsfältet.

Uppgift 3. Skriv ett program `min_ode` med anropet `[t,U]=min_ode(f,I,ua,h)` som löser begynnelsevärdesproblemet med Euler framåtmetoden. Använd det progamskal du finner på studiohemsidan. In- och ut-variablerna förklaras i programskalet.

Uppgift 4. Testa programmet på följande begynnelsevärdesproblem. För varje exempel måste du skriva en funktionsfil av typen `function y=funk(t,u)` för beskrivning av derivatan. Lös först begynnelsevärdesproblemet analytiskt (dvs. som en formel med penna och papper). Rita både den analytiska lösningen u och den approximativa lösningen U i samma figur.

(a). $\begin{cases} u'(t) = t^2,\ 1 \le t \le 3 \\ u(1) = 1 \end{cases}$

(b). $\begin{cases} u'(t) = u(t),\ 0 \le t \le 2 \\ u(0) = 1 \end{cases}$

(c). $\begin{cases} u'(t) = -t\,u(t),\ 0 \le t \le 3 \\ u(0) = 1 \end{cases}$

(d). $\begin{cases} u'(t) = -5u(t),\ 0 \le t \le 1 \\ u(0) = 2 \end{cases}$

5 Färdiga program i MATLAB

Det finns färdiga funktioner i MATLAB för att lösa differentialekvationer, en sådan funktion är `ode45` för "vanliga" begynnelsevärdesproblem, en annan är `ode15s` för styva problem. Med `ode45` kan vi beräkna en lösning till vårt inledande exempel för t.ex. $u(0) = 1$ enligt

```
>> a=0; b=4; ua=1;
>> f=@(t,u)-u+sin(t)+cos(t);
>> [t,U]=ode45(f,[a b],ua);
>> plot(t,U)
```

Här blir `t` en kolonnvektor, med t-värden mellan a och b, där lösningen är beräknad och `U` är en kolonnvektor med den beräknade lösningen för de olika t-värdena.

Uppgift 5. För en sista gång ser vi på begynnelsevärdesproblemet

$$\begin{cases} u' = -u(t) + \sin(5t) + \cos(2t),\ 0 \le t \le 5 \\ u(0) = 2 \end{cases}$$

Lös problemet med `ode45`. Rita en graf av lösningen i en figur som även innehåller riktningsfältet. Beräkna sedan en lösning med `min_ode` och rita upp den i samma figur. Använd olika färg för de olika graferna. Ser du någon skillnad mellan kurvorna, ta då och minska steget h i `min_ode` och rita ut även den nya lösningen.

6 System av differentialekvationer

Med MATLAB kan vi lika lätt lösa system av differentialekvationer

$$\begin{cases} \mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t)), \quad a \le t \le b \\ \mathbf{u}(a) = \mathbf{u}_a \end{cases}$$

där

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{u}) = \begin{bmatrix} f_1(t, u_1(t), \cdots, u_m(t)) \\ \vdots \\ f_m(t, u_1(t), \cdots, u_m(t)) \end{bmatrix}, \quad \mathbf{u}_a = \begin{bmatrix} u_{a1} \\ \vdots \\ u_{am} \end{bmatrix}$$

Skillnaden är att nu blir `U` en matris och vi finner $u_k(t)$ (för olika tidpunkter t_i) som kolonn `U(:,k)`, dvs. kolonn nr k i `U`.

Som exempel på ett system av ekvationer tar vi: *Populationsdynamik* – Vi betraktar population av bytesdjur (kaniner) som lever tillsammans med en population rovdjur (rävar). Låt $u_1(t)$ respektive $u_2(t)$ beteckna antalet kaniner respektive rävar vid tiden t.

En enkel matematisk modell för populationernas utveckling ges av Volterra-Lotka-ekvationerna:

$$\begin{cases} u_1'(t) = a\, u_1(t) - b\, u_1(t)\, u_2(t) \\ u_2'(t) = -c\, u_2(t) + d\, u_1(t)\, u_2(t) \end{cases}$$

Koefficienterna a, b, c, d är positiva. Termen $a\, u_1(t)$ representerar netto-födelse-dödstalet i en ensam kaninpopulation. Termen $-c\, u_2(t)$ är motsvarande för rävarna. Termen $-b\, u_1(t)\, u_2(t)$ är antalet

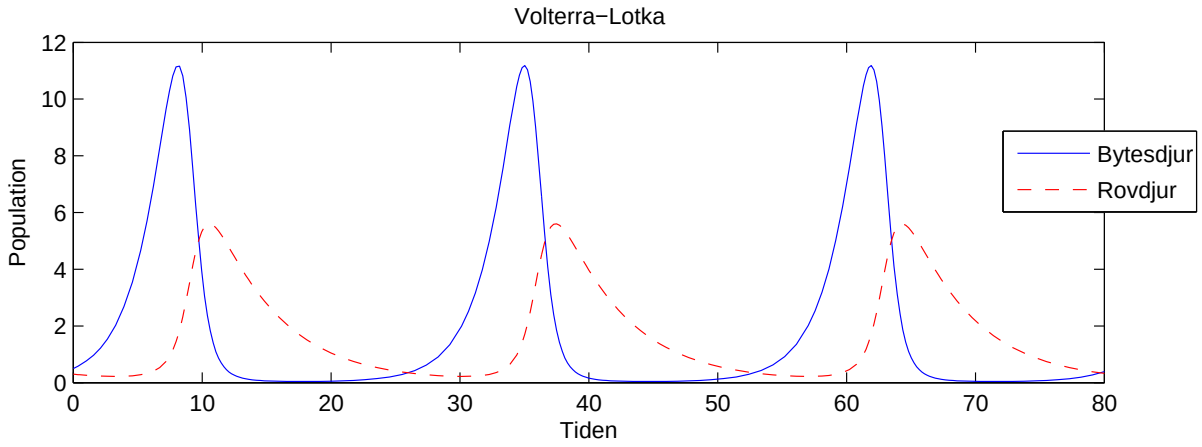
kaniner som blir uppätna per tidsenhet. Termen $d\,u_1(t)\,u_2(t)$ är antalet rävar per tidsenhet som överlever på grund av tillgång på föda. Observera teckenkombinationen i ekvationerna. Vad blir lösningen om populationerna är ensamma ($b = d = 0$)?

Vi beskriver högerledet i differentialekvationen med en funktion

```
function f=volterra(t,u)
    a=0.5; b=0.3; c=0.2; d=0.1;
    f=[ a*u(1)-b*u(1)*u(2)
       -c*u(2)+d*u(1)*u(2)];
```

Vi löser sedan med funktionen `ode45` och ritar upp enligt

```
>> u0=[0.5;0.3];
>> [t,U]=ode45(@volterra,[0 80],u0);
>> plot(t,U(:,1),t,U(:,2),'r--')
>> legend('Bytesdjur','Rovdjur')
>> xlabel('Tiden'), ylabel('Population')
>> title('Volterra-Lotka')
```



Uppgift 6. Lös Volterra-Lotka-ekvationerna med `ode45`. Ändra koefficienterna till $a = 0.5$, $b = 0.3$, $c = 0.2$, $d = 0.05$.

Vi skall se på en typ av "Belousov-Zhabotinsky reaction" (se t.ex. Atkins & Jones). Det vi ser på är den kemiska klockan som är ett något exotiska exempel på ett periodiskt förlopp i en kemisk reaktion. Materialet bygger på Field och Noyes, J. Chem. Phys. 60: 1877-1884 (1974).

För en blandning av

1.25 M	H_2SO_4
0.0125 M	$KBrO_3$
0.001 M	$Ce(NH_4)_2(NO_3)_5$
0.025 M	$CH_2(COOH)_2$

härleder Field och Noyes massbalanser för mellanprodukterna $HBrO_2$, Br^- och Ce^{4+} med dimensionslösa koncentrationer $u_1(t)$, $u_2(t)$ respektive $u_3(t)$;

$$\begin{cases} u_1'(t) = s(u_2(t) - u_1(t)u_2(t) + u_1(t) - q u_1(t)^2), & u_1(0) = 4 \\ u_2'(t) = s^{-1}(u_3(t) - u_2(t) - u_1(t)u_2(t)), & u_2(0) = 1.1 \\ u_3'(t) = w(u_1(t) - u_3(t)), & u_3(0) = 4 \end{cases}$$

där $s = 77.27$, $q = 8.375 \cdot 10^{-6}$ och $w = 0.1610$.

Integrerar man över långa tidsperioder ser man att lösningen är periodisk med en periodlängden på ungefär 300 tidsenheter (48.3 sekunder).

Blandningen kommer att växla färg. Först kommer den vara klar, sedan växlar den till gult, blir klar igen, blir sedan åter gul, och så vidare. Man kan tillsätta en indikator så att färgväxlingen blir mellan blått och rött.

Problemet är styvt så vi använder den styva lösaren `ode15s` i MATLAB.

Uppgift 7. Beräkna och rita upp lösningen under en period. Därefter studerar ni det initiala förloppet genom att beräkna och rita upp lösningen under de 5 första tidsenheterna. Rita ut logaritmerna av koncentrationerna, så att vi ser även små koncentrationer ($^{10}\log$ ges av `log10` i MATLAB). Använd `xlabel`, `ylabel` och `legend` för att göra tydliga grafer.

7 Högre ordningens differentialekvationer

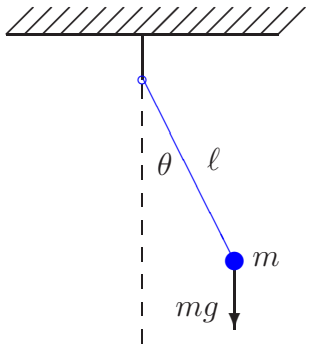
Högre ordningens differentialekvationer kan skrivas om som system av första ordningen. Dessa system kan sedan lösas numeriskt.

Som exempel tar vi åter igen den matematiska pendeln. En masspunkt med massan m hänger i en viktlös smal stav av längden ℓ.

Med beteckningarna i figuren och Newtons andra lag får vi rörelseekvationen

$$m\ell\,\ddot{\theta}(t) = -mg\sin(\theta(t))$$

Vi vill bestämma lösningen för olika begynnelseutslag θ_0, dvs. $\theta(0) = \theta_0$, då vi släpper pendeln från vila, dvs. $\dot{\theta}(0) = 0$.



Om vi låter $\varphi = \dot{\theta}$, dvs. inför vinkelhastigheten, kan ekvationen skrivas

$$\begin{cases} \dot{\theta} = \varphi, & \theta(0) = \theta_0 \\ \dot{\varphi} = -\frac{g}{\ell}\sin(\theta), & \varphi(0) = 0 \end{cases}$$

För att komma till standardform låter vi $u_1 = \theta$ och $u_2 = \varphi$ och får

$$\begin{cases} u_1' = u_2, & u_1(0) = \theta_0 \\ u_2' = -\frac{g}{\ell}\sin(u_1), & u_2(0) = 0 \end{cases}$$

Nu har vi standardformen

$$\begin{cases} \mathbf{u}' = \mathbf{f}(t,\mathbf{u}) \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \mathbf{f}(t,\mathbf{u}) = \begin{bmatrix} u_2 \\ -\frac{g}{\ell}\sin(u_1) \end{bmatrix}, \quad \mathbf{u}_0 = \begin{bmatrix} \theta_0 \\ 0 \end{bmatrix}$$

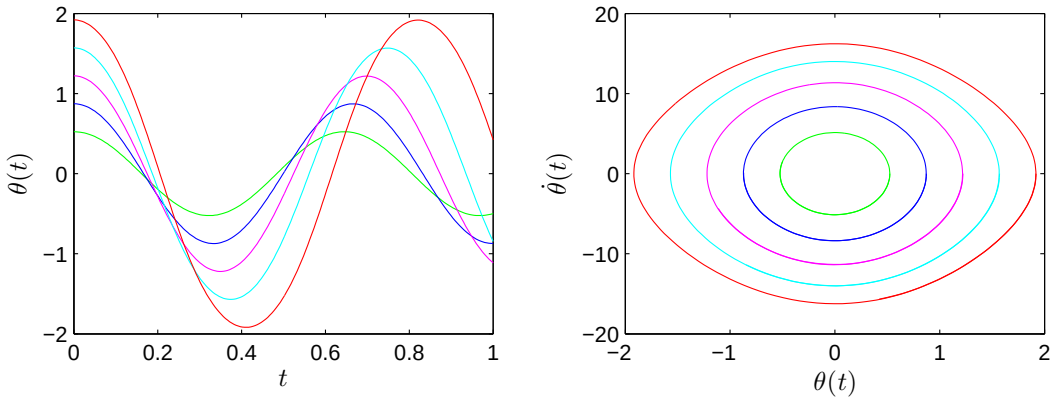
Detta system kan vi lösa på samma sätt som i förra avsnittet.

Vi beskriver differentialekvationen med funktionen

```
function f=pendel(t,u,g,l)
    f=[ u(2)
       -g/l*sin(u(1))];
```

Följer lösningskurvorna med `ode45` för några olika begynnelseutslag och ritar en bild som visar lösningarna $t \mapsto (t, \theta(t))$ och fasporträtten $t \mapsto (\theta(t), \dot{\theta}(t))$ för de olika begynnelseutslagen.

```
g=9.81; l=0.1;
theta0=[30:20:110]*pi/180; col=['g','b','m','c','r'];
tspan=linspace(0,1,200);
for k=1:length(theta0)
    u0=[theta0(k);0];
    [t,U]=ode45(@(t,u)pendel(t,u,g,l),tspan,u0);
    subplot(1,2,1), plot(t,U(:,1),col(k)), hold on
    subplot(1,2,2), plot(U(:,1),U(:,2),col(k)), hold on
end
subplot(1,2,1), hold off
xlabel('$t$','interpreter','latex','fontsize',12)
ylabel('$\theta(t)$','interpreter','latex','fontsize',12)
subplot(1,2,2), hold off
xlabel('$\theta(t)$','interpreter','latex','fontsize',12)
ylabel('$\dot{\theta}(t)$','interpreter','latex','fontsize',12)
```



Från figuren ser vi att periodlängden ökar med ökande begynnelseutslag.

Uppgift 8. En dämpad matematisk pendel beskrivs av

$$\begin{cases} m\ell\,\ddot{\theta}(t) = -mg\sin(\theta(t)) - c\ell\,\dot{\theta}(t), \quad t \geq 0 \\ \theta(0) = \theta_0, \quad \dot{\theta}(0) = 0 \end{cases}$$

där c är dämpningskonstanten. Lös problemet för $\ell = 0.1$, $m = 0.1$ och $c = 0.2$ och några olika begynnelseutslagsvinklar. Använd `ode45`.

När vi gjorde figuren ovan i MATLAB skrev vi formlerna med s.k. LaTeX-kod. Så brukar matematiker skriva formler för att de skall bli snygga. Men vi får vi se det som överkurs.