Ordinära differentialekvationer (ODE) 1[1]

I förra datorövningen löste vi begynnelsvärdesproblem av formen

$$\begin{aligned} &u'(x) = f(x), \quad x \in [0, b] \quad (b > 0) \\ &u(0) = u_0. \end{aligned}$$

Här beror f bara på x. Lösningen konstruerades med hjälp av Eulers metod. Den kan också skrivas analytiskt med en formel

$$u(x) = u_0 + \int_0^x f(y)\, dy$$

enligt differential- och integralkalkylens fundamentalsats (Fundamental Theorem of Calculus).

Nu ska vi lösa differentialekvationer av allmän form

$$\begin{aligned} &u'(x) = f(x, u(x)), \quad x \in [0, b] \quad (b > 0) \\ &u(0) = u_0. \end{aligned}$$

Här beror f både på x och u. Det kan också vara ett system av differentialekvationer; då är både u och f kolonnvektorer.

Vi börjar med att introducera ett viktigt specialfall, nämligen begynnelsevärdesproblemet som hör ihop med de trigonometriska funktionerna cos och sin.

1.1 Trigonometriska funktioner

Vi vet att de trigonometriska funktionerna uppfyller

$$\begin{aligned} &D \sin(t) = \cos(t),\ D^2 \sin(t) = -\sin(t),\ \sin(0) = 0, \\ &D \cos(t) = -\sin(t),\ D^2 \sin(t) = -\cos(t),\ \cos(0) = 1. \end{aligned}$$

(Notera att vi nu går över till att skriva den oberoende variabeln som t istället för x. Anledningen till detta är att den ofta är tiden i ett begynnelsevärdesproblem.) Det betyder att $u(t) = \sin(t)$ löser begynnelsevärdesproblemet

$$\begin{aligned} &u''(t) = -u(t), \quad t \in [0, b], \quad (b > 0) \\ &u(0) = 0,\ u'(0) = 1, \end{aligned}$$

och att $u(t) = \cos(t)$ löser begynnelsevärdesproblemet

$$\begin{aligned} &u''(t) = -u(t), \quad t \in [0, b], \quad (b > 0) \\ &u(0) = 1,\ u'(0) = 0, \end{aligned}$$

[1] 2007–11–17, Stig Larsson, Matematiska vetenskaper Chalmers tekniska högskola

Mer allmänt har vi

$$u''(t) = -c^2 u(t), \quad t \in [0, b], \quad (b > 0)$$
$$u(0) = u_0, \; u'(0) = u_1,$$

med lösningen

$$u(t) = u_0 \cos(ct) + \frac{u_1}{c} \sin(ct). \tag{1}$$

Verifiera detta genom att derivera och genom att sätta in $t = 0$!

Denna differentialekvation är av andra ordningen. För att kunna använda algoritmen från förra labben skriver vi om ekvationen som *ett system av två differentialkevationer av första ordningen*. Vi inför två nya variabler

$$w_1 = u, \; w_2 = u'.$$

Vi deriverar

$$w_1' = u' = w_2,$$
$$w_2' = u'' = -c^2 u = -c^2 w_1.$$

Begynnelsevillkoren blir $w_1(0) = u_0$, $w_2(0) = u_1$. Vi ser att w_1 och w_2 löser begynnelsevärdesproblemet

$$\begin{cases} w_1' = w_2, \\ w_2' = -c^2 w_1, \end{cases} \quad \begin{cases} w_1(0) = u_0, \\ w_2(0) = u_1. \end{cases}$$

Vi skriver detta på matrisform:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}' = \begin{bmatrix} w_1' \\ w_2' \end{bmatrix} = \begin{bmatrix} w_2 \\ -c^2 w_1 \end{bmatrix}, \quad \begin{bmatrix} w_1(0) \\ w_2(0) \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}.$$

(Vi skall i en senare kurs se att med beteckningarna

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -c^2 & 0 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad \boldsymbol{w}_0 = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix},$$

kan högerledet kan skrivas med matrismultiplikation

$$\begin{aligned} \boldsymbol{w}'(t) &= \boldsymbol{A}\boldsymbol{w}(t), \quad t \in [0, b], \\ \boldsymbol{w}(0) &= \boldsymbol{w}_0, \end{aligned} \tag{2}$$

men det behöver vi inte kunna i den här kursen.)

1.2 Allmänt system av ODE

Vi har nu diskuterat differentialekvationer

$$u'(x) = f(x, u(x))$$

i några specialfall där f bara beror av x (`myprim.m`):

$$u'(x) = f(x)$$

och bara beror av u

$$u'(x) = u(x)$$

och även system av denna senare typ

$$\begin{aligned} u_1'(x) &= u_2(x), \\ u_2'(x) &= -u_1(x). \end{aligned}$$

Vi ska nu behandla differentialekvationer där $f(x, u)$ är en allmän funktion av både x och u. Till exempel,

$$u'(x) = -xu(x),\ x \in [0, 3]; \quad u(0) = 1,$$

med analytisk lösning $u(x) = \exp(-x^2/2)$. Verifiera detta! Här ges alltså $f : \mathbf{R}^2 \to \mathbf{R}$ av formeln $f(x, s) = -xs$. När vi sätter in i högerledet av differentialekvationen ersätter vi den andra variabeln s med $s = u(x)$. Vi ska även studera system av sådana ekvationer.

Vi kommer då till ett allmänt system av n stycken differentialekvationer:

$$\begin{aligned} &u_1'(t) = f_1(t, u_1(t), \ldots, u_n(t)), \quad t \in [a, b], \\ &\vdots \\ &u_n'(t) = f_n(t, u_1(t), \ldots, u_n(t)), \quad t \in [a, b], \end{aligned}$$

tillsammans med lika många begynnelsevillkor:

$$\begin{aligned} &u_1(a) = u_{a1} \\ &\vdots \\ &u_n(a) = u_{an}. \end{aligned}$$

Vi skriver dessa på matrisform:

$$\boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{bmatrix}, \quad \boldsymbol{u}_a = \begin{bmatrix} u_{a1} \\ \vdots \\ u_{an} \end{bmatrix}, \quad \boldsymbol{f}(t, u) = \begin{bmatrix} f_1(t, u_1, \ldots, u_n) \\ \vdots \\ f_n(t, u_1, \ldots, u_n) \end{bmatrix},$$

och får följande begynnelsevärdesproblem

$$\begin{aligned} \boldsymbol{u}'(t) &= \boldsymbol{f}(t, \boldsymbol{u}(t)), \quad t \in [a, b], \\ \boldsymbol{u}(a) &= \boldsymbol{u}_a. \end{aligned} \tag{3}$$

För att lösa detta använder vi samma algoritm som tidigare (Eulers metod). Vi börjar med att dela in intervallet $[a, b]$ i N stycken delintervall av längden $h = (b - a)/N$:

$$
\begin{aligned}
&a = t_0 < t_1 < t_2 < \dots t_{i-1} < t_i < \dots < t_{N-1} < t_N = b, \\
&t_i = a + hi, \ h = (b - a)/N = t_i - t_{i-1}.
\end{aligned}
$$

Vi beräknar nu en approximativ lösning enligt

$$
\begin{aligned}
&\boldsymbol{U}(t_0) = \boldsymbol{u}_a \\
&\boldsymbol{U}(t_i) = \boldsymbol{U}(t_{i-1}) + h\boldsymbol{f}(t_{i-1}, \boldsymbol{U}(t_{i-1})).
\end{aligned}
$$

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

Man kan visa att $\boldsymbol{U}(t)$ konvergerar mot en unik lösning $\boldsymbol{u}(t)$ till (3) då antalet delintervall $N \to \infty$. Beviset är upplagt enligt samma princip som för bisektionsalgoritmen och fixpunktsiterationen men är ganska långt och lite för svårt och utlämnas därför. Observera att $\boldsymbol{u}(t)$ normalt inte går att skriva i termer av kända ('elementära') funktioner.

Implementering i Matlab

Algoritmen är

$$
\text{initiera:} \begin{cases} t_0 = a \\ \boldsymbol{U}(t_0) = \boldsymbol{u}_a \end{cases}
$$

$$
\text{uppdatera:} \begin{cases} \text{while } t_i < b \\ t_i = t_{i-1} + h \\ \boldsymbol{U}(t_i) = \boldsymbol{U}(t_{i-1}) + h\boldsymbol{f}(t_{i-1}, \boldsymbol{U}(t_{i-1})). \end{cases}
$$

Övning 1. Skriv ett program `myode.m` med anropet `[t,U]=myode(f,I,ua,h)` som löser begynnelsevärdesproblemet (3). Använd programskalet `myode.m`. (`facit`).

I Matlab sparas $\boldsymbol{U}(t_i)$ som i:te kolonnen `U(:,i)` i matrisen `U`, den blir av typ $n \times (N + 1)$. Matrisen skapas kolonnvis. Till exempel blir initieringen `U(:,1)=u0`. När beräkningen är klar bör du transponera matriserna `t` och `U` bland annat för att plottning ska fungera smidigt.

Det finns nu två sätt att plotta lösningen. Antag för enkelhets skull att vi har ett system av två ekvationer, dvs $n = 2$.

1. Vi kan plotta de två *lösningskurvorna* $y = U_1(t)$, $y = U_2(t)$ med kommandot `plot(t,U)`.

2. Vi kan plotta U_2 mot U_1 med kommandot `plot(U(:,1),U(:,2))`. Sådana kurvor med olika startpunkter bildar ett *fasporträtt* för differentialekvationssystemet.

Prova programmet på följande begynnelsevärdesproblem. För varje exempel måste du skriva en funktionsfil av typen

```
function y=funk(t,u)
```

Plotta den numeriska lösningen tillsammans med den analytiska lösningen (om du känner den analytiska).

$$(a) \quad \begin{cases} u'(x) = x^2, \quad x \in [1,3], \\ u(1) = 1. \end{cases}$$

$$(b) \quad \begin{cases} u'(t) = cu(t), \quad t \in [0,2], \\ u(0) = u_0. \end{cases}$$

$$(c) \quad \begin{cases} u'' = -c^2 u, \quad t \in [0,10] \\ u(0) = u_0,\ u'(0) = u_1. \end{cases}$$

$$(d) \quad \begin{cases} u'(x) = -xu(x), \quad x \in [0,3], \\ u(0) = 1. \end{cases}$$

analytisk lösning: $u(x) = \exp(-x^2/2)$

$$(e) \quad \begin{cases} u'' + 4u' + 13u = e^{-t}, \quad t \in [0,10] \\ u(0) = 1,\ u'(0) = 2. \end{cases}$$

□

Facit

```
(a)
function y=funka(t,u)
y=t^2;

>> [x,U]=myode(@funka,[1,3],1,1e-3)
>> plot(x,U), hold on
>> fplot(@(x)(1+(x.^3-1)/3),[1,3],'r')
```

```
(b)
function y=funkb(t,u)
c=1;
y=c*u;
```

```
>> u0=1, c=1
>> [t,U]=myode(@funkb,[0,2],u0,1e-3)
>> plot(t,U), hold on
>> fplot(@(t)exp(c*t),[0,2],'r')
```

```
(c)
function y=funkc(t,u)
c=1;
y(1,1)=u(2);
y(2,1)=-c^2*u(1);
% lite kompaktare kan vi skriva det med matrismultiplikation,
% men vi gör inte det i den här kursen:
% A=[0 1;-c^2 0];
% y=A*u;

>> u0=0, u1=1, c=1
>> [t,U]=myode(@funkc,[0,10],[u0;u1],1e-3)
>> figure(1)
>> plot(t,U), hold on
>> fplot(@(t)(u0*cos(c*t)+u1*sin(c*t)/c),[0,10])
>> figure(2)
>> plot(U(:,1),U(:,2))
>> axis equal
```

```
(d)
function y=funkd(t,u)
y=-t*u;

>> [x,U]=myode(@funkd,[0,3],1,1e-3)
>> plot(x,U)
```

```
(e)
function y=funke(t,u)
y(1,1)=u(2);
y(2,1)=-13*u(1)-4*u(2)+exp(-t);

>> [t,U]=myode(@funke,[0,10],[1;2],1e-3)
>> plot(t,U)
```

Övning 2. 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:

$$\begin{aligned} u_1'(t) &= au_1(t) - bu_1(t)u_2(t) \\ u_2'(t) &= -cu_2(t) + du_1(t)u_2(t) \end{aligned} \tag{4}$$

Koefficienterna a, b, c, d är positiva. Termen $au_1(t)$ representerar netto-födelse-dödstalet i en ensam kaninpopulation. Termen $-cu_2(t)$ är motsvarande för rävarna. Termen $-bu_1(t)u_2$ är antalet kaniner som blir uppätna per tidsenhet. Termen $du_1(t)u_2(t)$ är antalet rävar per tidsenhet som överlever på grund av tillgång på föda.

Observera teckenkombinationen i ekvationerna. Vad blir lösningen om populationerna är ensamma ($b = d = 0$)?

Lös Volterra-Lotka-ekvationerna med ditt program `myode.m`. Lämpliga värden är $a = .5, b = 1, c = .2, d = 1$. □

Facit

Med $b = d = 0$ får vi $u_1(t) = u_{01} \exp(at)$, $u_2(t) = u_{02} \exp(-ct)$, dvs kaninpopulationen exploderar och rävarna dör ut.
Matlab funktionsfil:

```
function y=volterra(t,u)
a=.5; b=1; c=.2; d=1;
y=zeros(2,1);
y(1)= a*u(1)-b*u(1)*u(2);
y(2)=-c*u(2)+d*u(1)*u(2);
```

Matlab kommandon:

```
>> [t,U]=myode(@volterra,[0 50],[.5;.3],1e-2);
>> figure(1)
>> plot(t,U)
>> figure(2)
>> plot(U(:,1),U(:,2))
```