Ordinära differentialekvationer

1 Inledning

Vi skall se på *begynnelsevärdesproblem* för första ordningens differentialekvation

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

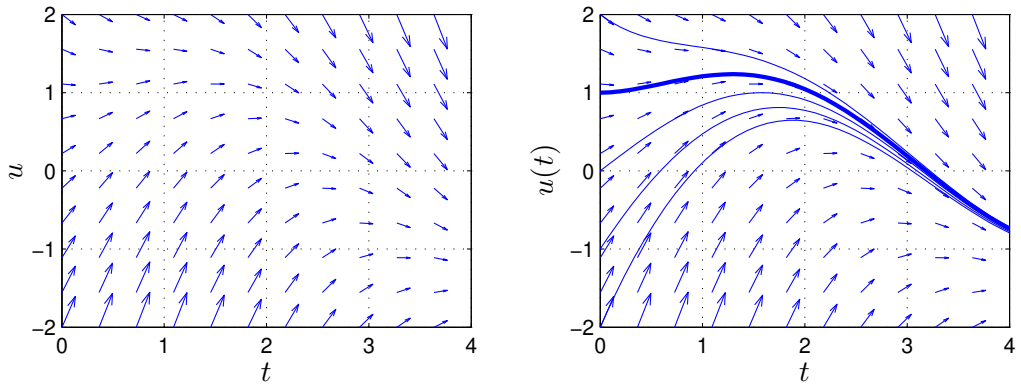
där f en given funktion och u_a en given konstant.

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 *framåt differenskvot* $D_+u(t_n)$ 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.

Det finns många olika sätt att approximera derivator och de ger alla upphov till metoder för att lösa differentialekvationer. Mer komplicerade metoder ger större noggrannhet och effektivitet. Vi nöjer oss dock med den enkla Eulers metod.

3 Riktningsfält

Ett riktningsfält består av en samling punkter (t_i, u_j) i tu-planet (ett gitter). I varje punkt ritar vi en liten pil (eller linje) i den riktning som en lösningskurva $(t, u(t))$ genom punkten har precis i punkten, dvs. en pil i riktningen $(1, u'(t_i)) = (1, f(t_i, u_j))$.

Uppgift 1. Rita riktningsfält till följande begynnelsevärdesproblem

$$u' = -u(t) + \sin(5t) + \cos(2t),\ 0 \le t \le 5$$

Använd funktionen `riktningsfaelt` på studiohemsidan för att rita riktningfält. Titta gärna på programkoden, den är ganska lätt att förstå.

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

$$\begin{aligned} U(t_0) &= u_a \\ U(t_{n+1}) &= U(t_n) + hf(t_n, U(t_n)). \end{aligned}$$

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)`. 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 s.k. *styva* problem. (Ett styvt begynnelsevärdesproblem beskriver förlopp eller processer vilka utspelas under tidsintervall av mycket olika storleksordning. T.ex. inom kemisk reaktionsteknik är styva problem vanliga.) 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

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 ($b = d = 0$)?

Vår differentialekvation har formen

$$\begin{cases} \boldsymbol{u}' = \boldsymbol{f}(t, \boldsymbol{u}) \\ \boldsymbol{u}(0) = \boldsymbol{u}_0 \end{cases}$$

där

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\quad \boldsymbol{f}(t, \boldsymbol{u}) = \begin{bmatrix} a\,u_1 - b\,u_1\,u_2 \\ -c\,u_2 + d\,u_1\,u_2 \end{bmatrix},\quad \boldsymbol{u}_0 = \begin{bmatrix} u_1(0) \\ u_2(0) \end{bmatrix}$$

Vi skall använda `ode45` för att beräkna en numerisk lösning.

Först beskriver vi högerledet i differentialekvationen med en funktion

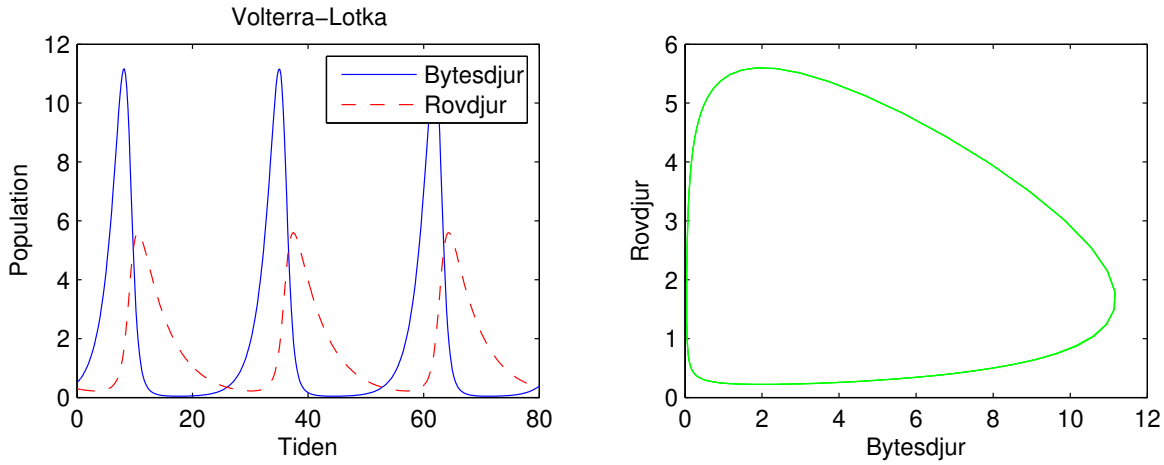
```
function f=volterra(t,u)
    a=0.5; b=0.3; c=0.2; d=0.1;
    f=[ a*u(1)-b*u(1)*u(2)
       -c*u(2)+d*u(1)*u(2)];
```

och sedan löser vi med `ode45` och ritar upp enligt

```
>> u0=[0.5;0.3];
>> tspan=linspace(0,80,400);
>> [t,U]=ode45(@volterra,tspan,u0);
>> plot(t,U(:,1),t,U(:,2),'r--')
```

```
>> legend('Bytesdjur','Rovdjur')
>> xlabel('Tiden'), ylabel('Population')
>> title('Volterra-Lotka')
```

Vi får bilden nedan till vänster.



Skillnaden mot i förra avsnittet är att när vi använder `ode45` blir `U` inte en vektor utan en matris. Vi finner $u_1(t)$ och $u_2(t)$ (för olika tidpunkter t_n) som `U(:,1)` respektive `U(:,2)`, dvs. första och andra kolonnen i `U`.

I bilden ovan till höger har vi rita även upp ett s.k. *fasporträtt*, dvs. man ritar antal rovdjur mot antalet bytesdjur.

```
>> plot(U(:,1),U(:,2),'g')
>> xlabel('Bytesdjur'), ylabel('Rovdjur')
```

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

Läs gärna lite mer om Volterra-Lotka-ekvationerna i Stewart avsnitt 9.6. Där ser vi hur man kan modifiera ekvationerna om man vill beskriva hur bytesdjuren kan drabbas vid ett överutnyttjande av resurser (kaninerna får ont om föda)?

Om $b = 0$, dvs. bytesdjuren slipper träffa på rovdjuren, så beskriver ekvationen för bytesdjuren en exponentiell tillväxt $u_1'(t) = au_1(t)$. Det är rimligt att bl.a. mängden föda och levnadsutrymme kommer begränsa populationens tillväxt. Därför skulle man kunna modifiera ekvationen med en begränsningsterm, dvs. samma modifiering som ledde till den logistiska ekvationen (Stewart avsnitt 9.4). Vi får ekvationen

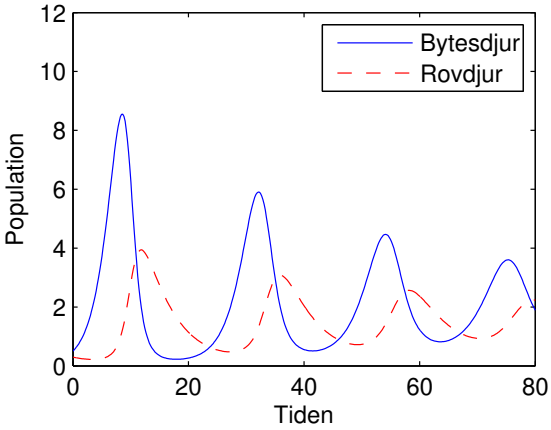
$$u_1'(t) = au_1(t)\left(1 - \frac{u_1(t)}{M}\right)$$

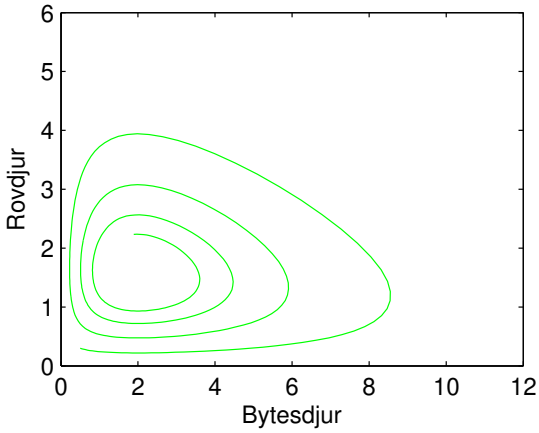
där M är en konstant som ger den största mängden individer ekosystemet kan livnära.

Vi har kommit fram till följande modifierade Volterra-Lotka-ekvationer

$$\begin{cases} u_1'(t) = a\,u_1(t)(1 - u_1(t)/M) - b\,u_1(t)u_2(t) \\ u_2'(t) = -c\,u_2(t) + d\,u_1(t)u_2(t) \end{cases}$$

Dessa löser vi för $M = 30$ och ritar upp resultatet.





Vi ser att vi får en utdämpning med tiden. Lösningsbanan går i spiral in mot ett jämviktsläge.