Ordinära differentialekvationer

1 Inledning

Vi skall se på *begynnelsevärdesproblem* för första ordningens differentialekvation

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

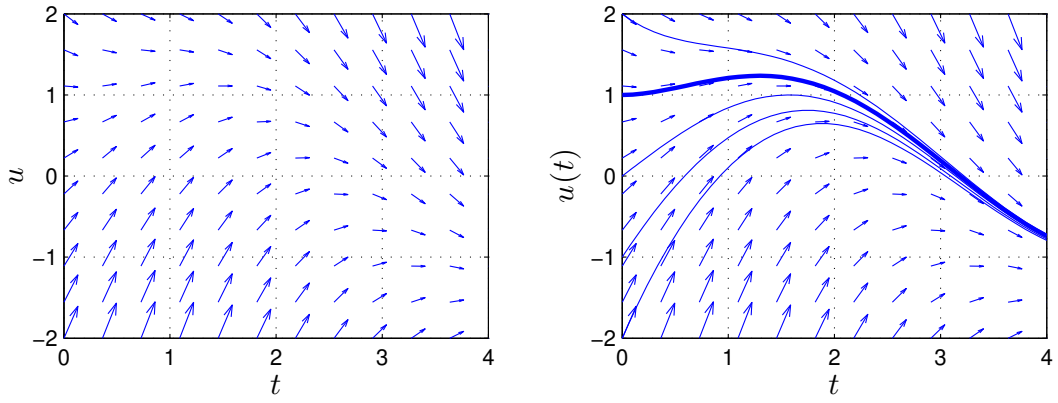
där f en given funktion och u_a en given konstant.

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 differenskvot får vi

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

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

Detta ger **Eulers metod**

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

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 Eget program i MATLAB

Vi skall nu beskriva hur man med Eulers metod kan beräkna en numerisk lösning till begynnelsevärdesproblem

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

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=10; 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)
```

Uppgift 1. Lös följande differentialekvation med begynnelsevillkor

$$\begin{cases} u' = \cos(3t) - \sin(5t)u,\ 0 \le t \le 15 \\ u(0) = 2 \end{cases}$$

med Eulers metod. Tag $h = 0.1, h = 0.01$ och $h = 0.001$ som steglängder. Rita grafer av de olika lösningarna (med olika färger).

Uppgift 2. Skriv en funktion med namnet `min_ode` och anropet `[t,U]=min_ode(f,I,ua,h)` som löser begynnelsevärdesproblem. Du skall använda programskalet `min_ode.m` på MATLAB-hemsidan.

Uppgift 3. 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 upp både den analytiska lösningen u och den approximativa lösningen U i samma figur. Tag $h = 0.1$ och $h = 0.001$ som steglängder.

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

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

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

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

4 Färdiga program i MATLAB

Det finns färdiga funktioner i MATLAB för att lösa differentialekvationer. En sådan funktion är `ode45` och vi använder den för att beräkna en lösning till vårt inledande exempel för t.ex. $u(0) = 1$ enligt

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

Uppgift 4. Lös begynnelsevärdesproblemet i uppgift 1 med `ode45`. Rita upp lösningskurvan och rita även upp, i samma bild, lösningskurvorna beräknade med `min_ode`.

5 Målsättning

Avsikten med laborationen är att få en första inblick i hur man löser begynnelsevärdesproblem för differentialekvationer. Många differentialekvationer, speciellt i samband med tekniska beräkningar, saknar användbara formler för lösningen. Då återstår endast att approximera dessa lösningar. Vi skriver ett litet eget program `min_ode`, där vi prövar Eulers metod. Avslutningsvis bekantar vi oss med `ode45`, ett färdigt program för lösning av begynnelsevärdesproblem i MATLAB.

6 Kommentarer och förklaringar

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 $(t, u(t))$ 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,20); u=linspace(-2,2,20);
>> [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.

7 Lärandemål

Efter denna laboration skall du kunna

- redogöra för idén bakom Eulers metod
- lösa begynnelsevärdesproblem för differentialekvationer med `min_ode` och `ode45`