MATLAB övningsuppgifter

1 Inledning

Vi skall först se hur man beräknar numeriska lösningar till differentialekvationer. Därefter skall vi rita motsvarigheten till en graf för en funktion i två variabler. Avslutningsvis skall vi se lite på några gränsvärden.

2 Ordinära differentialekvationer

Vi skall se på begynnelsevärdesproblem för första ordningens differentialekvation

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

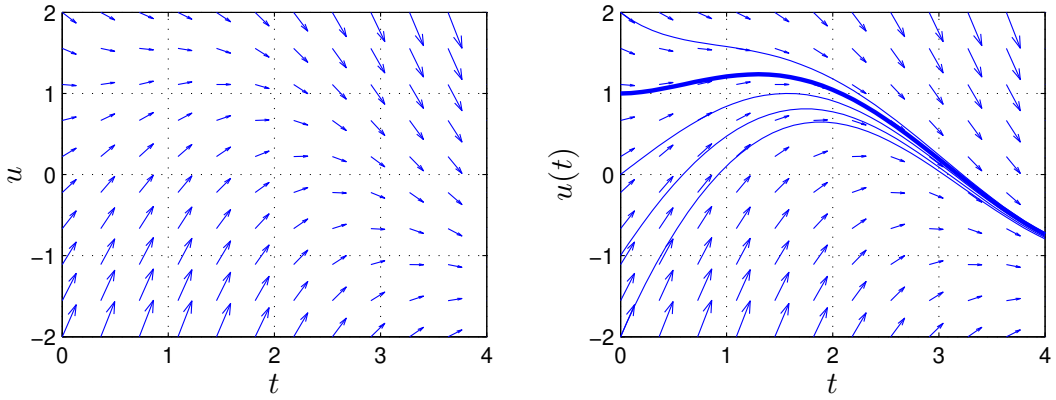
där f en given funktion och u_a en given konstant.

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 $u(t) = \sin(t) + u_0 \exp(-t)$.

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.

Med `ode45` i MATLAB kan vi beräkna en numerisk (approximativ) lösning för t.ex. $u(0) = 1$ enligt

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

Beräkningsmetoderna som `ode45` använder bygger på idén att försöka följa riktningsfältet så noggrannt och effektivt som möjligt.

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

Ersätter vi $u(t_{n+1})$ med en bakåt differenskvot $D_-u(t_{n+1})$ 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 approximation 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.

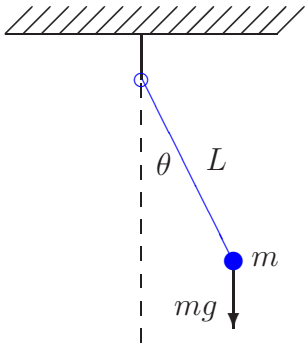
Metoderna vi sett på är alla konvergenta, dvs. tar vi tillräckligt liten steglängd kan vi få godtyckligt bra approximation på ett ändligt intervall. För Euler metoderna gäller att om vi halverar steglängden så (ungefär) halveras felet i approximationen. För trapetsmetoden och Heuns metod gäller att om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

Högre ordningens differentialekvationer

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

Som exempel tar vi 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

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

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 $\omega = \dot{\theta}$, dvs. inför vinkelhastigheten, kan ekvationen skrivas

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

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

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

Nu har vi standardformen

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

Nu skall vi se hur vi löser detta system numeriskt i MATLAB.

Först beskriver vi differentialekvationen med funktionen

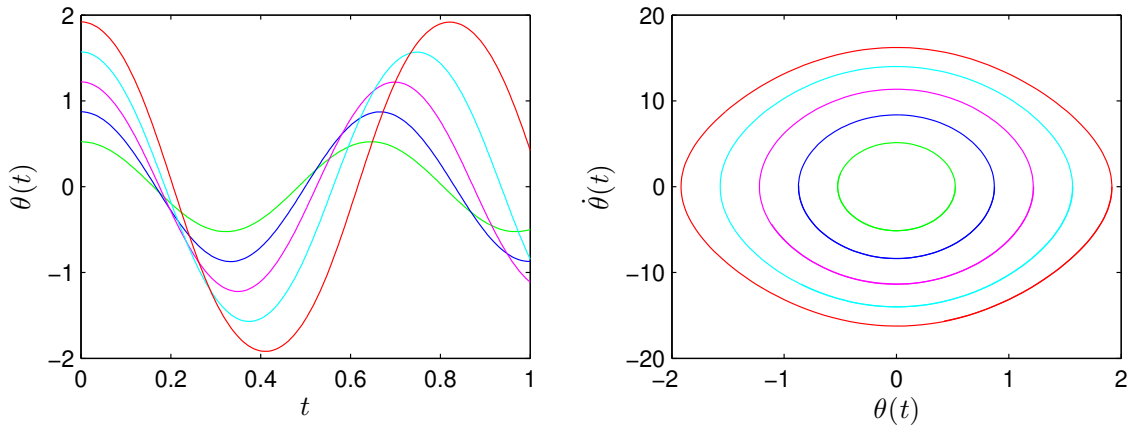
```
function f=pendel(t,u,g,L)
    f=[ u(2)
       -g/L*sin(u(1))];
```

Sedan följer vi 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, något vi sett redan förra läsperioden i en uppgift i "Tillämpningar i matematik med MATLAB".

Uppgift 1. En dämpad matematisk pendel beskrivs av

$$\begin{cases} mL\,\ddot{\theta}(t) = -mg\sin(\theta(t)) - cL\,\dot{\theta}(t), & t \geq 0 \\ \theta(0) = \theta_0, \quad \dot{\theta}(0) = 0 \end{cases}$$

där c är dämpningskonstanten.

Lös problemet för $L = 0.1$, $m = 0.1$ och $c = 0.2$ och några olika begynnelseutslagsvinklar.

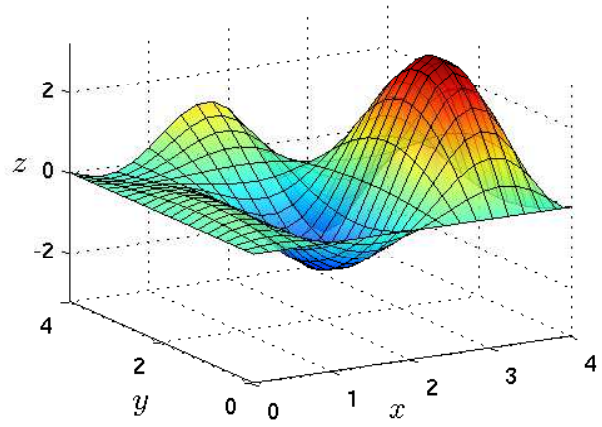
3 Funktioner i två variabler

Vi skall rita funktionsytor och nivåkurvor till funktioner i två variabler. Det blir framförallt i nästa läsperiod, när ni läser flervariabelanalys, som ni ser den stora nyttan med detta, men redan nu kan vi kanske förstå gränsvärden för funktioner i flera variabler lite bättre tack vare funktionsytor.

Funktionsytor

En graf till en funktion i en variabel $f : \mathbb{R} \to \mathbb{R}$ är mängden $\{(x, y) : y = f(x)\}$, dvs. en kurva i planet. En graf till en funktion i två variabler $f : \mathbb{R}^2 \to \mathbb{R}$ är mängden $\{(x, y, z) : z = f(x, y)\}$, dvs. en *yta* i rummet.

Som exempel tar vi $f(x, y) = x\cos(2x)\sin(y)$ över området $0 \le x \le 4$, $0 \le y \le 4$.



Resultatet får vi med kommandot `surf`, vilket är motsvarigheten till `plot` då vi skall rita ytor.

```
>> x=linspace(0,4,30); y=linspace(0,4,30);
>> f=@(x,y)x.*cos(2*x).*sin(y);
>> [X,Y]=meshgrid(x,y);
>> Z=f(X,Y);
>> surf(X,Y,Z)
>> grid on
>> xlabel('x'), ylabel('y'), zlabel('z')
```

Uppgift 2. Rita funktionsytan till funktionen $f : \mathbb{R}^2 \to \mathbb{R}$ där

$$f(x, y) = -xy\exp(-2(x^2 + y^2))$$

över området $-2 \le x \le 2$, $-2 \le y \le 2$.

Hur fungerar det?

Den yta vi skall rita upp består ju av alla punkter $(x, y, f(x, y))$, där $x_{\min} \le x \le x_{\max}$ och $y_{\min} \le y \le y_{\max}$.

Då vi skall rita ytan med `surf` krävs att vi bildar en $m \times n$-matris $\mathbf{Z}$ med elementen

$$z_{ij} = f(x_j, y_i)$$

där $x_{\min} = x_1 < x_2 < \cdots < x_n = x_{\max}$ och $y_{\min} = y_1 < y_2 < \cdots < y_m = y_{\max}$.

Lägg märke till ordningen på indexen, element z_{ij} skall innehålla funktionsvärdet för $x = x_j$ och $y = y_i$.

Stigande x-värden längs rader i $\mathbf{Z}$, dvs. stigande kolonn-index, och stigande y-värden längs kolonner i $\mathbf{Z}$, dvs. stigande rad-index.

Vi går igenom några alternativa lösningar och vi tänker oss att vi i MATLAB redan skapat en funktion `f` och koordinat-vektorer `x` och `y` med `n` respektive `m` element.

Alternativ 1. Vi bildar matrisen `Z` i MATLAB med

```
Z=zeros(m,n);
for i=1:m
    for j=1:n
        Z(i,j)=f(x(j),y(i));
    end
end
surf(X,Y,Z)
```

Alternativ 2. I första alternativet fick hålla reda på var det skulle vara `i` respektive `j`. Med funktionen `meshgrid` skapas två matriser `X` och `Y` så att `X(i,j)` har värdet `x(j)` och `Y(i,j)` har värdet `y(i)`, dvs. indexproblemet är borta.

```
[X,Y]=meshgrid(x,y);
Z=zeros(m,n);
for i=1:m
    for j=1:n
        Z(i,j)=f(X(i,j),Y(i,j));
    end
end
surf(X,Y,Z)
```

Alternativ 3. Den allra smidigaste lösningen får vi då vi låter de nästlade repetitionssatserna i tidigare alternativ ersättas av elementvisa operationer. Dvs. vår funktion `f` måste använda komponetvisa operationer. Vi slipper även initieringen av `Z`.

```
[X,Y]=meshgrid(x,y);
Z=f(X,Y);
surf(X,Y,Z)
```

Nivåkurvor

Ett annat sätt att åskådliggöra en funktion i två variabler $f\colon \mathbb{R}^2 \to \mathbb{R}$ är att rita *nivåkurvor*, dvs. mängderna $\{(x, y) :\ f(x, y) = c\}$, där c är en konstant som anger nivån.

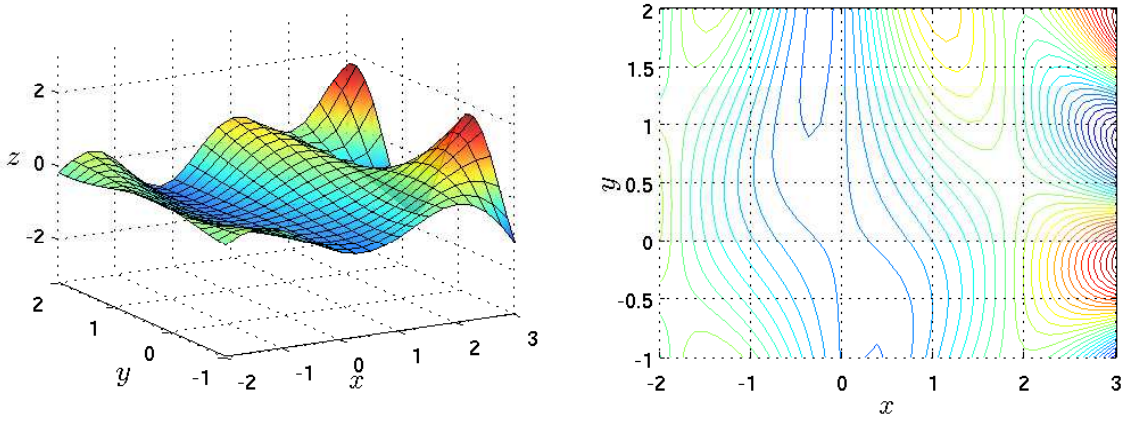
Som exempel tar vi

$$f(x, y) = (\frac{1}{3}x^2 - 1)\sin(1 - xy)$$

över området $-2 \le x \le 3$, $-1 \le y \le 2$.

Vi ritar några nivåkurvor med

```
>> x=linspace(-2,3,40); y=linspace(-1,2,40);
>> f=@(x,y)(0.3*x.^2-1).*sin(1-x.*y);
>> [X,Y]=meshgrid(x,y);
>> Z=f(X,Y);
>> contour(X,Y,Z,30)
```



Vi får en slags topografisk karta av funktionen om vi ser funktionsytan som ett landskap och då blir nivån helt enkelt höjden över havet.

Uppgift 3. Rita nivåkurvor till funktionen från uppgift 2, dvs. $f : \mathbb{R}^2 \to \mathbb{R}$ där

$$f(x, y) = -xy \exp(-2(x^2 + y^2))$$

över området $-2 \leq x \leq 2,\ -2 \leq y \leq 2$.

Gränsvärden

Vi ser på exempel 26, sid 38 i Persson-Böiers, *Analys i flera variabler*. Där konstateras att gränsvärdet av funktionen

$$f(x, y) = \frac{x^4 y^2}{(x^4 + y^2)^2}, \quad (x, y) \neq (0, 0)$$

i $(0, 0)$ inte existerar.

Vi närmar oss $(0, 0)$ längs vägen $y = kx$ och får

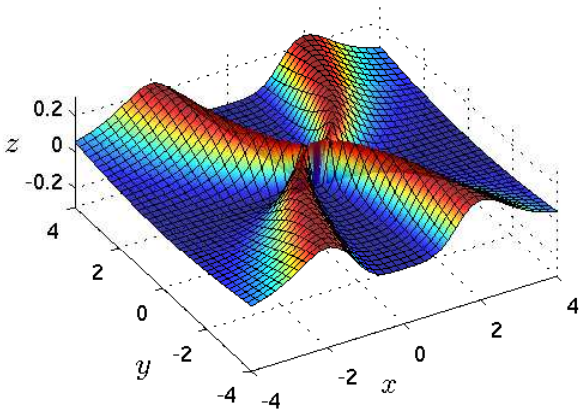
$$f(x, y) = f(x, kx) = \frac{x^4 k^2 x^2}{(x^4 + k^2 x^2)^2} = \frac{k^2 x^2}{(x^2 + k^2)^2} \to 0 \ \text{ då } \ x \to 0$$

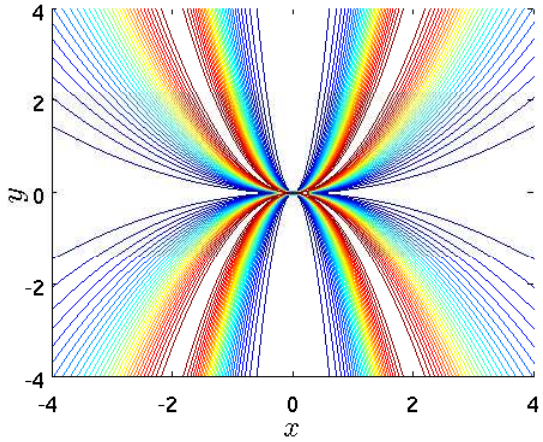
Så om gränsvärdet finns, så skall det vara 0. Men om vi istället närmar oss $(0, 0)$ längs vägen $y = x^2$ får vi

$$f(x, y) = f(x, x^2) = \frac{x^8}{(x^4 + x^4)^2} = \frac{1}{4}$$

Alltså finns inte gränsvärdet.

Vi ritar en bild med funktionsytan och nivåkurvor, som visar hur vi kan gå uppe på åsen på nivån $\frac{1}{4}$ eller nere i plattlandet nära nivån 0.


0.2
0
-0.2
z
4
2
0
-2
-4
y
-4
-2
0
2
4
x


4
2
0
-2
-4
y
-4
-2
0
2
4
x