# Envariabelanalys med Matlab

Under denna kurs kommer vi framförallt att använda Matlab som verktyg i Envariabelanalys. Bl.a skall vi se hur man mha Matlab kan vi rita kurvor i *xy*-planet, rita grafer till funktioner och deras derivator samt lösa ekvationer och beräkna integraler numeriskt. Denna handledning i Matlab innehåller fyra avsnitt, vars innehåll är anpassat till att läsa under olika skeden av Envariabelanalys: I texten ges ofta exempel på kommandorader för att illustrera hur ett visst kommando kan användas. Dessa rader inleds med tecknet >> (som är kommandomarkören i matlab,och ingår alltså inte i själva kommandot). Varje delsnitt innehåller några uppgifter , men det står naturligtvis var och en fritt att experimentera lite på egen hand för att upptäcka Matlab möjligheter och finesser. Inte minst rekommenderas att ni tar hjälp av Matlab vid lösandet av övningsuppgifter i Envariabelanalys eller för att bättre förstå exempel i kursboken ' Analys i en variabel' av Person/Böiers. Endel av övningsuppgifterna nedan refererar till övningsuppgifter ur kompendium 'Övningar i Envariabelanalys.

# 1 Kurvrikttning, gränsvärden och derivator

## 1.1 Plotta punkter och kurvor

Antar att vi vill plotta ett antal punkter $(x_1,y_1),(x_2,y_2),(x_2,y_2),....(x_n,y_n)$ i *xy*-planet. Vi skapar då i Matlab en vektor $x=(x_1,x_2,x_3,....x_n)$ med alla $x$ –koordinater och en vektor $y=(y_1,y_2,y_3....y_n)$ med alla $y$ –koordinater. Sedan plottar vi punkterna med kommandot **plot** . Vi kan om så önskas, i viss utsträckning styra form och färg på de plottade punkterna. Priva t.ex. kommandot

$>> x=[1\ 3\ 8\ -1\ 0];\ y=[2\ -1\ 5\ 0\ 3];\text{plot}(x,y,'\text{bo}')$

Om vi inte vill att punkterna skall ,sammanfalla med axlarna så kan vi skala om fönstret med kommandot **axis** och vill vi, i figuren, lättare kunna avläsa koordinatrena för olika punkter kan vi använda kommandot **grid** . Pröva t.ex.

>>axis ([-2 10 -2 7]) , grid on

Om vi utlämnar punkttyp så kommer Matlab att dra heldragna linjer mellan punkterna, i den ordning de är listade. Pröva t.ex.

>> plot(x,y,'b')

Om vi vill att de s.k. polygontåget (typ den kurva vi nyss ritat) skall börja och sluta i samma punkt måste naturligtvis den första och sista $x$ – resp. $y$ koordinaten stämma överens. Följande kommandot ritar t.ex. en polygon med fem hörn:

```
>>x=[cos(0)  cos(2*pi/5)  cos(4*pi/5 )  cos(6*pi/5) )  cos(8*pi/5)  cos(0)];
>>y=[ sin(0)  sin(2*pi/5) )  sin(4*pi/5) )  sin(6*pi/5) )  sin(8*pi/5)  sin(0)];
>>plot(x,y)
```

När punkmängderna följer en given regel som i exemplet ovan (där punkterna är jämt fördelade på en cirkel) så är det effiktivare att använda ''kolon''-kommandot för att generera vektorna x och y;

$>>\ t=0:2*\text{pi}/5:2*\text{pi};\text{x=cos(t);y=sin(t);}\ \text{plot(x,y)}$

Om vi vill rita en polygon med många fler hörn, så att figuren nästan sammanfaller med cirkeln

$x^2 + y^2 = 1$, så är det nu lätt att modifiera kommandot;

```
>> t=0:0.1:2*pi;x=cos(t);y=sin(t);plot(x,y)
```

Vill vi ha samma skalning på de båda axlarna så att vi ser att det är en cirkel (och ingen ellips) vi ritat, så ger vi kommandot;

```
>> axis('equal')
```

Om vi vill rita flera kurvor i samma bild så kan vi använda kommandot `hold` (se sid 266). T.ex.

```
>> t=0:0.1:2*pi;hold on,plot(2*cos(t),2*sin(t)),plot(cos(t),sin(t)),hold off
```

Eller enklare bara ge fler argument i `plot`-kommandot;

```
>> t=0:0.1:2*pi;plot(2*cos(t),2*sin(t),cos(t),sin(t))
```

Märkte du någon skillnad ?

Vi kan också styra plottningen av kurvorna till olika figurfönster med kommandot `figure` (se sid 266). Pröva t.ex.

```
>> t=0:0.1:2*pi;figure(1),plot(cos(t),sin(t),'r'),figure(2),plot(t,sin(t),'pentagram')
```

Glöm bara inte att byta till rätt figurfönster innan du plottar dina figurer. Glöm heller inte bort att stänga av `hold` om du inte vill rita över den senast gjorda plottningen.

Vi kan också dela in varje figurfönster i flera små delfönster med kommandot `subplot` (se sid 266). Pröva t.ex.

```
>> t=0:0.1:2*pi;subplot(2,1,1),plot(cos(t),sin(t)),subplot(2,1,2),plot(t,sin(t))
```

**Uppgift 1:** Använd `plot` för att plotta följande kurva (spiral) i $xy$-planet $\begin{cases} x = e^{\theta/10}\cos\theta \\ y = e^{\theta/10}\sin\theta \end{cases}$, $-10 \le \theta \le 100$. Plotta även kurvan $\begin{cases} x = e^{-10\cdot\theta}\cos\theta \\ y = e^{-10\cdot\theta}\sin\theta \end{cases}$, $-1 \le \theta \le 10$. På vilket sätt skär de båda kurvorna varandra ?

Testa också att plotta de båda kurvorna i polära koordinater med `polar`. Den första kurvan får vi t.ex. med kommandot; `>> t=-10:0.01:100;polar(t,exp(t./10))`. Förklara ! □

**Uppgift 2:** Generera i Matlab en vektor `x` med 100 tal (i stigande ordning) mellan $-10$ och 10. Generera sedan en vektor `y` med motsvarande sinusvärden ($y = \sin x$). Undersök och förklara sedan kommandona `plot(x,y)`, `plot(y,x)`, `plot(x)` och `plot(y)`. □

**Uppgift 3:** Undersök och förklara följande kommandorader;

(a) `>> x=0:0.01:1;y=rand(101);plot(x,y)`

(b) `>> x=rand(1,10);y=rand(1,10);area(x,y)`

(Upprepa gärna kommandona m.h.a. ↑-tangenten på skrivbordet.) □

**Uppgift 4:** Datamaterial kan ibland med fördel illustreras med olika typer av diagram.

(a) Undersök kommandot `>> x=rand(1,n);hist(x)`, för lite olika värden på `n`. Förklara varför variationen i staplarnas höjd verkar bli mindre för allt större `n` (jämför t.ex. då `n=10` med `n=1000000`)

(b) Producera i var sitt figurfönster diagram genererade av kommandona `bar`, `bar3`, `pie` och `pie3`

(c) Plotta bilderna från (b)-uppgiften i samma figurfönster, men i var sitt delfönster. Använd kommandot `title` för att skriva en lämplig rubrik överst i varje delfönster. □

## 1.2 Funktionskurvor

Det går naturligtvis bra att använda `plot` för att plotta funktionskurvor. Här ett exempel;

```
>> x=-10:0.01:10;y=sin(x)./x;plot(x,y)
```

Observera att Matlab varnar för att en otillåten beräkning gjorts, i det här fallet division med

noll. Detta kan undvikas genom att istället ge kommandot

```
>> x=-10:0.013:10;y=sin(x)./x;plot(x,y)
```

Varför ger Matlab inget varningsmeddelande nu ? Det finns naturligtvis fler otillåtna beräkningar som man kan försöka utföra. Vilket felmeddelande ger följande kommando ?

```
>> x=-10:0.013:10;y=log(x)./x;plot(x,y)
```

Logaritmen av ett negativt tal har vi ju inte definierat ännu (och kommer heller inte att göra så, under denna kurs), så låt oss istället plotta kurvan $y = \ln x/x$ för positiva $x$;

```
>> x=0.01:0.013:10;y=log(x)./x;plot(x,y)
```

Matlab anpassar skalningen av axlarna automatiskt, så att det största och minsta $x$ resp. $y$-värdet kommer med i figuren. Delvis p.g.a. att kurvan har en lodrät asymptot för $x = 0$, så blev skalningen i detta fall sådan att kurvans huvudsakliga karaktär inte framgick lika tydligt. Detta kan vi lösa på två sätt. Antingen tar vi hänsyn till detta när vi ger plottningskommandot och nöjer oss med att plotta kurvan för t.ex. $x \geq 0.8$, eller så skalar vi om axlarna enl. egna önskemål. T.ex.

```
>> x=0.01:0.013:10;y=log(x)./x;plot(x,y),axis([0 10 -0.5 0.5])
```

Ofta vill man använda en funktion flera gånger under samma arbetspass. Man kan då spara både tid och arbete genom att definiera en egen funktion med hjälp av kommandot `inline`. Här ett exempel;

```
>> f=inline('1./(1+x.^2)','x')
```

Vi kan sedan plotta den (som i avsnitt 1.1) med kommandot

```
>> x=-5:0.1:5; plot(x,f(x))
```

men ännu enklare är det i detta fall att använda kommandot `fplot` (funktionsplott);

```
>> fplot(f,[-5 5])
```

Det går bra att använda `fplot` även om man inte fördefinierat en funktion; t.ex. kan vi rita grafen till $e^{-x^2}$ med kommandot;

```
>> fplot('exp(-x.^2)',[-5 5])
```

Läs mer om `fplot` och alla dess finesser på sid 261.

Det går också bra att ha med flera parametrar/variabler i en `inline`-funktion. T.ex.

```
>> f=inline('1./(a+x.^2)','x','a')
```

Antag att vi vill plotta dessa funktionskurvor för lite olika värden på parametern $a$, i samma figur. Ett sätt att göra detta är enl. följande

```
>> x=-5:0.1:5; plot(x,f(x,1),x,f(x,2),x,f(x,3),x,f(x,4),x,f(x,5))
```

Vi skall senare se, i avsnitt 1.6, hur vi lättare gör detta om vi vill plotta riktigt många sådana kurvor, och kanske t.om. göra en film av alla kurvorna.

I analyskursen kommer vi ofta att definiera nya funktioner med hjälp av redan kända. Antag att vi definierat funktionen $f(x) = e^{x+1} \arctan x$ , och vill studera funktionen $g(x) = 2f(x-1)+3$. Vi börjar då med att göra en `inline`-funktion av $f$;

```
>> f=inline('exp(x+1).*atan(x)','x')
```

Sedan kan vi t.ex. rita graferna till $f$ och $g$ i samma figur med kommandot

```
>> x=-4:0.1:1.5; plot(x,f(x),x,2*f(x-1)+3)
```

Om man istället vill plotta speglingen av grafen till $f$ i linjen $x = y$ så ger man kommandot

```
>> x=-4:0.1:1.5; plot(f(x),x)
```

Detta kan vara bra att känna till om man vill studera inverser till funktioner. Varför ?

Ibland kommer vi också att studera funktioner som är definierade av olika uttryck på olika intervall. Antag att vi vill studera funktionen $h(x) = \begin{cases} 2^x & , \text{ för } x < 1 \\ 3 - x & , \text{ för } 1 \leq x < 4 \\ {}^{10}\log x & , \text{ för } x \geq 4 \end{cases}$.
Vi kan då använda relationsoperatorerna (se sid 59) för att på ett enkelt sätt definiera $f$ i Matlab;

```
>> h=inline('2.^(x).*(x<1)+(3-x).*(1<=x).*(x<4)+log10(x).*(x>=4)','x')
```

Om vi sedan plottar denna funktion på intervallet $[-5, 10]$ med `>> fplot(h,[5 10])`, så kommer Matlab att plotta en sammanhängande kurva, om än med lite "spetsar". Finns det någon del av den plottade kurvan du helst inte skulle vilja att den ritades ? Hur bär du dig i så fall åt för att undvika att den delen ritas ?

Det finns många fler sätt att styra utseendet på sina figurer, varav några kommer att tas upp i kommande delavsnitt. I den Windows-version av Matlab som ni jobbar i kan man emellertid ändra på och lägga till saker i bilder, genom att använda menyerna högst upp på figurfönstret. Om du vänsterklickar på t.ex. Tools och väljer Zoom In så kan du förstora upp bilden nära de punkter du vänsterklickar på. Detta kan vara användbart om man t.ex. vill få en grov uppskattning av nollställen eller punkter där funktioner har lokalt maximum/minimum. Vidare kan man med hjälp av knappen A skriva in text i figuren. Detta kan vara praktiskt om man vill göra en utskrift av bilden och där kunna se vilken funktion som ritats. Utskrift kan man för övrigt också få via menyn i figurfönstret (välj Print under File).

**Uppgift 5:** Använd kommandot `linspace` för att generera en `x`-vektor. Använd sedan `fplot` för att plotta funktionerna $\arctan x$, $\arcsin x$ och $\arccos x$. Försök att grafiskt verifiera dina svar på övningsuppgift 2.66 och 2.67, ur övningskompendiet 'Övningar i Envariabelanalys'. □

**Uppgift 6:** Gör `inline`-funktioner av $f$ och $g$ från övningsuppgift 2.17. Försök sedan avgöra svaren på så många som möjligt av de delfrågor som finns i övningsuppgiften, genom plottning i Matlab. □

**Uppgift 7:** Plotta lämplig funktion och använd zoomning för att få en grov uppskattning av lösningarna på ekvationerna i övningsuppgift 2.50. Bestäm sedan lösningarna exakt. □

**Uppgift 8:** Programmet `fplot` använder sig av en s.k. adaptiv metod, för att plotta grafer. Det betyder att den succesivt bestämmer de punkter i vilket funktionsvärdena beräknas. Först beräknas funktionsvärdena i jämnt fördelade punkter på anvisat intervall. Mellan de punkter där funktionsvärdena varierar mycket, kommer ytterligare funktionsvärden beräknas. Detta upprepas om och om igen tills avvikelsen från den exakta grafen är tillräckligt liten. Programmet kommer alltså att ta med fler punkter på grafen där lutningen är stor. Jämför nu i detta avseende `fplot` med `plot` genom att ge följande kommandon;

```
>> x=0.01:0.9:10; plot(x,log(x),'*'), grid
```

resp.

```
>> fplot('log(x)',[0.01 10], 0.03, '*'), grid
```

Adaptiva metoder är vanliga för många olika typer av numeriska beräkningar. Vi kommer titta lite närmare på en annan sådan, i samband med integralavsnittet, lite längre fram. □

## 1.3 Nivåkurvor

Ibland kommer vi i analyskursen ha användning av att rita kurvor vars punkter utgör lösningar till en ekvation. Då kan vi använda kommandona `meshgrid` och `contour` (se sid.283 resp. 280). Om vi t.ex. vill plotta lösningsmängden till ekvationen $yx^4 + 3xy^4 = a$, för lite olika värden på $a$ (t.ex. $a = 1, 5, 10$) så skriver vi

```
>> [x,y]=meshgrid(-3:0.1:3);z=y.*x.^(4)+3*x.*y.^(4);contour(x,y,z,[1,5,10])
```

Vi kommer i denna kurs bara att studera funktioner av en variabel, men låt oss bara tillfälligt (för skojs skull) undersöka hur det ser ut när Matlab plottar grafer till funktioner av två variabler. Låt oss plotta en del av grafen till funktionen $f(x, y) = yx^4 + 3xy^4$. Med `x,y` och `z=f(x,y)` som ovan, så räcker det nu att ge någon av kommandona;

```
>> mesh(x,y,z)
```

```
>> plot3(x,y,z)
```

```
>> surf(x,y,z)
```

```
>> surfl(x,y,z)
```

eller

```
>> surfc(x,y,z)
```

Pröva alla och jämför. Om man går in på Tools och väljer Rotate 3D, i figurfönstrets menyer, så kan man rotera ytan och se den från olika håll. Prova gärna med någon annan funktion (t.ex. `z=peaks(x,y)`) och låt dig imponers av Matlabs grafik. Till sist ett litet exempel på en yta som är gjord med kommandot `surf`;

```
>> knot
```

**Uppgift 9:** Använd `meshgrid` och `contour` för att plotta lösningsmängderna till ekvationerna;

(a) $x^2+y^2=1$ (b) $x^2-y^2=1$

(c) $x^2-y^2=0$ (d) $x^{10}+y^{10}=1$

Testa även `surf` på motsvarande funktioner. □

## 1.4 Gränsvärden

Gränsvärdesbegreppet är ett centralt begrepp i Analyskursen och därmed inom alla vetenskaper som använder Matematik som redskap. I avsnitt 1.2 studerade vi graferna till funktionerna $\sin x/x$ och $\ln x/x$. Ingen av dessa funktioner är definierade för $x=0$, men på de båda graferna såg vi en väsentlig skillnad i utseendet för $x$ när 0. De båda graferna indikerar att $\lim_{x\to 0^+} \dfrac{\ln x}{x} = -\infty$ och att $\lim_{x\to 0^+} \dfrac{\sin x}{x} = 1$. Det första gränsvärdet är uppenbart, från kännedom om de elementära funktionerna. Varför ? Det andra däremot är lite knepigare. Både täljaren $\sin x$ och nämnaren $x$ går mot 0, då $x$ går mot 0 från positiva $x$. Vad som händer med värdena på $\dfrac{\sin x}{x}$ för små värden på $x$, beror då på hur "snabbt" täljaren resp. nämnaren närmar sig 0, i förhållande till varandra. Man kan emellertid strikt (ej genom plottning) visa att $\lim_{x\to 0^+} \dfrac{\sin x}{x} = 1$ (Det är ett s.k. standardgränsvärde, se sid sid 85 i Analysboken).

Även om vi inte kan bevisa något m.h.a. plottning, så kan en grafisk studie av gränsvärden vara till hjälp om vi snabbt vill bilda oss en uppfattning om huruvida gränsvärde existerar eller ej. Om det existerar kan vi också approximativt avläsa gränsvärdet. Ni kan också använda plottning för att kontrollera en del av era svar från övningar i Analyskursen.

Om vi inte känner oss övertygade om att den indelning vi valt för att plotta funktionskurvan är fin nog för att vara säker på vårt gränsvärde, så kan vi plotta med logaritmisk skala på $x$-axeln med kommandot `semilogx` (se sid. 262). För att begränsa antalet punkter gör vi också en logaritmisk fördelning av plottpunkterna $x$ med kommandot `logspace` (se sid. 77). En ny plottning av $\sin x/x$, med logaritmisk skala på $x$-axeln, får vi t.ex. med kommandot

```
>> x=logspace(-50,0,600);y=sin(x)./x;semilogx(x,y)
```

Samma teknik går att använda för att undersöka gränsvärden då $x\to\infty$. Pröva t.ex.

```
>> x=logspace(0,100,600);y=sin(x)./x;semilogx(x,y)
```

Vi kan även studera talföljder på samma sätt som ovan. Talföjder kan vi ju betrakta som funktioner definierade på heltalen. Om vi t.ex. vill undersöka $\dfrac{n^2+n-1}{1+3n+2n^2}$ då $n\to\infty$, så kan vi ge kommandot

```
>> n=0:20;y=(n.^(2)+n-1)./(1+3*n+2*n.^2);plot(n,y,'*')
```

Alternativt kan vi använda oss av observationen att en talföljd har gränsvärde om motsvarande funktion har det (omvändningen gäller dock ej i allmänhet). I så fall kan vi använda oss av logaritmisk fördelning av punkterna;

```
>> x=logspace(0,100,600);y=(x.^(2)+x-1)./(1+3*x+2*x.^2);semilogx(x,y)
```

**Uppgift 10:** Undersök m.h.a. plottning några av gränsvärdena i övningsuppgift 3.33. Försök sedan beräkna gränsvärdena exakt. □

**Uppgift 11:** Undersök höger- och vänstergränsvärde i $x = 0$ av funktionen $\arctan \frac{1}{x}$ , $x \neq 0$. □

**Uppgift 12:** Undersök gränsvärdet $\lim_{x \to 0^+} \sin(1/x)$. Plotta med logaritmisk skalning av $x$-axeln. Förklara den bild du får upp. □

**Uppgift 13:** Använd uttrycket $\left(1 + \frac{1}{n}\right)^n$ för att bestämma ett bra närmevärde till $e$. □

**Uppgift 14:** Undersök m.h.a. plottning vad som händer med uttrycket $\sin\left(\frac{\pi x^2}{x+1}\right)$ då

(a) $x$ är heltal som går mot oändligheten.

(b) $x$ är reella tal som går mot oändligheten.

Förklara eventuella skillnader! □

## 1.5 Derivator

Derivatan av en funktion $f$ i en punkt $x$ definieras som gränsvärdet $\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

För små värden på $h$ är således $\frac{f(x+h) - f(x)}{h}$ ett bra närmevärde till $f'(x)$.
Betrakta t.ex. följande `inline`-funktion;

```
>> f=inline('x.*sin(x)','x')
```

Vi kan då beräkna ett närmevärde till $f'(2)$ med följande kommando;

```
>> x=2;h=0.01;df=(f(x+h)-f(x))/h
```

Ju mindre vi väljer $h$ desto bättre approximation får vi. Om $h$ är positiv så får vi högerderivatan och om $h$ är negativ så får vi vänsterderivatan.

Om vi vill rita grafen till $f'(x)$ så behöver vi nu bara beräkna närmevärden i lite fler punkter;

```
>> x=-1:0.01:5;h=0.01;df=(f(x+h)-f(x))/h;plot(x,df)
```

Om vi vill plotta grafen till $f$ i samma figur så byter vi här bara ut plotkommandot `plot(x,df)` mot `plot(x,f(x),x,df)`.
Eftersom vi kan beräkna derivatan exakt, nämligen $f'(x) = \sin x + x \cos x$, så skulle vi kunna jämföra grafen till den numeriska derivatan med den exakta derivatan, genom att plotta dom i samma figur med kommandot;

```
>> plot(x,df,x,sin(x)+x.*cos(x))
```

Approximationen är så bra att kurvorna i princip sammanfaller. Vill vi trots allt försöka få en uppfattning om felets storlek kan vi istället plotta skillnaden;

```
>> plot(x,df-(sin(x)+x.*cos(x)))
```

Hur stort är felet i detta exempel ?
Beräknar vi istället derivatan genom att ta medelvärdet av vänster och högerderivatan, dvs. $f'(x) = \frac{f(x+h) - f(x-h)}{2h}$, så ger motsvarande differenskvot en ännu bättre approximation;

```
>> x=-1:0.01:5;h=0.01;df=(f(x+h)-f(x-h))/(2*h);plot(x,df-(sin(x)+x.*cos(x)))
```

Hur stort är felet nu ?

Oavsett vilken differenskvot som används så beror noggranheten mest på hur litet $h$ väls. Pröva nu föregående kommando, fast med $h = e^{-12}$;

```
>> x=-1:0.01:5;h=exp(-12);df=(f(x+h)-f(x-h))/(2*h);plot(x,df-(sin(x)+x.*cos(x)))
```

Nu är avvikelsen så liten att avrundningsfelen som Matlab begår börjar få betydelse. Undersök vad som händer med ett ännu mindre värde på $h$. Förklara vad som händer då $h = e^{-50}$.

Vi kan också studera andraderivatan av funktioner på liknande sätt som ovan. Från definitionen av derivata följer efter lite manupilation (se övning 4.7) att $f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$.

För små värden på $h$ är således $\dfrac{f(x+h)-2f(x)+f(x-h)}{h^2}$ ett bra närmevärde till $f''(x)$.

Följande kommandorader plottar grafen till $f(x) = \sin(e^x)$, samt dess första och andra derivata, i olika delfönster;

```
>> f=inline('sin(exp(x))','x')
>> x=-1:0.01:4;h=0.01;df=(f(x+h)-f(x))/h;ddf=(f(x+h)-2*f(x)+f(x-h))/(h^2);
>> subplot(3,1,1),plot(x,f(x)),subplot(3,1,2),plot(x,df),subplot(3,1,3),plot(x,ddf)
```

**Uppgift 15:** Plotta derivatan av funktionen $\arctan 2x$. Uppskatta även avikelsen från den exakta derivatan. □

**Uppgift 16:** Utför följande kommandorader;

```
>> f=inline('polyval([1 -8 18 -1 1],x)','x')
>> x=-1:0.01:4;h=0.01;df=(f(x+h)-f(x-h))/(2*h);ddf=(f(x+h)-2*f(x)+f(x-h))/(h^2);
>> plot(x,f(x),x,df,x,ddf),grid
```

Då plottas grafen till en funktion, samt dess första och andra derivata. Utan att studera dessa kommandon närmare, försök ur figuren avgöra vilken som är vilken. □

**Uppgift 17:** Plotta funktionerna i övningsuppgift 4.25, tillsammans med ev. derivata. Försök att grafiskt avgöra vilken som är kontinuerlig, deriverbar samt har kontinuerlig derivata. Försök sedan verifiera eventuella observationer. □

**Uppgift 18:** Bestäm en metod för numerisk beräkning av tredjederivatan och plotta m.h.a. denna tredjederivatan av $f(x) = xe^x$ (obs! du får inte använda det exakta uttrycket på derivatan). □

## 1.6 Filmvisning

Vi har, i avsnitt 1.1, sett hur man kan lägga in flera kurvor i samma figur. Om man istället vill plotta dom efter varandra i samma figurfönster som en bildsekvens, så kan vi göra en liten snurra. Antag att vi vill plotta graferna till funktionerna $(x+2a)\sin ax$, för lite olika värden på $a$. Följande snurra är ett exempel på hur man skulle kunna göra detta;

```
>> f=inline('(x+2*a).*sin(a*x)','x','a'); x=-9:0.01:9;
>> for a=-3:0.05:3; plot(x,f(x,a)), axis([-9 9 -9 9]), pause(0.1), end
```

Här har vi använt kommandot `axis` för att inte skalningen av axlarna skall ändras från bild till bild. Kommandot `pause` är helt enkelt till för att vi skall hinna med att se vad som händer.

Om man vill spara alla bilderna, som bilder i en "filmsekvens", så behöver vi bara ändra lite i ovanstående snurra. Först måste vi skapa minnesutrymme för en film (som vi här låter ha namnet "film") och ange hur många bilder som skall lagras. Detta gör vi med kommandot;

```
>> film=moviein(120);
```

Sedan använder vi kommandot `getframe` (se sid. 386) för att lagra bild för bild i snurran;

```
>> n=0; for a=-3:0.05:3; n=n+1;plot(x,f(x,a)), axis([-9 9 -9 9]), film(:,n)=getframe; end
```

Varje gång vi vill spela upp vår film behöver vi nu bara skriva

```
>> movie(film,1)
```

Här anger siffran 1 att filmen skall visas en gång.

**Uppgift 19:** Gör en film över spiralerna $\begin{cases} x = e^{\theta/10}\cos(a\theta) \\ y = e^{\theta/10}\sin(a\theta) \end{cases}$, $-10 \le \theta \le 100$, då $a$ varierar mellan 0 och 6 (tag med ca. 100 bilder). Använd sedan kommandot `movie` för att spela upp filmen några gånger. □

**Uppgift 20:** Betrakta övningsuppgift 4.32

(a) Plotta, i samma figur, graferna till polynomen för några olika värden på $a$ mellan 0 och 1, t.ex. 0,0.3,0.6,0.9 (Tips: plotta graferna där $-1 < x < 3$). Verkar påståendet i deluppgift (c) stämma ? Vad tror du om svaret på deluppgift (d) ?

(b) Gör en snurra över några $a$-värden som varierar mellan 0 och 1 (förslagsvis ca. 20 stycken). Plotta i varje steg grafen till motsvarande funktion. Om du lägger in kommandot `pause` inuti snurran så kommer du att kunna mata fram bild för bild genom att trycka på valfri tangent på skrivbordet. Försök nu uppskatta svaret på deluppgift (b).

(c) Låt nu istället $a$ anta värden mellan 0 och 10 och modifiera din snurra så att det blir till en filminspelning. Vad tror du om svaret på deluppgift (f) ?

(d) Lös övningsuppgift 4.32 med papper och penna och ev. bekräfta dina gissningar ovan. □

# 2 Ekvationslösning och anpassning till mätdata

## 2.1 Ekvationslösning

En del ekvationer kan vi lösa exakt, t.ex. linjära ekvationssystem och andragradsekvationer, men för de flesta ekvationer får vi i bästa fall nöja oss med en numerisk lösning.

Låt oss först titta lite på hur man hittar nollställen till polynom m.h.a. Matlab. Betrakta t.ex. polynomet $p(x) = 2x^5 - 3x^4 + x^2 - 1$. Det är alltid bra att först plotta grafen till den funktion man vill studera. Det går naturligtvis bra att plotta polynom på samma sätt som tidigare beskrivet (avsnitt 1.2), men polynom är så vanliga att Matlab har många speciella (och tidsbesparande) kommandon för hantering av polynom. Till exempel kan vi beräkna funktionsvärdena med kommandot `polyval`;

```
>> x=-2:0.01:2; y=polyval([2 -3 0 1 -1 0],x);
```

Istället för att skriva hela polynomet behövde vi här bara ange koefficienterna framför varje term med vektorn `[2 -3 0 1 0 -1]`. Nu är det bara att plotta punkterna på samma sätt som vanligt;

```
>> plot(x,y)
```

Vi ser att det finns minst en, och kanske flera, nollställen nära $x = 0$. Vi kan nu låta Matlab numeriskt beräkna alla nollställen med kommandot `roots`;

```
>> roots([2 -3 0 1 -1 0])
```

Förutom de speciella polynom-kommandon vi använt ovan så finns det t.ex. kommandon (program) som multiplicerar polynom (`conv(p,q)`), dividerar polynom (`deconv(p,q)`) och anpassar polynom till ett antal punkter ((`polyfit`).

För mer allmänna funktioner finns det ett annat kommando, nämligen `fzero`, för att bestämma nollställen. Med detta kommando hittar man emellertid högst en rot i taget. Som de flesta numeriska metoder använder sig `fzero` av en iterativ metod ($x_{n+1} = F(x_n)$), och behöver förutom funktionen också ha ett startvärde ($x_0$). Om vi söker en rot nära en viss punkt $p$, så är det naturligt att ange $p$ som startvärde. Till exempel söker följande kommando efter ett nollställe till $e^x + \sin x$ nära $x = -1$;

```
>> fzero('exp(x)+sin(x)',-1)
```

Om man använder `fzero` så är det speciellt viktigt att rita en graf, för att bilda sig en uppfattning om var nollställena ligger (om det finns några);

```
>> x=-10:0.1:1;plot(x,exp(x)+sin(x)),grid
```

Det är inte lätt att på förhand alltid veta vilka intervall som är intressanta att plotta. Man får då experimentera sig fram eller ännu bättre försöka studera funktionen med papper och penna. När man i den plottade grafen väl observerat ett förmodat nollställe, så kan man t.ex. använda sig av zoomning (vänsterklicka på Zoom In under Tools på menyraden högst upp i figurfönstret) för att hitta lämpliga startvärden till `fzero`.

Observera dock att kommandot `fzero` inte kommer att hitta nollställen där funktionen inte växlar tecken. Pröva t.ex.

```
>> fzero('x.^2',1)
```

I läsperiod 1 såg vi (avsnitt 1.6 i 'Matematik med Matlab') hur man kunde hitta nollställen till funktioner m.h.a. Newton-Raphsons metod. Då använde vi tekniken med en s.k. mekanisk snurra (dvs. upprepad användning av ↑-tangenten på skrivbordet). Låt oss istället göra en liten snurra som löser ekvationen $\cos x = x$ med Newton–Raphsons metod;

```
>> x0=1;tol=1;
>> while tol,x=x0;x0=x-(x-cos(x))/(1+sin(x));tol=(abs(x-x0)>0.00001);end,x0
```

Förklara varför denna snurra fungerar. Vilken innebörd har t.ex. `tol` ? Pröva att hitta lösningar till en annan ekvation på formen $f(x) = 0$. Visa att Newton-Raphsons metod även fungerar om funktionen inte växlar tecken i nollstället.

Metoden med snurra fungerar naturligtvis på alla talföljder som är rekursivt definierade av $x_{n+1} = F(x_n)$ för någon funktion $F$. Under lämpliga villkor på $F$ och startvärdet $x_0$ (se t.ex. Sats.3 sid 201) så kommer en sådan talföljd att konvergera. Låt oss till exempel studera talföljden definierad genom $x_{n+1} = \sqrt{1+x_n},\ n \in \mathbb{R},\ x_0 = 1$. En snurra för att testa ev. gränsvärde $n \to \infty$ skulle kunna se ut enl. följande;

```
>> x0=1;tol=1;n=1;
>> while (tol&(n<100)),x=x0;x0=sqrt(1+x);tol=(abs(x-x0)>0.00001);n=n+1;end,x0,n
```

Räknaren `n` ser här till att snurran inte pågår för evigt, utifall talföljden inte skulle konvergera.

Antag att man är intresserad av att hitta lokala extrempunkter till en funktion. Ett alternativ är då att för hand först derivera funktionen och sedan undersöka derivatans nollställen med någon av metoderna ovan. Om man vet på ett ungefär var extrempunkterna finns så skulle man också kunna använda någon form av intervallhalverings teknik. Fundera själv igenom hur det skulle kunna gå till (skissa t.ex. på någon funktionsgraf med lokalt maximum och rita in i din figur hur du halverar intervall). Matlab har emellertid ett redan inbyggt kommando som hjälper oss att hitta största och minsta värde till en funktion på ett intervall, nämligen `fmin` (se sid. 179). Som kommandonamnet antyder så hittar `fmin` ett lokalt minimum på intervallet (vilket också kan vara en ändpunkt). Finns det flera lokala minimipunkter så hittar den en av dem. T.ex. ger kommandot;

```
>> x=fmin('x+3*sin(x)',3,6)
```

den punkt, på intervallet $[3, 6]$, i vilket funktionen $f(x) = x + 3\sin x$ antar sitt minsta värde. Plotta grafen till funktionen. Verkar `fmin` ha jobbat bra ?
I detta fallet antogs det minsta värdet i en inre punkt på intervallet, men `fmin` fungerar även om minsta värdet antas i en ändpunkt på intervallet. Pröva t.ex.

```
>> x=fmin('x+3*sin(x)',3,4)
```

Vill vi beräkna minimum av en `inline`-funktion, så går det också bra. Då är det även snabbt gjort att beräkna det minsta värdet;

```
>> f=inline('x+3*sin(x)','x');x=fmin(f,3,6),y=f(x)
```

Vi kan också använda `fmin` för att hitta det största värdet till en funktion $f(x)$. Största värde antas ju för det $x$ i vilket funktionen $-f(x)$ antar sitt minsta värde (Varför ?). Största värdet till funktionen ovan, på intervallet $[1, 3]$, får vi därför t.ex. med kommandot

```
>> max=f(fmin('-(x+3*sin(x))',1,3))
```

**Uppgift 21:** Pröva `roots` på polynomen $1 + 3x - 2x^3 + x^5,\ x^2$ och $x^2 + 1$. □

**Uppgift 22:** Pröva `fzero` på funktionerna $\tan x - x,\ 1 - \cos x$ och $x^2 + 1$. □

**Uppgift 23:** Gör en snurra som undersöker gränsvärdet av den talföljd som är rekursivt definierad av $x_{n+2} = \sqrt{x_{n+1} + x_n + 1}, n \in \mathbb{N},\ x_0 = x_1 = 1$. □

**Uppgift 24:** Hitta största värdet av funktionen $f(x) = e^x \sin x$ på intervallet $[0, 1]$ genom att beräkna derivatan för hand och sedan använda någon numerisk metod för att hitta nollstället till derivatan. Ange approximativt funktionens största värde på intervallet. Använd sedan papper och penna för att ta fram det exakta värdet. Hitta avslutningsvis det största värdet m.h.a. `fmin`. □

**Uppgift 25:** Gör en funktionsfil för Newton-Raphsons metod. Som invärden i funktionen bör man kunna ge en funktion, ett startvärde och en feltolerans. Första raden i funktionsfilen skulle kunna ha utseendet; `function rot=NRrot(f,x0,tolerans)`. Du kan komma att ha nytta av din funktion i bl.a. Analyskursen. □

## 2.2 Anpassning till mätdata

I kursen 'Linjär Algebra' i läsperiod 1 lärde vi oss hur man kunde hitta det polynom $p(x)$, av en viss grad, som i minsta kvadratmening bäst anpassade sig till en massa punkter $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, ..., $(x_n, y_n)$. Mer precist så bestämde vi koefficienterna i $p(x)$ så att $\sum_{k=1}^{n}(y_k - p(x_k)^2$ blev så litet som möjligt. Vi såg också hur man kunde hitta detta minimum genom att lösa det s.k. felutjämnade systemet. I Matlab kan man använda kommandot `polyfit` för att lösa detta. Låt oss titta på ett exempel. Vi börjar med att generera lite slumppunkter, som får symbolisera mätdata från något experiment;

```
>> x=7*rand(1,10); y=sin(x)+rand(1,10)./10; plot(x,y,'*')
```

Antag sedan att vi söker det polynom $p(x) = ax^2+bx+c$, av grad två, som i minsta kvadratmening bäst anpassar sig till våra punkter. Problemet löses i Matlab av kommandot;

```
>> p=polyfit(x,y,2)
```

Resultatet blir en vektor med koefficienterna för det sökta polynomet. Vi kan sedan plotta polynomet i samma figur som slumppunkterna;

```
>> px=0:0.1:7;py=polyval(p,px);plot(x,y,'*',px,py)
```

Eftersom slumppunkterna är skapade som en liten avvikelse från en sinuskurva, så finns det ingen andragradskurva som stämmer bra överens. Om vi istället anpassar något polynom av högre grad så blir det lite bättre. Pröva t.ex. grad 3 och 10. Även om man genom att höja graden kan få polynomet att överenstämma riktigt bra, så känns det mer naturligt att försöka hitta den sinuskurva som bäst approximerar punkterna. Låt oss därför nu istället försöka bestämma konstanter $a, b, c$ och $d$, så att $f(x) = a\sin(bx + c)+d$ , approximerar punkterna så bra som möjligt, dvs. minimerar $\sum_{k=1}^{n}(y_k - f(x_k)^2$. Problemet är bara att motsvarande ekvationssystem $y_k = f(x_k),\ k = 1, 2, 3, ..., n$, inte är linjärt i $a, b, c$ och $d$, så vi kan inte använda samma metod som ovan. Vi måste då använda en mer allmän metod för minimering av funktioner som beror av flera variabler/parametrar. Det går utanför ramen av denna kurs att göra en studie av sådana metoder. Vi nöjer oss helt enkelt med att se hur Matlab kan hjälpa oss att lösa detta problem. Först måste vi göra en funktionsfil av den funktion vi skall minimera. Den skulle kunna se ut så här;

```
function f=felfunk(a)
global x y
f=sum((y-(a(1)*sin(a(2)*x+a(3))+a(4))).^2)
```

Vi hittar sedan funktionens minimipunkt med kommandot `fmins`;

```
>> global x y, a=fmins('felfunk',[1 1 1 1],[0 0.001]);
```

En plottning bekräftar avslutningsvis att vi erhållit en bra kurvanpassning;

```
>> t=0:0.1:7;plot(x,y,'*',t,a(1)*sin(a(2)*t+a(3))+a(4))
```

Låt oss nu titta på ett lite mer tillämpat exempel. Antag att ett mätinstrument, vid olika tidpunkter, registrerat positionen hos en stålkula som skjuts ur en kanon och lagrat alla mätdata i en m-fil med namnet `data.m`;

```
% data.m: längd och höjd
8.9520  1.0438
9.4239  0.6719
3.3508  2.3072
4.3736  2.5154
1.4931  1.4092
5.3250  2.5439
```

Enligt Newtons andra lag beskrivs emellertid kulans rörelse av ett allmänt samband av formen; $y = x\tan\alpha - \dfrac{gx^2}{2v_0^2\cos^2\alpha}$, där $v_0$ är utgångshastigheten, $\alpha$ är utgångsvinkeln, g är accelerationen vid fritt fall (dvs. gravitationskonstanten som är $\approx 9.8$) och $(x, y)$ är kulans position. Låt oss nu försöka bestämma $\alpha$ och $v_0$ så att detta sambandet, så bra som möjligt, stämmer in på de mätdata som är givna i filen `data.m`. Vi börjar då med skriva en funktionsfil för felfunktionen;

```
function f=parabelfel(a)
global x y
f=sum((y-(x*tan(a(1))-(9.8*x.^2)/(2*a(2)^2*(cos(a(1)))^2))).^2);
```

Sedan omvandlar vi mätdatan i filen `data.m` till en matris med namnet `data`;

```
>> load('data.m')
```

Det `a` som minimerar felfunktionen hittar vi sedan med kommandoraden;

```
>> x=data(:,1); y=data(:,2); global x y, a=fmins('parabelfel',[1 1],[0 0.001]);
```

Avslutningsvis kan vi plotta mätdata och den anpassade kurvan i samma figur;

```
>> t=0:0.1:11;plot(x,y,'*',t,t*tan(a(1))-(9.8*t.^2)/(2*a(2)^2*(cos(a(1)))^2))
```

Om vi på ett överskådligt sätt vill studera kurvans avvikelse från datapunkterna så kan vi också plotta skillnaden;

```
>> plot(x,y-(x*tan(a(1))-(9.8*x.^2)/(2*a(2)^2*(cos(a(1)))^2)),'*')
```

**Uppgift 26:** Ta hjälp av `fmins` för att lösa övningsuppgift 2.25 □

**Uppgift 27:** Anpassa en andragradskurva $y = ax^2 + bx + c$ till de mätdata som är givna i filen `data.m` i texten ovan. Studera sedan andragradskurvans avvikelse från de uppmätta punkterna. Hur stor är den största avvikelsen ? Hur stort är minstakvadratfelet (dvs summan av felen i kvadrat) ? Hur många mätpunkter ligger över resp. under den anpassade kurvan ? I texten ovan anpassades ju också en andragradskurva till samma mätdata. Varför ger de båda modellerna inte samma anpassning ? Vilken modell är att föredra om man vill uppskatta kulans nedslagsplats ? □

**Uppgift 28:** Antag att vi vill anpassa en exponentialkurva $y = Ce^{kx}$ till punkterna $(0, 3.43)$, $(0.20, 6.41)$, $(0.40, 10.58)$, $(0.60, 22.38)$, $(0.8, 34.0472)$ och $(1.00, 68.70)$. Skriv kurvans ekvation på formen $\ln y = \ln C + kx$ och använd `polyfit` för att hitta den kurva som i minsta kvadratmening bäst anpassar sig till punkterna (beräkna $\ln y$ för alla givna $y$-värden och bestäm minsta kvadratlösningen m.a.p. $\ln C$ och $k$). Använd sedan metoden med `fmins` för att anpassa kurvan $y = Ce^{kx}$, utan att först göra någon omskrivning av ekvationen. Varför ger de båda metoderna inte samma exponentialkurva ? □

**Uppgift 29:** På kursens hemsida (under kursmaterial) kan du hämta en m-fil med namnet `matdata.m`, som innehåller lite simulerade mätdata. Bestäm den funktion på formen $f(x) = a \cdot \arctan(bx) + b \cdot \ln(ax)$ vars graf bäst ansluter sig till mätdatan i filen `matdata.m`. □

# 3 Summor och Integration

## 3.1 Summor

Om vi t.ex. vill beräkna summan $\sum_{k=1}^{5} \frac{1}{1+k^2}$, så är det inte fler termer än att vi kan addera alla tal direkt i Matlabfönstret;

```
>> 1/2+1/5+1/10+1/17+1/26
```

Alternativt kan vi använda kommandot `sum` som summerar alla element i angiven vektor;

```
>> sum([1/2 1/5 1/10 1/17 1/26])
```

Detta blir speciellt effektivt om summan består av många fler termer. Summan $\sum_{k=1}^{100} \frac{1}{1+k^2}$ beräknar vi t.ex. enkelt med kommandot;

```
>> x=1:100;y=1./(1+x.^2);sum(y)
```

Man skulle kunna ställa sig frågan hur värdet på summan förändras om vi tar med allt fler och fler termer i summan. Vi kan beräkna delsummorna $s_n = \sum_{k=1}^{n} \frac{1}{1+k^2}$ för succesivt ökande $n$, m.h.a.

en liten snurra;

```
>> s(1)=0.5; for n=2:1000, s(n)=s(n-1)+1/(1+n^2); end
```

och sedan plotta talföljden $s_n$ ;

```
>> plot(s)
```

Låter vi $n$ gå mot oändligheten så får vi en summa $\sum_{k=1}^{\infty} \frac{1}{1+k^2}$ med oändligt många termer. En sådan summa kallas även för en serie (läs mer om sådana i avsnitt 2.5.4 i boken 'Analys i en variabel' av Persson/Böijers). Om gränsvärdet $\lim_{n\to\infty} s_n$ existerar så säger vi att serien är konvergent, i annat fall säger vi att serien är divergent. Med utgångspunkt från plottningen av de tusen första delsummorna ovan så verkar det i detta fall som att serien är konvergent med en summa som är ca. 1.0757. Låt oss nu på samma sätt studera delsummorna till serien $\sum_{k=1}^{\infty} \frac{1}{\sqrt{1+k^2}}$;

```
>> clear s, s(1)=1/sqrt(2); for n=2:1000, s(n)=s(n-1)+1/sqrt(1+n^2); end, plot(s)
```

Trots att denna serie har mycket gemensamt med den förra, som t.ex. att termerna är positiva och avtar mot 0, så verkar serien $\sum_{k=1}^{\infty} \frac{1}{\sqrt{1+k^2}}$ vara divergent. Naturligtvis duger det inte med plottning för att konstatera detta faktum, utan en närmare analys är nödvändig.

**Uppgift 30:** Beräkna de tusen första delsummorna till följande serier och försök avgör om de är konvergenta eller divergenta.

(a) $\sum_{k=1}^{\infty} \frac{1}{k^{0.99}}$ (b) $\sum_{k=1}^{\infty} \frac{1}{k^{1.01}}$ (c) $\sum_{k=1}^{\infty} (2\arctan k - \pi)$ (d) $\sum_{k=1}^{\infty} \frac{k^2}{2^k}$ □

**Uppgift 31:** Använd kommandot `sum` för att beräkna ett approximativt värde på den konvergenta serien $\sum_{k=1}^{\infty} e^{-k}$. Beräkna sedan seriens exakta värde m.h.a. formel för geometriska summor. □

**Uppgift 32:** Man kan visa att $\ln(x+1) = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{x^k}{k}$ för $-1 < x \leq 1$. Undersök VL och HL i denna identitet för några olika värden på $x$. Bekräftar dina iaktagelser att identiteten verkar stämma ? □

**Uppgift 33:** Tag hjälp av Matlab för att besvara frågorna i uppgift 1.29a ur övningskompendiet 'Övningar i Envariabelanalys'. Gör sedan uppgift 1.29b. □

## 3.2 Numerisk Integration

Bra metoder för att approximativt beräkna integraler är mycket betydelsefulla. Sådana metoder behövs bl.a. för integraler som inte kan beräknas exakt. Men även om en exakt beräkning är möjlig, kan en numerisk beräkning vara väl så intressant, särskilt som en sådan kan göras snabbt med hjälp av dator. En naturlig metod för numerisk beräkning av integraler är genom approximation med Riemannsummor. Om $f$ är en kontinuerlig funktion på ett intervall $[a,b]$, så gäller t.ex. att

$$(*) \qquad \frac{b-a}{n} \sum_{k=1}^{n} f(a + \frac{k}{n}(b-a)) \quad \longrightarrow \quad \int_a^b f(x)\,dx \quad , \text{ då } \quad n \to \infty$$

Verifiera detta ! (se Sats 4 på avsnitt 6.2 i boken 'Analys i en variabel').

Med $a = 1, b = 3$ och $f(x) = \ln x$ så får vi t.ex. att $\frac{2}{n} \sum_{k=1}^{n} \ln(1 + \frac{2k}{n}) \to \int_1^3 \ln x\,dx$ då $n \to \infty$.

För stora $n$ är således $\frac{2}{n} \sum_{k=1}^{n} \ln(1 + \frac{2k}{n})$ en bra approximation av värdet på integralen $\int_1^3 \ln x\,dx$.

Beräkningen;

```
>> n=1000; x=1:1000; y=log(1+2*x/n); summan=(2/n)*sum(y)
```

ger t.ex. att $\int_1^3 \ln x\,dx \approx 1,30$ (visa att det exakta värdet är $3\ln 3 - 2$).

För generaliserade integraler, där intervallet är obegränsat, får vi något modifiera tekniken ovan.

Betrakta t.ex. den generaliserade integralen $\int_1^\infty \frac{1}{1+x^2}\,dx$. Vi kan som ovan approximativt beräkna integralerna $\int_1^b \frac{1}{1+x^2}\,dx$, för ändliga värden på $b$. Om $b$ är stort så kommer dessa approximationer också att vara bra approximationer av $\int_1^\infty \frac{1}{1+x^2}\,dx$ , eftersom denna integral är konvergent. Vi skulle också kunna succesivt öka den övre integrationsgränsen $b$ för varje steg $n$ av förfinad intervallindelning. Om vi vill undersöka om en viss generaliserad integral $\int_a^\infty f(x)\,dx$ är konvergent, och i så fall uppskatta dess värde, så ersätter vi helt enkelt $b$ i (*) med en funktion $b(n)$, sådan att $b(n) \to \infty$ och $\frac{b(n)}{n} \to 0$ då $n \to \infty$. Med $a = 1, b = \sqrt{n}$ och $f(x) = \frac{1}{1+x^2}$ så ger;

```
>> n=10000; x=1:10000; y=1./(1+(1+(sqrt(n)-1)*x/n).^2); summan=((sqrt(n)-1)/n)*sum(y)
```

att $\int_1^\infty \frac{1}{1+x^2}\,dx \approx 0.77$ (obs! det exakta värdet är $\pi/4$).

Det finns andra mer effektiva metoder för beräkning av integraler. I kursbokens avsnitt 7.11 finns förutom "rektangeluppskattningen" ovan också metoden med "trapetsuppskattning" beskriven. En annan, ännu effektivare metod, bygger på den s.k. Simpsons formel;

$$\int_a^b f(x)\,dx \approx \frac{b-a}{3n}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right)$$

där $a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$ (obs! I Simpsons formel måste $n$ vara ett jämnt tal). Istället för att som i rektangeluppskattningen approximera funktionen med styckvis konstanta funktioner, eller som i trapetsuppskattningen med styckvis linjära funktioner, så erhålls Simpsons formel genom att på delintervall approximera funktionen med andragradspolynom. Låt oss t.ex. beräkna integralen $\int_0^1 \frac{1}{1+x^2}\,dx$ med hjälp av Simpsons formel. Med $x_k = \frac{k}{4}, k = 0, 1, 2, 3, 4$, får vi

$$\int_0^1 \frac{1}{1+x^2}\,dx \approx \frac{1-0}{3\cdot 4}(f(0)+4f(0.25)+2f(0.5)+4f(0.75)+f(1)) = \frac{1}{12}(1+4\frac{16}{17}+2\frac{4}{5}+4\frac{16}{25}+\frac{1}{2}) \approx 0.785392$$

Detta närmevärde ligger väldigt nära integralens exakta värde som är $\frac{\pi}{4} \approx 0.785398$.

Matlab har ett redan inbyggt kommando för beräkning av integraler som heter `quad`. Detta bygger på Simpsons metod på ett ganska smart sätt genom att varje delintervall delas tills förändringen i integraluppskattningen är tillräckligt liten. `quad` använder en s.k. adaptiv metod där intervallindelningen succesivt bestäms utgående från beräkningsresultaten. Titta gärna på filen genom att ge kommandot;

```
>> type quad
```

eller öppna filen `quad.m` i editorn (sök efter filen i Matlabs eget programbibliotek). I princip fungerar `quad` på följande sätt: Beräkna integralen med Simpsons formel utan att dela intervallet. Dela sedan intervallet i två delar. Beräkna integralen igen och jämför med det tidigare värdet. Om ändringen är för stor så behandlas integralen över de två intervallhalvorna var för sig. Vi har då redan ett utgångsvärde, delar intervallet i två delar, beräknar med Simpsons formel och jämför. Om ändringen även nu är för stor så behandlas integralen över de två nya intervallhalvorna var för sig. Detta upprepas tills man fått tillräckligt liten ändring över samtliga delintervall. Syftet med den adaptiva metoden är att minska antalet beräkningssteg. Med beräkningssteg menar man oftast en multiplikation och en addition tillsammans. Detta kallar man också för en flyttalsoperation eller en s.k. *flop*. Ett mått på hur omfattande en kalkyl är ges av hur många *flops* som har utförts. I Matlab finns ett räkneverk `flops` som håller reda på hur många flyttalsoperationer som har utförts sedan räknaren nollställdes, vilket görs med `flops(0)`

Antag att vi vill beräkna integralen $\int_0^\pi x\sin x\,dx$ med hjälp av `quad`. Vi börjar då med att definiera $x\sin x$ som en `inline`-funktion;

```
>> f=inline('x.*sin(x)','x')
```

Integralen beräknas sedan med kommandot;

```
>> quad(f,0,pi)
```

Om vi vill bestämma precisionen i beräkningen kan vi t.ex. ange toleransen $10^{-3}$;

```
>> quad(f,0,pi,10^(-3))
```

En femte variabel ger möjlighet att se vilken intervallindelning som använts;

```
>> quad(f,0,pi,10^(-3),1)
```

**Uppgift 34:** Beräkna approximativt följande integraler med hjälp av metoden med rektangeluppskattningar (som användes på integralerna $\int_1^3 \ln x\,dx$ och $\int_1^\infty \frac{1}{1+x^2}\,dx$ i texten ovan)

(a) $\int_1^5 \sqrt{1+x^3}\,dx$ (b) $\int_0^1 \frac{\sin x}{x}\,dx$ (c) $\int_0^\infty e^{-x^2}\,dx$ □

**Uppgift 35:** Använd `quad` för att beräkna integralerna i uppgift 34 ovan. Undersök hur många *flops* som krävs för olika feltoleranser. □

**Uppgift 36:** Beräkna approximativt integralen $\int_0^2 \frac{x}{1+x}\,dx$ med hjälp av Simpsons formel, utan att använda `quad`. Jämför med integralens exakta värde. □

**Uppgift 37:** Vid beräkning av den generaliserade integralen $\int_1^\infty \frac{1}{1+x^2}\,dx$ i texten ovan, satte vi $b = \sqrt{n}$ i Riemannsummorna i (*). Pröva och se vad som händer om man istället sätter $b = n$. Kommer Riemannsummorna att konvergera mot integralens rätta värde ? Förklara! □

**Uppgift 38:** I avsnitt 3.1 studerade vi serien $\sum_{k=1}^\infty \frac{1}{1+k^2}$. På våra beräkningar där så verkade det som att denna serie var konvergent. Visa nu att integralen faktiskt är konvergent genom att verifiera integraluppskattningen $\sum_{k=1}^\infty \frac{1}{1+k^2} < \int_0^\infty \frac{1}{1+x^2}\,dx = \frac{\pi}{2}$ (se avsnitt 7.9 i boken 'Analys i en variabel').

I avsnitt 3.1 studerade vi även serien $\sum_{k=1}^\infty \frac{1}{\sqrt{1+k^2}}$. Med utgångspunkt från dessa beräkningar, vad tror du om integralen $\int_1^\infty \frac{1}{\sqrt{1+x^2}}\,dx$. Är den konvergent eller divergent ? □

# 4 Differentialekvationer

## 4.1 Riktningsfält

Differentialekvationen $y' = f(x, y)$ kan tolkas geometriskt genom att $y'$ är riktningskoefficienten för tangenten till kurvan $y = y(x)$. I punkten $(x, y)$ är alltså $f(x, y)$ tangentens riktningskoefficient. Genom att i många punkter $(x, y)$ rita en kort linje (eller pil) med riktningskoefficient $f(x, y)$ kan vi få en uppfattning om hur lösningskurvorna ser ut. I matlab kan man plotta sådana s.k. riktningsfält med kommandot `quiver`. Antag t.ex. att vi vill plotta riktningsfältet till differentialekvationen $y' = y - x^2$. Vi börjar då med att skapa en mängd av punkter i $xy$-planet i vilket vi vill rita riktningspilar;

```
>> [x y]=meshgrid(-2:0.2:2);
```

Sedan beräknar vi värdet på $y - x^2$ i alla dessa plottpunkter;

```
>> dy=y-x.^2;
```

och avslutar med att plotta riktningsfältet med kommandot;

```
>> quiver(x,y,ones(size(dy)),dy)
```

Differentialekvationen är linjär och kan lösas exakt. Låt oss plotta några lösningar i samma figur som riktningsfältet;

```
>> hold on, t=-2:0.01:1; plot(t,t.^2+2*t+2-exp(t),'r')
>> t=-2:0.01:1.8; plot(t,t.^2+2*t+2-1.5*exp(t),'r')
>> t=-2:0.01:1.5; plot(t,t.^2+2*t+2-2*exp(t),'r')
>> t=-2:0.01:0.7; plot(t,t.^2+2*t+2-3*exp(t),'r')
>> t=-2:0.01:0; plot(t,t.^2+2*t+2-4*exp(t),'r'), hold off
```

I riktningsfältet ovan varierar pilarnas längd med storleken på $y - x^2$ i respektive plottpunkt. Om vi vill att alla pilar i riktningsfältet skall ha samma längd ger vi istället följande kommando;

```
>> l=sqrt(1+dy.^2); quiver(x,y,1./l,dy./l)
```

**Uppgift 39:** Plotta riktningsfälten till följande differentialekvationer. Lös också ekvationerna för hand och plotta några lösningskurvor i samma figur som respektive riktningsfält.

(a) $(1 + x^2)y' - 2xy = (1 + x^2)\arctan x$ (plotta för $-1 < x < 3, -1 < y < 3$).

(b) $y' = \dfrac{y - y^3}{x^3 + x}$ (plotta för $-2 < x < 2, -3 < y < 3$). □

## 4.2 Numerisk lösning av första ordningens differentialekvationer

En metod för att hitta approximativa lösningar till differentialekvationer av typen $y' = f(x, y)$ ges av Eulers (polygon-) metod (se sid. 326-327 i boken 'Analys i en variabel'). Ide'n i denna metod är att succesivt stega sig fram med räta linjestycken mellan punkter $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$, där $(x_0, y_0)$ är given och linjestycket från $(x_k, y_k)$ till $(x_{k+1}, y_{k+1})$ har riktningskoefficient $f(x_k, y_k)$.

Matlab har många inbyggda kommandon för att hitta numeriska lösningar på differentialekvationer. En nackdel med dessa program är emellertid att funktionen måste vara definierad i en funktionsfil, `inline`-funktioner fungerar inte. Ett av dessa program är `ode45` som använder en förbättrad variant av Eulers metod som kallas Rugge-Kuttas metod (se under `ode45` i `helpdesk` för andra program av samma typ). Låt oss se hur vi använder `ode45` för att hitta approximativa lösningar till differentialekvationen $y' = y - x^2$. Börja då med att öppna Matlabeditorn och skapa följande funktionsfil;

```
% funk.m
function z=funk(x,y)
z=y-x.^2;
```

Om vi vill hitta den lösning till differentialekvationen på intervallet $[-2, 2]$ som går genom punkten $(-2, 1.8)$ så ger vi t.ex. kommandot;

```
>> [x y]=ode45('funk',[-2 2],1.8);
```

För att plotta lösningen ger vi sedan kommandot;

```
>> plot(x,y)
```

Låt oss jämföra med den exakta lösningen;

```
>> hold on, t=-2:0.01:2; plot(t,t.^2+2*t+2-0.2*exp(t+2),'r'), hold off
```

Eftersom kurvorna i stort sett sammanfaller så är det som vanligt en bättre ide' att plotta skillnaden mellan kurvorna;

```
>> plot(x,y-(x.^2+2*x+2-0.2*exp(x+2)))
```

Vi ser då att skillnaden huvudsakligen ökar ju längre bort från begynnelsepunkten vi stegat oss.

Programmet `ode45` kan också lösa system av differentialekvationer. Detta ger oss då även möjlighet att lösa högra ordningens ekvationer. För att lösa ekvationssystemet $\begin{cases} y_1' = xy_1 + y_2 \\ y_2' = y_2 - y_1 \end{cases}$ så måste vi först skriva följande funktionsfil;

```
% syst.m
function z=syst(x,y)
z=[x.*y(1)+y(2); y(2)-y(1)];
```

Om vi sedan t.ex. vill hitta de lösningskurvor $y = y_1(x)$ och $y = y_2(x)$, på intervallet $0 \leq x \leq 1$, som uppfyller $y_1(0) = 1$ och $y_2(0) = -2$ så ger vi kommandot;

```
>> [x y]=ode45('syst',[0 1],[1;-2]);
```

Kurvorna plottas avslutningsvis med kommandot;

```
>> plot(x,y(:,1),x,y(:,2))
```

**Uppgift 40:** Bestäm några approximativa lösningar till differentialekvationerna i uppgift 39 ovan, med hjälp av `ode45` (välj själv lämpliga begynnelsevillkor). Hur mycket avviker dessa från de exakta lösningarna ? □

**Uppgift 41:** Gör om ekvationen $y'' + 2y' + 2y = \cos t,\ y(0) = 1,\ y'(0) = -2$ till ett system av första ordningens differentialekvationer (sätt $y_1 = y$ och $y_2 = y'$). Ändra sedan i filen `syst.m`, från texten ovan, så att du kan lösa detta system med `ode45` i intervallet $0 \leq t \leq 10$. Plotta lösningskurvorna . Vilken av dem är $y(t)$ ? Vad illustrerar den andra kurvan ? Jämför med exakta lösningen. □

## 4.3 Svängningsekvationen

En av de viktigaste differentialekvationerna är svängningsekvationen $my''(t)+dy'(t)+ky(t) = f(t)$. Denna ekvation uppträder då man studerar svängningar i en sträng, en stav, en elektrisk krets m.m. (se även avsittet 'Ett svängningsproblem' på sid 347-349 i boken 'Analys i en variabel'). En enkel situation är dämpad fjädersvängning, då $m$ är en massa som är kopplad till ett fast underlag via en fjäder med fjäderkonstant $k$ och en dämpare med dämpkonstant $d$. Svängningen drivs av en kraft $f(t)$ som verkar på massan. Vi skall nu utnyttja `ode45` för att något analysera svängningsekvationen. För enkelhets skull antar vi att $m = 1$.

Börja med att hämta funktionsfilen `svfun.m` från kursens hemsida (under kursmaterial). Med hjälp av denna funktionsfil kan vi nu studera lösningarna på ekvationen $y''(t) + dy'(t) + ky(t) = \sin at$, för olika dämpningskonstanter $d$, fjäderkonstanter $k$ och frekvenser $a$. Funktionen `svfun` använder sig av de globala variablerna `FJADER`, `FREKVENS` och `DAMPNING`, så innan vi kan börja använda funktionen måste vi deklarera dessa variabler som globala i Matlabfönstret;

```
>> global FJADER FREKVENS DAMPNING
```

Om vi t.ex. vill studera fallet $d = 0.1, k = 4$ och $a = 1$ så gör vi följande tilldelning i Matlabfönstret;

```
>> DAMPNING=0.1; FJADER=4; FREKVENS=1
```

Sedan löser vi motsvarande differentialekvation, med begynnelsevillkoren $y(0) = 1, y'(0) = 2$, med kommandot;

```
>> [t y]=ode45('svfun',[0 50],[1;2]);
```

och plottar lösningen;

```
>> plot(t,y(;,1))
```

Fortsätt nu att analysera svängningsekvationen genom att göra följande övningsuppgift. Försök förklara dina observationer matematiskt och relatera till situtionen med dämpad fjädersvängning.

**Uppgift 42:**

(a) Välj $d = 1$ och drivfunktionen $f(t) = 0$ (det får man med `FREKVENS=0`). Undersök hur lösningskurvans utseende varierar med $k$. Rita grafen till $y(t)$ för $0 \leq t \leq 100$. Hur många max/min har kurvan ? Finns det några "magiska" gränser för detta ? Kanske det finns skäl att skriva en snurra som gör detta effektivt. Annars kan du skriva en enkel skriptfil där du kan ändra $k$, köra `ode45` och rita grafen.

(b) Välj $k = 4, d = 0.1$ och undersök hur lösningskurvans utseende varierar med $a$. Lösningen blir så småningom som en ren sinusfunktion. För vilket $a$ har denna störst amplitud ? Kan du upptäcka några roliga fenomen på vägen mot bästa $a-$värdet ? Vänd på problemet: välj $a$ och variera $k$ tills du får maximal amplitud.

(c) I stället för konstant $k$-värde kan vi låta $k$ vara en funktion av någon av de andra variablerna, $t, y$ eller $y'$. Vi behöver då bara göra några små modifieringar av filen `svfun.m`. Om vi t.ex. vill att $k$ skall bero på tiden $t$ enligt ett samband $k = k(t)$ så byter vi ut raden `k=FJADER` mot t.ex. raden `k=fjader(t)` där `fjader` är en funktionsfil för funktionen $f(t)$. Istället för att göra en separat funktionsfil `fjader.m` så kan vi skriva in den sist i m-filen `svfun` (ta bort `%`-tecknet på de sista två raderna i filen och lägg till själva beräkningsdelen i funktionen `fjader`). Vi kan göra på ett liknande sätt om vi vill låta $k$ vara lägesberoende resp. hastighetsberoende.

Låt $k = 10$ om $y \geq 0$ och $k = 0.01$ om $y < 0$. Låt $a = 2$ och variera $d$ mellan 0.5 och 0.05. Vad händer ? □