Laborationstillfälle 1 – Lite mer om MATLAB och matematik

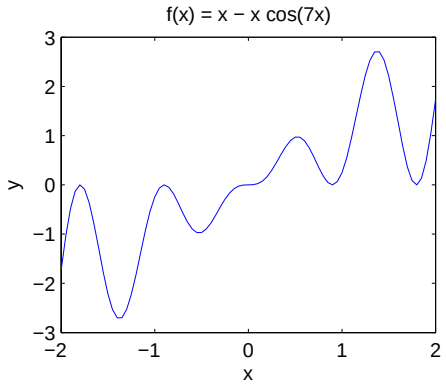
En första introduktion till MATLAB har ni fått under kursen i inledande matematik. Vid behov av repetition kan materialet till de övningar som gjordes då hämtas på den kursens hemsida:
www.math.chalmers.se/Math/Grundutb/CTH/tmv120/0506/
Vidare man man via helpdesken i MATLAB finna en introduktion **Getting Started**. Under helpdesken finns också en utförlig dokumentation av hela MATLAB. Boken **Användarhandledning till** MATLAB av Pärt-Enander och Sjöberg ger en lättläst och utförlig introduktion.

Under denna matematikkurs, liksom under kursen i linjär algebra i läsperiod 3, kommer du använda MATLAB för att lösa laborationsuppgifter med inriktning mot numeriska beräkningar vid lösning av matematiska problem. Du kommer då även lära dig mer om MATLAB. Mycket av inlärningen blir via exempel.

Målsättning vid labtillfälle 1: Att få större vana vid MATLAB. Arbeta igenom så mycket av detta material som möjligt, så att vi lättare klarar av nästa labtillfälle. Då skall vi göra laborationsuppgift 1. Lägg inte för mycket tid på övningsuppgift 2, den är främst med för att det blir roliga kurvor.

Exempel 1. Vi kan rita grafen till $f(x) = x - \cos(7x)$ över intervallet $-2 \leq x \leq 2$ med MATLAB enligt följande

```
>> x=linspace(-2,2,100);
>> y=x-x.*cos(7*x);
>> plot(x,y)
>> xlabel('x'), ylabel('y'), title('f(x) = x - x cos(7x)')
```



Det här exemplet kräver en förklaring. Vi börjar med att göra en vektor (eller radmatris) med 100 värden jämnt fördelade mellan -2 och 2. Funktionen **linspace** är gjord för detta ändamål. Alternativt skulle vi kunna använda kolonoperatorn (**:**). Sedan bildar vi en vektor med motsvarande funktionsvärden. Observera den elementvisa eller punktvisa multiplikationen (**.***), som gör att komponenterna i vektorn $\mathbf{y}$, $y_i = f(x_i) = x_i - x_i\cos(7x_i)$ för de olika x_i-värdena i vektorn $\mathbf{x}$.

Övning 1. Rita grafen av funktionen $f(x) = 1 + (2x^2 + 5x + 2)\exp(-x)$ över intervallet $-3 \leq x \leq 6$.

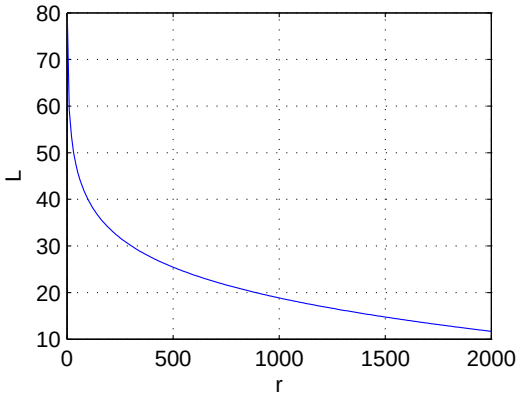
Lösning till övningen finner du på sidan 4.

Nu följer ett exempel med tillämpningsbakgrund. Vi kan bortse ifrån akustiken nu och bara se på matematiken.

Exempel 2. Ljudnivån L (i decibel) på avståndet r meter från en ljudkälla ges av formeln $L = L_0 - 20\,^{10}\log(r) - \beta r$, där L_0 är ljudnivån 1 meter från ljudkällan och β är en parameter som anger hur ljudet försvagas då det går genom luften.

Vi skall för $L_0 = 80$ dB bestämma grafiskt ungefär det avstånd där ljudnivån är 20 dB då $\beta = 1.15 \cdot 10^{-3}$.

```
>> r=linspace(1,2000,200);
>> L=80-20*log10(r)-1.15e-3*r;
>> plot(r,L), xlabel('r'), ylabel('L')
>> grid on
```

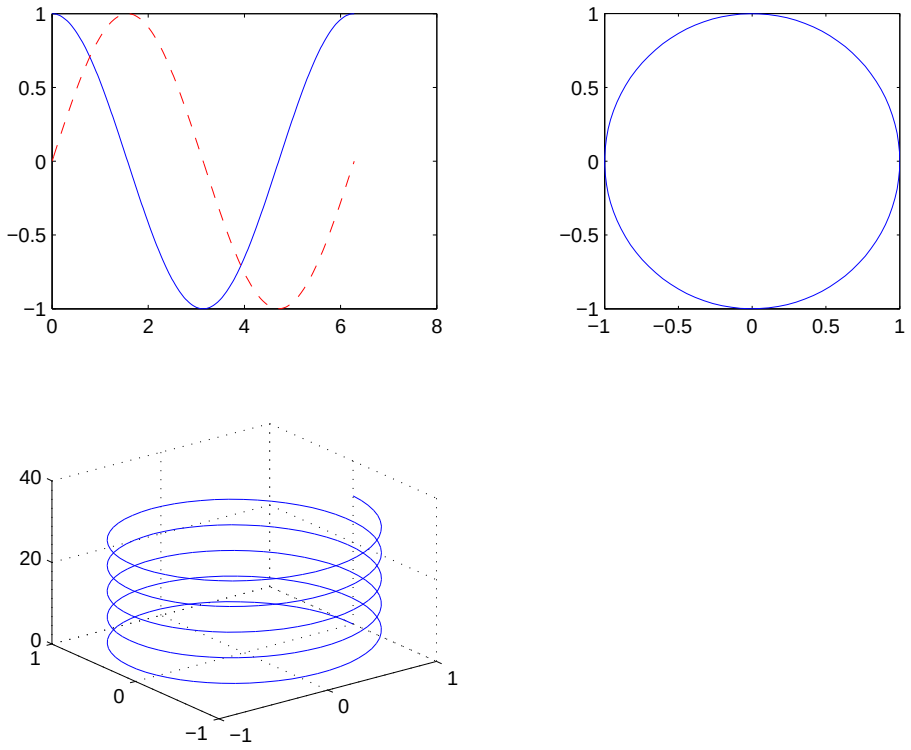



```
>> p=ginput(1);
>> avst=p(1)
avst =
  895.1613
```

Vi ritade grafen, därefter fick vi ett rutnät med **grid**. Nu kan vi se ungefär var ljudnivån är 20 dB, men vi får en noggrannare avläsning med **ginput**. Efter vi har givit detta kommando blir markören ett hårkors som vi placerar över en punkt vi vill läsa in. Ett tryck på vänster musknapp gör att koordinaterna hamnar i vektorn **p**, vars första komponent innehåller den horisontella koordinaten medan den andra komponenten innehåller den vertikala koordinaten. Senare kommer vi se hur vi skulle kunna använda **fzero** för att få en ännu noggrannare bestämning av avståndet. Observera att $1.15 \cdot 10^{-3}$ skrivs **1.15e-3** och att $^{10}\log$ (tio-logaritmen) ges av **log10**.

Exempel 3. Nu skall vi i samma figur göra tre olika koordinatsystem. I det första skall vi rita $\sin(x)$ och $\cos(x)$, i det andra skall vi rita kurvan $t \mapsto (x(t), y(t)) = (\cos(t), \sin(t))$ för $0 \leq t \leq 2\pi$ och i det tredje skall vi rita spiralen $t \mapsto (x(t), y(t), z(t)) = (\cos(t), \sin(t), t)$ för $0 \leq t \leq 10\pi$.

```
>> subplot(2,2,1), x=linspace(0,2*pi,200); plot(x,cos(x),x,sin(x),'r--')
>> subplot(2,2,2), t=linspace(0,2*pi,100); plot(cos(t),sin(t)), axis('square')
>> subplot(2,2,3), t=linspace(0,10*pi,300); plot3(cos(t),sin(t),t), grid on
```



Med **subplot** delar vi upp i flera koordinatsystem, första talet anger antal koordinatsystem ovanför varandra, andra talet hur många bredvid varandra och det tredje talet vilket som skall var aktivt för ritning (numreringen görs som vi läser). Vi fick graferna av både $\sin(x)$ och $\cos(x)$ med **plot** genom att skicka med fyra vektorer, dels

paret **x, cos(x)** dels paret **x, sin(x)**. Sinus-grafen blir röd med (**r**) och streckad med (**- -**). Observera att vi kan göra vektorer med funktionsvärden på 'plats' i anropet av **plot**.

Med **axis('square')** får vi ett kvadratiskt koordinatsystem, så att cirkeln inte blir tillplattad.

När vi ritar en kurva i rummet används **plot3** på samma sätt som **plot**.

Övning 2. Rita kurvan

$$t \mapsto (x(t), y(t)) = (4\cos(-\frac{at}{4}) + 7\cos(t), 4\sin(-\frac{at}{4}) + 7\sin(t))$$

med $0 \leq t \leq 18\pi$ för t.ex. $a = 111, 112, 113$ och 117.

Vi ska nu se hur **for**-satsen fungerar i MATLAB.

Exempel 4. Man kan beräkna $\sqrt{c}$ med upprepade additioner och divisioner med iterationsformeln

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{c}{x_k}\right),\ k = 0, 1, 2, \cdots,\ x_0 = c$$

Vi skall beräkna en approximation av $\sqrt{2}$ med iterationsformeln. Iterationen avbryts efter $k_{max} = 20$ iterationer eller då $|x_{k+1} - x_k| < 10^{-6}$.

```
>> x=2
>> for k=1:20
      xny=0.5*(x+2/x);
      if abs(x-xny)<1e-6
         break
      end
      x=xny;
   end
>> x
x =
    1.4142
```

Först får **x** värdet 2, motsvarar $x_0 = c$, sedan får **xny** värdet **0.5*(x+2/x)**, motsvarar $x_1 = \frac{1}{2}(x_0 + \frac{c}{x_0})$. Inför nästa repetition måste vi ge **x** värdet **xny** så att satsen **xny=0.5*(x+2/x)** motsvarar $x_2 = \frac{1}{2}(x_1 + \frac{c}{x_1})$ osv.

Med **if**-satsen testar vi om $|x_{k+1} - x_k| < 10^{-6}$, är så fallet avbryts **for**-satsen med **break**. Som ni förstår ger **abs** absolutbeloppet. Lägg märke till att räckvidden av **for** och **if** begränsas av ett **end**.

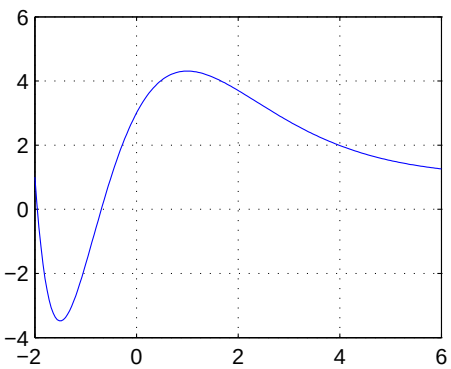
Exempel 5. Vi skall finna nollställena till funktionen $f(x) = 1 + (2x^2 + 5x + 2)\,e^{-x}$ vars graf vi ritade i övning 1.

Vi skapar en MATLAB-funktion med namnet `funex6.m`

```
function f=funex6(x)
f=1+(2*x.^2+5*x+2).*exp(-x);
```

Ritar grafen med

```
>> x=linspace(-2,6,200);
>> plot(x,funex6(x))
>> grid on
```



Vi ser att det högra nollstället tycks ligga i intervallet $-1 \leq x \leq 0$. Läser in en approximativ rot.

```
>> p=ginput(1); x0=p(1)
x0 =
   -0.6524
```

Sedan finjusterar vi approximationen med `fzero`

```
>> x=fzero(@funex6,x0)
x =
   -0.6914
```

Nu är det dags att rita om grafen så vi tydligare ser nollstället i intervallet $-2 \leq x \leq -1$ och upprepa beräkningarna för denna rot. Detaljerna lämnas som övning.

Vår funktion **funex6** kan användas som de inbyggda funktionerna (**sin**, **cos** osv.). Lägg märke till att funktionen **fzero** får med en pekare till vår funktion **funex6**, den måste ju 'veta' vilken funktion som nollställe skall bestämmas för. Filen med beskrivningen av **funex6** skapar vi med MATLABs inbyggda editor. Gå in under **File** och välj **New** och sedan **M-File**.

Övning 3. En lång stång upphettas momentant vid tiden $t = 0$ i mittpunkten. Med denna punkt i $x = 0$ och x-axeln utmed stången bestäms temperaturen $T(x,t)$ för $t > 0$ av uttrycket

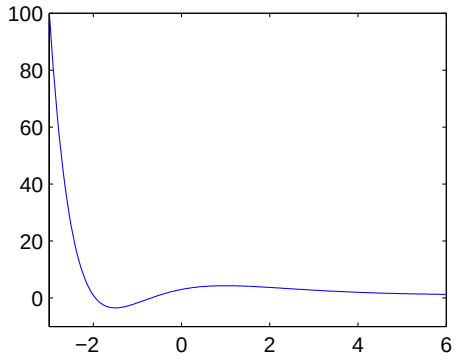
$$T(x,t) = Ce^{-x^2/kt}/\sqrt{t}$$

där parametrarna $k = 5.17$ och $C = 28.3$ beror av värmeledningsförmågan respektive den tillförda värmen.

Beräkna under hur lång tid som temperaturen vid $x = 1$ överstiger 20°.

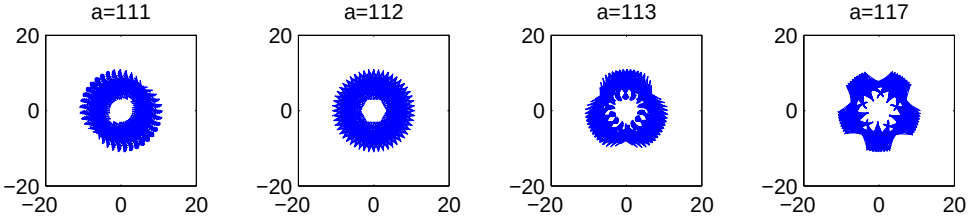
Lösningar till övningsuppgifter

1.
```
>>x=linspace(-3,6,200);
>>plot(x,1+(2*x.^2+5*x+2).*exp(-x)
>>axis([-3,6,-10,100])
```



   Kommandot **axis** kan användas för att ge önskad variation i horisontal- respektive vertikalled.

2.
```
>> avekt=[111,112,113,117]; T=18*pi; t=linspace(0,T,600);
>> for i=1:length(avekt)
      a=avekt(i); x=4*cos(-a*t/4)+7*cos(t); y=4*sin(-a*t/4)+7*sin(t);
      subplot(1,4,i), plot(x,y), axis('square'), title(['a=',num2str(a)])
   end
```



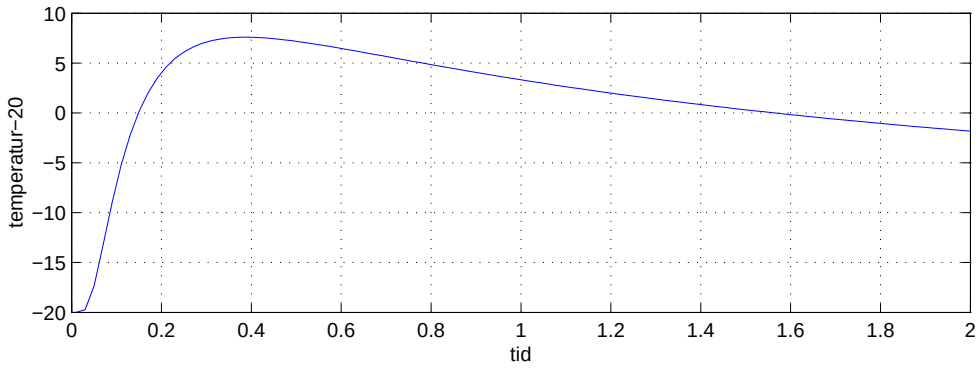
   Kommandot **length** ger antal element i en vektor och **num2str** omvandlar ett tals värde till en textsträng. Observera hur textsträngarna limmas ihop till en lång sträng med hjälp av [ ], t.ex. [**'hej','san'**] blir **'hejsan'**.

3. Vi skapar en fil med beskrivning av vår funktion

```
function f=varm(t)
k=5.17; C=28.3; x=1;
f=C*exp(-x^2/k./t)./sqrt(t)-20;
```

och ritar graf med

```
>> t=linspace(0.01,2,100); plot(t,varm(t)), grid
>> xlabel('tid'), ylabel('temperatur-20')
```



Sedan läser vi in startapproximationer samt finjusterar dessa med

```
>> [t0,T0]=ginput(1); t0=fzero(@varm,t0);
>> [t1,T1]=ginput(1); t1=fzero(@varm,t1);
>> tiden=t1-t0
tiden =
   1.4145e+00
```

Här använder vi **ginput** på ett lite annorlunda sätt. Den första variabeln på vänster sida kommer innehålla x-koordinaten (tiden) och den andra y-koordinaten.

Kommandon och funktioner i MATLAB som vi använt.

plot, **plot3** – rita vektorer mot varandra parvis, i planet respektive rummet.
axis – styr hur koordinatsystem skall se ut.
grid – lägger rutnät på grafer.
ginput – grafisk inmatning av talpar.

abs, **cos**, **sin**, **exp**, **log10**, **sqrt** – standard elementära funktioner.
linspace – bildar en vektor av ett antal jämnt fördelade tal.
length – ger antalet element i en vektor.

subplot – delar upp i flera koordinatsystem.
num2str – omvandla tal till textsträng
xlabel, **ylabel**, **title** – sätter text på axlar samt rubrik.

for, **if**, **break**, **function** – språkelement.

fzero – finner nollställe till en funktion.

clc, **clf** – suddar **Command Window** respektive suddar **Figure**, dvs. det aktiva grafikfönstret.

För att få reda på mer om ett kommando eller en funktion, t.ex. **plot** gör **help plot** i **Command Window**.

En **function**-fil i MATLAB är en s.k. **m**-fil och skapas med den inbyggda editorn (eller med en extern). Filen skall då inledas med **function**, se exempel 6. En **m**-fil kan också vara en **script**-fil, denna inleds inte med **function**, och är en samling av kommandon som utförs när du ger filens namn som kommando i **Command Window**. Man kan också 'köra' filen genom att välja **Run** under **Debug** i menyraden.
