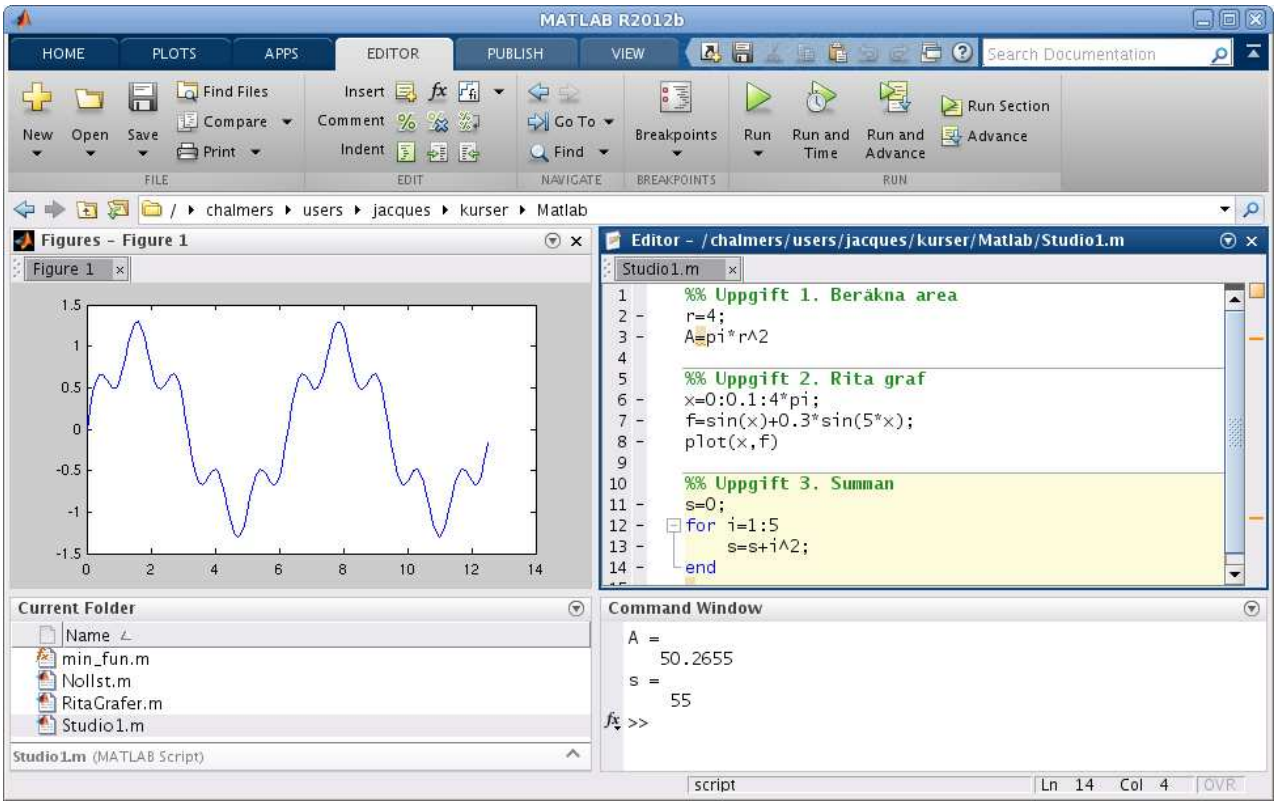
Funktioner och grafritning i MATLAB

1 Inledning

Först skall vi se lite på matriser, vilket är den grundläggande datatypen i MATLAB. Sedan ser vi lite på (elementära) matematiska funktioner i MATLAB, som sinus och cosinus. Därefter ser vi på grafritning och hur vi definerar egna funktioner. Avslutningsvis ser vi lite på ritning av allmännare kurvor.

Men allra först: För att arbeta på ett överskådligt och effektivt sätt, dokumentera arbetet och redovisa för handledare snabbt och smidigt skall ni använda er av script (som vi såg på i första studioövningen).

Så här kunde det sett ut efter det att vi löst de tre första uppgifterna i den studioövningen.



Editorn i MATLAB har använts i Cell Mode (cell-läge). Skriver man en kommentar som börjar med två procent-tecken, så avgränsar det en cell. Poängen är att man kan låta MATLAB utföra kommandona från en cell, istället för hela filen.

På så sätt kan man dela upp ett stort script (för en hel studioövning) i flera delar (varje deluppgift). En cell kan utföras utan att textfilen är sparad, så gör Save då och då. Har du glömt bort script och cell-läge repetera i så fall studioövning 1, avsnitt 4.

Vi har också gjort en egen desktop layout så att Figures (figurfönster), Editor samt Command Window syns samtidigt. Läs i texten för studioövning 1, avsnitt 7 hur man gör.

2 Något om matriser

Matematikkursen heter ”Analys och linjär algebra”. Grundstenen i linjär algebra är matrisbegreppet. Matriser är även den grundläggande datatypen i MATLAB. Den linjära algebran behandlas framför allt under läsperioderna 2 och 3, men vi behöver matrisbegreppet redan nu.

En matris är ett rektangulärt talschema

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Matrisen ovan har m rader och n kolonner, vi säger att den är av typ $m \times n$. Ett matriselement i rad nr i, kolonn nr j tecknas a_{ij}, där i är radindex och j är kolonnindex. I MATLAB skrivs detta `A(i,j)` och `size(A)` ger matrisens typ.

En matris av storleken $m \times 1$ kallas kolonnmatris (kolonnvektor) och en matris av storleken $1 \times n$ kallas radmatris (radvektor):

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_n \end{bmatrix}$$

Element nr i ges i MATLAB av `b(i)`, `c(i)` och antalet element ges av `length(b)`, `length(c)`. Som exempel tar vi

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 & 2 & 4 & 6 & 8 \end{bmatrix}$$

Vi beskriver dessa i MATLAB enligt

```
>> A=[1 4 7 10; 2 5 8 11; 3 6 9 12]
```

och som svar får vi i Command Window utskriften

```
A =
     1     4     7    10
     2     5     8    11
     3     6     9    12
```

Man använder hakparanteser (`[ ]`) för att bygga upp matriserna. Semikolon (`;`) innanför hakparanteserna betyder radbyte.

Så här beskriver vi kolonnvektorn

```
>> b=[1; 3; 5]
b =
     1
     3
     5
```

och så här radvektorn

```
>> c=[0 2 4 6 8]
c =
     0     2     4     6     8
```

Uppgift 1. Skriv in matriserna **A**, **b** och **c** i MATLAB. Skriv sedan ut matriselementen a_{23}, b_2, c_3. Ändra a_{23} genom att skriva `A(2,3)=15`. Gör ett script och använd cell-läge så att ni kan bygga på med kommande uppgifter.

Ibland vill vi se en tabell som en matris. Som exempel tar vi: Värmeförlusten hos den som vistas i kyla beror inte enbart på temperaturen, utan även på hur mycket det blåser. Tabellen visar vilken *effektiv temperatur* det blir vid olika temperaturer T (°C) och vindhastigheter v (m/s).

v \ T	10	6	0	-6	-10	-16	-26	-30	-36
2	9	5	-2	-9	-14	-21	-33	-37	-44
6	7	2	-5	-13	-18	-26	-38	-44	-51
10	6	1	-7	-15	-20	-28	-41	-47	-55
14	6	0	-8	-16	-22	-30	-44	-49	-57
18	5	-1	-9	-17	-23	-31	-45	-51	-59

Om vi ville göra något med dessa data i MATLAB så skulle vi lagra temperaturer och vindhastigheter i rad- eller kolonnvektorer och effektiva temperaturerna i en matris. (Vill du läsa mer om värmeförlust kan du gå till SMHI:s hemsida och söka på vindavkylning.)

Radvektorn **c** från vårt exempel ovan har samma avstånd mellan talen i elementen och kan därför även bildas med

```
>> c=0:2:8
c =
     0     2     4     6     8
```

Här är 0 första värdet (startvärde), 2 är avstådet till nästa tal (steg) och 8 är det sista värdet (slutvärde), dvs. vi har strukturen

```
variabel=startvärde:steg:slutvärde
```

Detta sätt att bilda en vektor kallas *kolon-notation* och är enklare att använda (om det är möjligt) då vi har många element i vektorn. Utelämnar man steg så blir steget 1 som standardvärde.

Vi kan även bilda en radvektor med den inbyggda funktionen `linspace`. Detta fungerar ungefär på samma sätt som med kolon-notationen, men med den skillnaden att man inte ger steget eller avståndet mellan värdena, utan man ger det totala antalet värden man vill ha jämnt fördelade mellan ett start- och ett slutvärde enligt

```
variabel=linspace(startvärde,slutvärde,antal)
```

Man anger alltså hur många element vi skall ha. Utelämnar man antal så får man 100 som standardvärde. I samband med grafritning i förra studioövningen bildade vi radvektorer med x-värden på detta sätt.

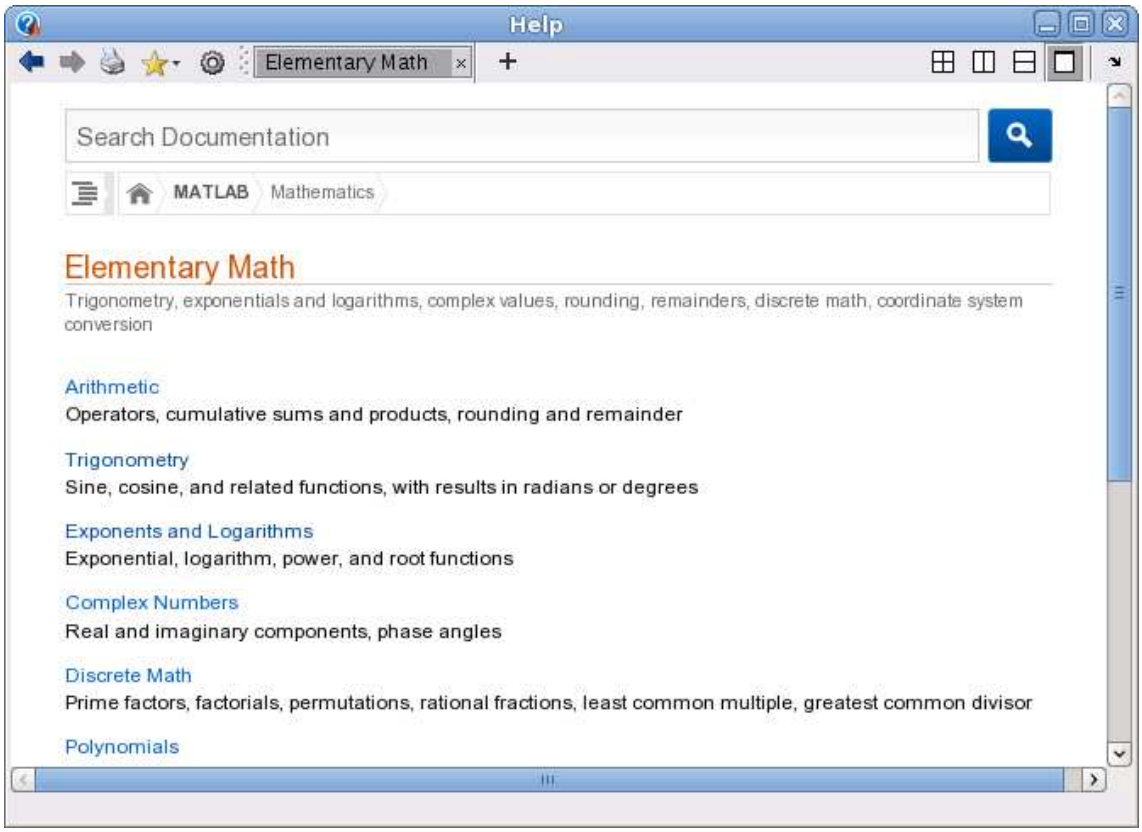
Att använda `linspace` är det effektivaste sättet vid grafritning då vi har ett intervall vi är intressede av (start- och slutvärde) och vill bara ha tillräckligt många punkter (antal) i intervallet för att kunna rita en jämn och snygg graf. T.ex.

```
>> x=linspace(0,10*pi,400);
```

ger en vektor som ger 400 värden jämnt fördelade i intervallet $0 \leq x \leq 10\pi$.

3 Elementära funktioner

Vi letar upp hjäptexterna för elementära eller matematiska funktioner i Help genom att successivt öppna MATLAB, Mathematics och sedan Elementary Math.



Vi ser att funktionerna är grupperade, t.ex. en grupp med trigonometriska funktioner och en grupp med exponent- och logaritmfunktioner.

Funktioner som exempelvis sinus och cosinus, kan operera både på enskilda tal och på matriser. Man får som resultat en matris av samma storlek, vars element är funktionsvärdet av respektive element i argumentet.

Som exempel tar vi radmatrisen (radvektorn) $\mathbf{x} = (0, 0.1, 0.2, 0.3, 0.4, 0.5)$ som vi skriver in i MATLAB enligt

```
>> x=0:0.1:0.5
x =
         0    0.1000    0.2000    0.3000    0.4000    0.5000
```

Nu beräknar vi $\mathbf{y}$ som är sinus av radvektorn $\mathbf{x}$ med

```
>> y=sin(x)
y =
         0    0.0998    0.1987    0.2955    0.3894    0.4794
```

Här blir `sin(x)` en radvektor eftersom `x` var en radvektor.

Uppgift 2. Leta upp hjälptexten till tangens, som ju är kvoten mellan sinus och cosinus. Rita upp tangensfunktionen (som heter `tan` i MATLAB) enligt exemplet i hjälptexten. Repetera hur du startar hjälpverktyget (studioövning 1, avsnitt 8), om du har glömt. Varför ritar man grafen över intervallet $-\frac{\pi}{2} + s \leq x \leq \frac{\pi}{2} - s$, där s är ett litet positivt tal?

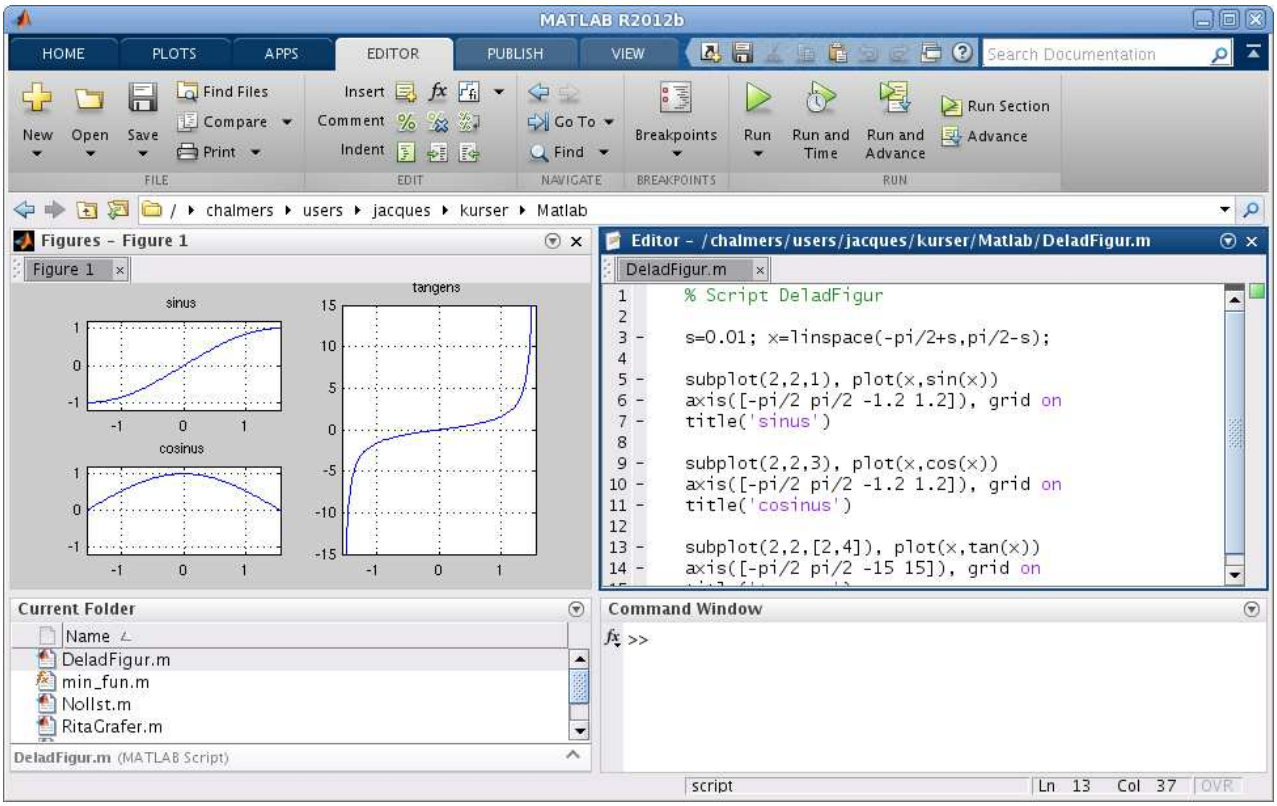
4 Funktionsgrafer

Ibland vill man rita flera grafer i samma koordinatsystem. Efter att ha ritat första grafen ger man kommandot `hold on` för att bevara den, sedan kan man rita fler grafer ovanpå tills man tar bort skyddet med `hold off`. Vi påminner oss att vi kan lägga på ett rutnät med `grid on` och ta bort det igen med `grid off`, om vi vill det. Med `xlabel` och `ylabel` kan vi sätta texter på axlarna och med `title` kan vi sätta rubrik på koordinatsystemet. Allt detta har vi redan gjort, kommer du inte ihåg det är det kanske läge att kort repetera (studioövning 1, avsnitt 3).

Ibland vill man ha flera koordinatsystem i samma figur-fönster (Figure). Då använder man kommandot `subplot`. Vi ser på ett exempel.

Exempel 1. Vi skall i samma figur göra tre olika koordinatsystem. I dessa skall vi rita graferna av $\sin(x)$, $\cos(x)$ respektive $\tan(x)$ över intervallet $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$.

Så här kommer det se ut



Vi ser lite på programkoden

```
>> s=0.01; x=linspace(-pi/2+s,pi/2-s);
>> subplot(2,2,1)          % delar in Figure i 2x2 delar och gör 1:a aktiv
>> plot(x,sin(x))
>> axis([-pi/2 pi/2 -1.2 1.2]), grid on, title('sinus')
```

Den första 2:an i `subplot` förbereder för två rader av koordinatsystem och den andra förbereder för två kolonner av koordinatsystem. Dessa numreras stigande vänster till höger, uppifrån och nedåt. Vi anger att det 1:a systemet skall vara aktivt och där hamnar grafen av sinus.

```
>> subplot(2,2,3)          % delar in Figure i 2x2 delar och gör 3:e aktiv
>> plot(x,cos(x))
```

```
>> axis([-pi/2 pi/2 -1.2 1.2]), grid on, title('cosinus')
>> subplot(2,2,[2,4])       % samma indelning men gör 2:a och 4:e aktiva
>> plot(x,tan(x))
>> axis([-pi/2 pi/2 -15 15]), grid on, title('tangens')
```

I det 3:e systemet ritade vi grafen av cosinus. Eftersom grafen av tangens behöver få sträcka sig ganska mycket vertikalt, fogar vi samman det 2:a och 4:e systemet, genom att bilda vektorn `[2,4]`, och där ritar vi sedan grafen an tangens.

Kommandot `axis` använder vi när vi inte nöjer oss med de skalor på axlarna som vi får automatiskt. För t.ex. tangens vill vi ha intervallet $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$ horisontellt och vertikalt blir intervallet $-15 \leq y \leq 15$ rätt lagom. Vi har alltså vertikalt skurit bort en bra bit av grafen för att få en snygg bild.

Exempel 2. Rita grafen till $f(x) = x\sin(x)$ över intervallet $0 \leq x \leq 8$.

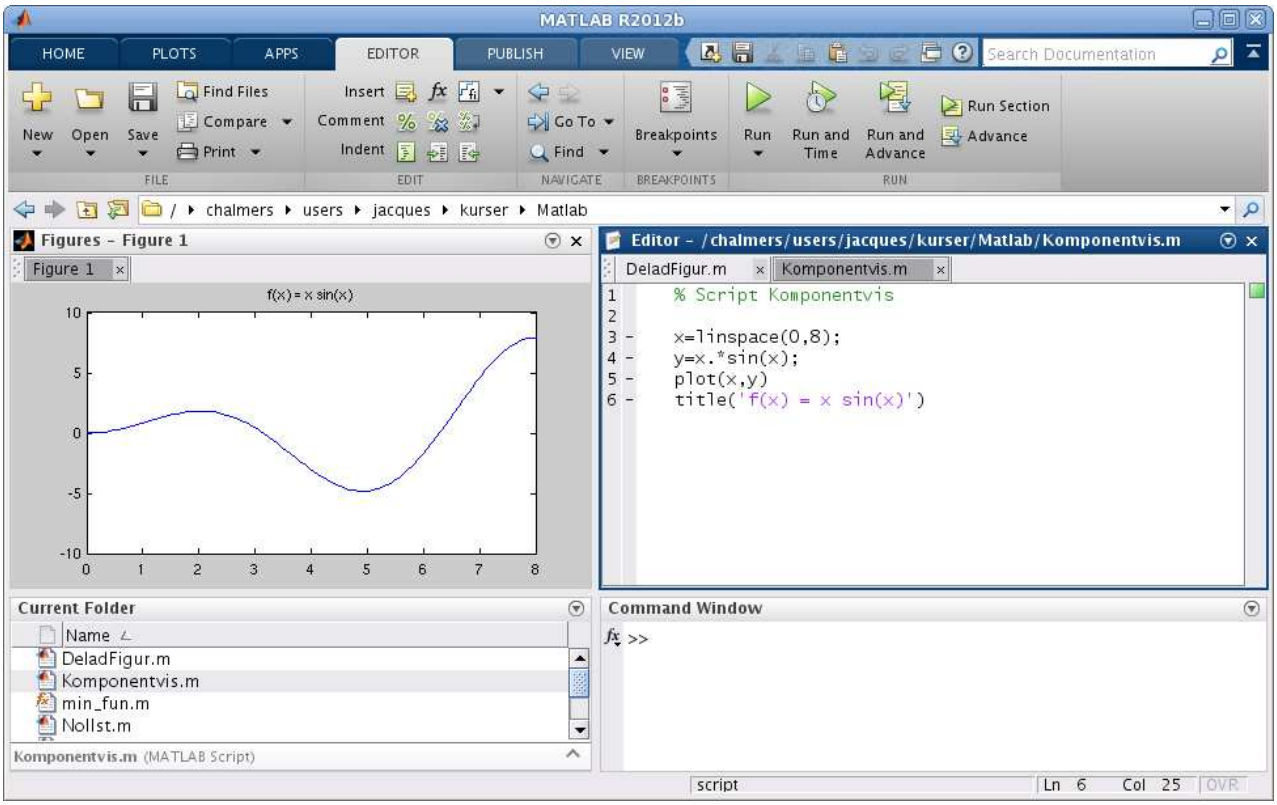
Vi bildar en vektor $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ med värden jämnt fördelande över intervallet $0 \leq x \leq 8$. Sedan bildar vi vektorn

$$\mathbf{y} = (f(x_1), f(x_2), \cdots, f(x_n)) = (x_1\sin(x_1), x_2\sin(x_2), \cdots, x_n\sin(x_n))$$

och ritar upp grafen. För att bilda vektorn $\mathbf{y}$ behövs komponentvis multiplikation som skrivs `.*` i MATLAB, vi vill ju att $y_i = f(x_i) = x_i\sin(x_i)$ för alla $i = 1, 2, \cdots, n$. Vi ritar grafen med

```
>> x=linspace(0,8);
>> y=x.*sin(x);
>> plot(x,y)
>> title('f(x) = x sin(x)')
```

och så här ser resultatet ut



Uppgift 3. Rita grafen till $f(x) = x - x\cos(7x)$ över intervallet $0 \leq x \leq 8$. Tänk på att använda komponentvis multiplikation.

5 Egna funktioner

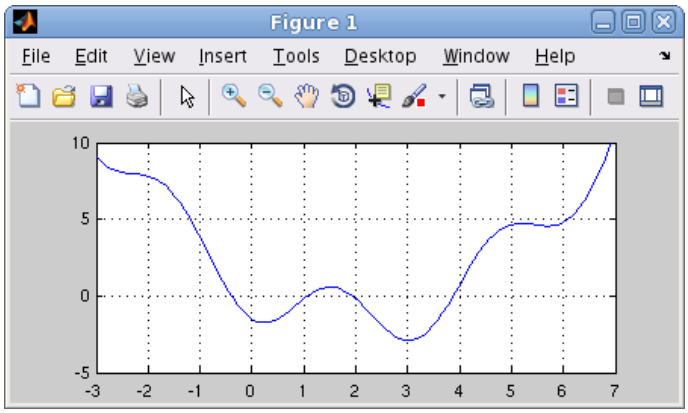
I senare studioövningar skall vi se på beräkningsmetoder för att lösa ekvationer av typen $f(x) = 0$, dvs. söka nollställen till en funktion f.

Som exempel kan vi ta

$$f(x) = 0.5\,(x-2)^2 - 2\,\cos(2x) - 1.5 = 0$$

Det vi alltid kommer börja med är att rita grafen till f för att få en uppfattning om hur många nollställen vi har och ungefär var de ligger.

```
>> f=@(x)0.5*(x-2).^2-2*cos(2*x)-1.5;
>> x=linspace(-3,7);
>> plot(x,f(x))
>> axis([-3 7 -5 10]), grid on
```



Här införde vi en anonym funktion (anonymous function) med ett funktionshandtag (function handle) enligt

```
handtagsnamn = @(parametrar) sats
```

Här är delen `@(parametrar) sats` den anonyma funktionen och `handtagsnamn` är det namn vi väljer på funktionshandtaget som kopplas till funktionen. Med `parametrar` avser vi indata till funktionen, ofta en variabel ibland flera.

I denna konstruktion är det bara tillåtet med en enda beräkningssats. En mer komplicerad funktion (som kan bestå av flera beräkningssatser) kräver att vi definierar en funktion (function).

Exempel 3. Kastbana utan luftmotstånd beskrivs av

$$y(x) = y_0 - \frac{g}{2v_0^2\cos^2(\theta)}\left(x - \frac{v_0^2\sin(2\theta)}{2g}\right)^2 + \frac{v_0^2\sin^2(\theta)}{2g}$$

där v_0 är utkastfarten, y_0 är utkasthöjden, θ är utkastvinkeln och g är tyngdaccelerationen.

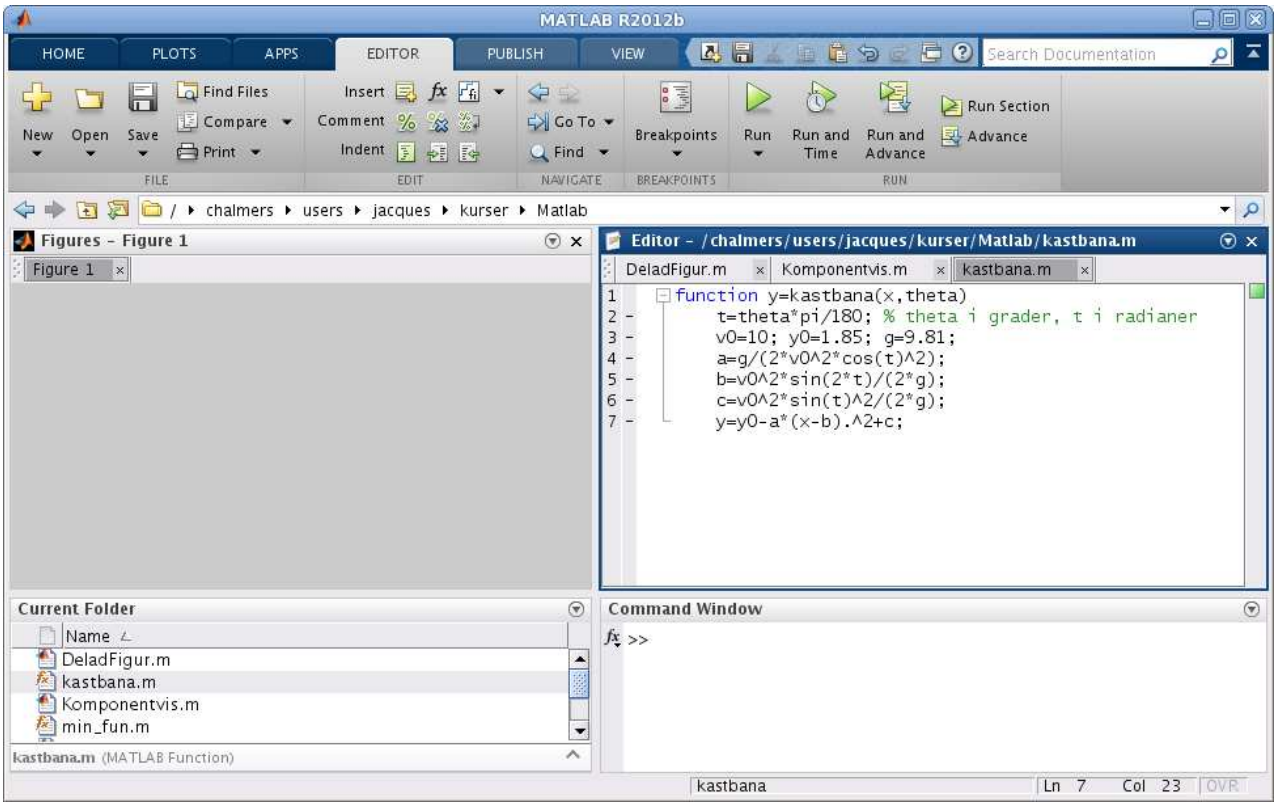
Vi gör en function med namnet `kastbana` som beskriver kastbanan för olika utkastvinklar.

```
function y=kastbana(x,theta)
    t=theta*pi/180;     % theta i grader, t i radianer
    v0=10; y0=1.85; g=9.81;
    a=g/(2*v0^2*cos(t)^2); b=v0^2*sin(2*t)/(2*g); c=v0^2*sin(t)^2/(2*g);
    y=y0-a*(x-b).^2+c;
```

Första raden inleds med `function`, för att tala om att det just är en funktion vi beskriver, och `kastbana` är namnet på funktionen.

Funktionens värde kommer ges till variabeln `y` (utdata) och funktionens argument (indata) är `x`, så klart, samt utkastvinkeln `theta` (lämpligt i vårt fall då vi skall rita flera grafer). Lägg märke till omvandlingen från grader till radianer.

Vi skriver in funktionen i editorn och `kastbana.m` ges som namn till textfilen.

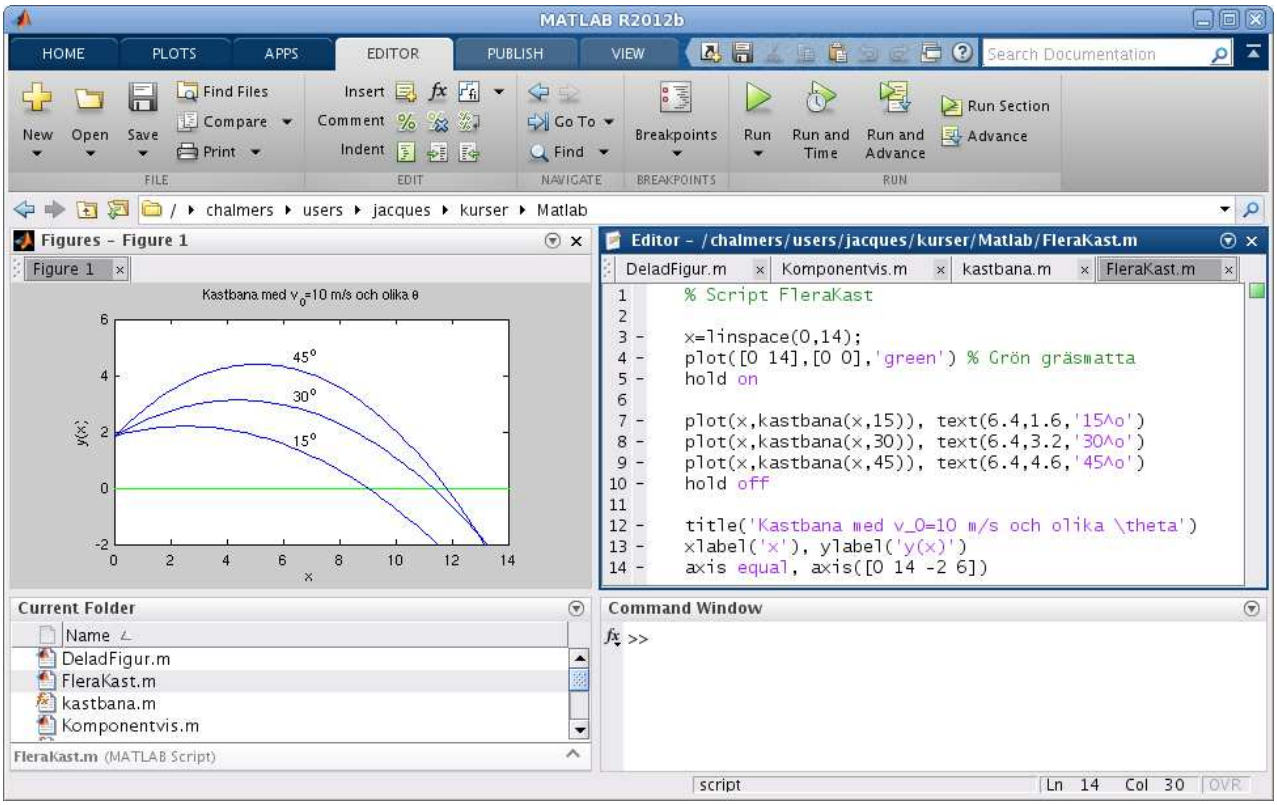


En funktion (function) är alltså en textfil med följande struktur

```
function ut = funktionsnamn(parametrar)
    satser
```

Här är `funktionsnamn` det namn vi ger funktionen och `funktionsnamn.m` är det namn vi ger textfilen där programkoden lagras. Med `parametrar` avser vi indata till funktionen, ofta en variabel ibland flera. Funktionen måste innehålla en sats där `ut`, som står för utdata eller funktionsvärdet, tilldelas ett värde.

Vi gör sedan ett script där vi tar $v_0 = 10$ m/s, $y_0 = 1.85$ m och ritar kastbanorna för några olika utkastvinklar. Så här ser det ut när vi ritat graferna. Vi har också placerat ut lite förklarande text vid graferna med kommandot `text`.

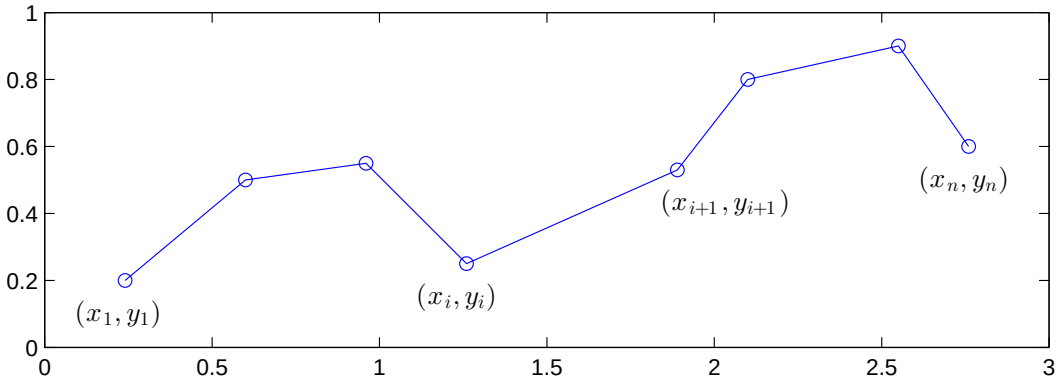


Uppgift 4. Skriv den function och det script för kastbanan som vi pratar om i exemplet. Rita graferna. Varför delar vi upp funktionsuttrycket för $y(x)$ i flera delar?

Avsnittet ”Egna funktioner” får man repetera flera gångar, gradvis kommer man vänja sig och börja förstå. Arbeta med det, ge det tid! Försök tänka efter vad ni gör, undvik ”Copy and Paste”, det finns inga magiska genvägar.

6 Kurvritning

Ett polygontåg är en följ av punkter $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, som vi successivt förbinder mer räta linjer.



Polygontåget kan ritas upp i MATLAB genom att man bildar vektorerna $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ och $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ och sedan ger kommandot `plot(x,y)`.

Grafritning är ett polygontåg vi ritar upp. Tag t.ex. grafen till $f(x) = \sin(x)$ för $0 \leq x \leq 2\pi$. Vi har då $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ med $0 = x_1 < x_2 < \cdots < x_n = 2\pi$ och $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ med $y_i = \sin(x_i)$. Sedan ritar vi upp med `plot(x,y)`.

Om polygontåget är slutet, dvs. $x_n = x_1$ och $y_n = y_1$, och om det inte korsar sig självt så omsluter det ett område i planet, ett s.k. polygonområde. Vi kan använda `fill` för att färglägga ett sådant område.

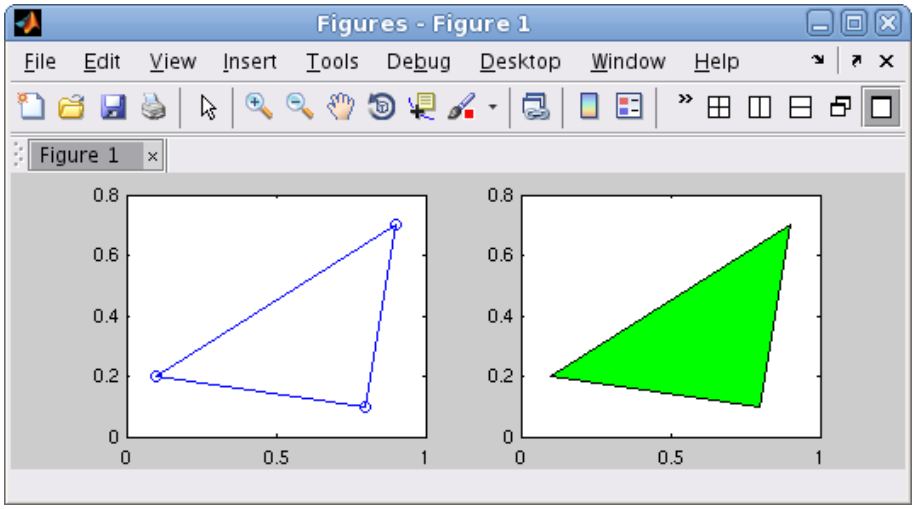
Vi ritar upp polygontåg som ges av punkterna $(0.1, 0.2), (0.8, 0.1), (0.9, 0.7), (0.1, 0.2)$, dvs. en triangel.

```
>> x=[0.1 0.8 0.9 0.1];
>> y=[0.2 0.1 0.7 0.2];
>> subplot(1,2,1)
>> plot(x,y,'-o'), axis([0 1 0 0.8])
```

Med `'-o'` anger vi att punkterna både skall förbindas med räta linjer och markeras med små ringar.

Vi fyller området med grön färg och vi använder `axis` för att få lite "luft" runt triangeln.

```
>> subplot(1,2,2)
>> fill(x,y,'g'), axis([0 1 0 0.8])
```



Uppgift 5. Rita en cirkel fylld med grön färg, rita sedan en kvadrat inskriven i cirkeln och fyll kvadraten med gul färg. Använd `hold on`.

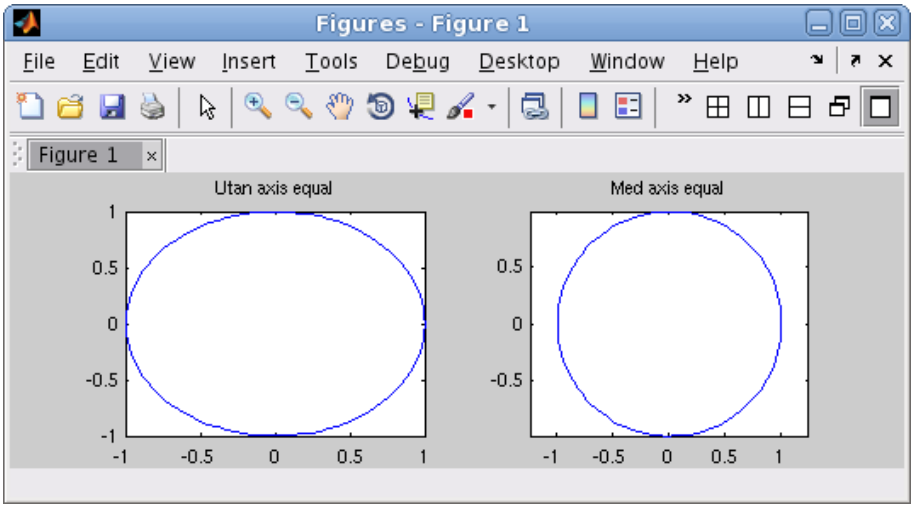
Nu skall vi rita s.k. parameterframställda kurvor. Som exempel tar vi enhetscirkeln

$$(x(t), y(t)) = (\cos(t), \sin(t)), \quad 0 \leq t \leq 2\pi$$

När man ritar sådana kurvor ritar man inte ut parametern t utan enbart x- och y-värdena.

```
>> t=linspace(0,2*pi);
>> x=cos(t); y=sin(t);
>> subplot(1,2,1)
>> plot(x,y)
>> title('Utan axis equal')
```

```
>> subplot(1,2,2)
>> plot(x,y)
>> axis equal    % annars blir cirkeln tillplattad
>> title('Med axis equal')
```



Uppgift 6. Rita kurvorna $(x(t), y(t)) = (\cos(t)+\cos(3t), \sin(2t))$ och $(x(t), y(t)) = (\cos(t)+\cos(4t), \sin(2t))$, för $0 \leq t \leq 2\pi$. Använd `subplot` och rita kurvorna i olika koordinatsystem.