

# Funktioner och grafitning i MATLAB

## 1 Inledning

I den här laborationen ser vi först 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 grafitning och hur vi definerar egna funktioner. Avslutningsvis ser vi lite på uppritning av allmänare kurvor i planet.

## 2 Något om matriser

Grundstenen i linjär algebra är matrisbegreppet. Matriser är även den grundläggande datatypen i MATLAB. Den linjära algebran behandlas till våren, 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 typen  $m \times 1$  kallas kolonnmatris (kolonnvektor) och en matris av typen  $1 \times n$  kallas radmatris (radvektor):

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

Element nr  $i$  ges i MATLAB av `b(i)`, `c(i)` och antalet element ges av `length(b)`, `length(c)`.

Som exempel tar vi

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

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
```

Så här beskriver vi kolonnvektorn respektive radvektorn

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

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

Man använder hakparanteser ([ ]) för att bygga upp matriser och vektorer. Semikolon (;) innanför hakparanteserna betyder radbyte.

**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.

Nu skall vi bilda (eller generera) en radvektor på ett par andra sätt. Vi ser åter på radvektorn **c** från exemplet. Eftersom det är samma avstånd mellan talen i elementen, kan den bildas enligt

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

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

```
variabel=start:steg:slut
```

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** enligt

```
variabel=linspace(start,slut,antal)
```

Här ger man det totala antalet värden man vill ha jämnt fördelade mellan start- och slutvärdet. Utelämnar man **antal** så får man 100 värden som standardvärde.

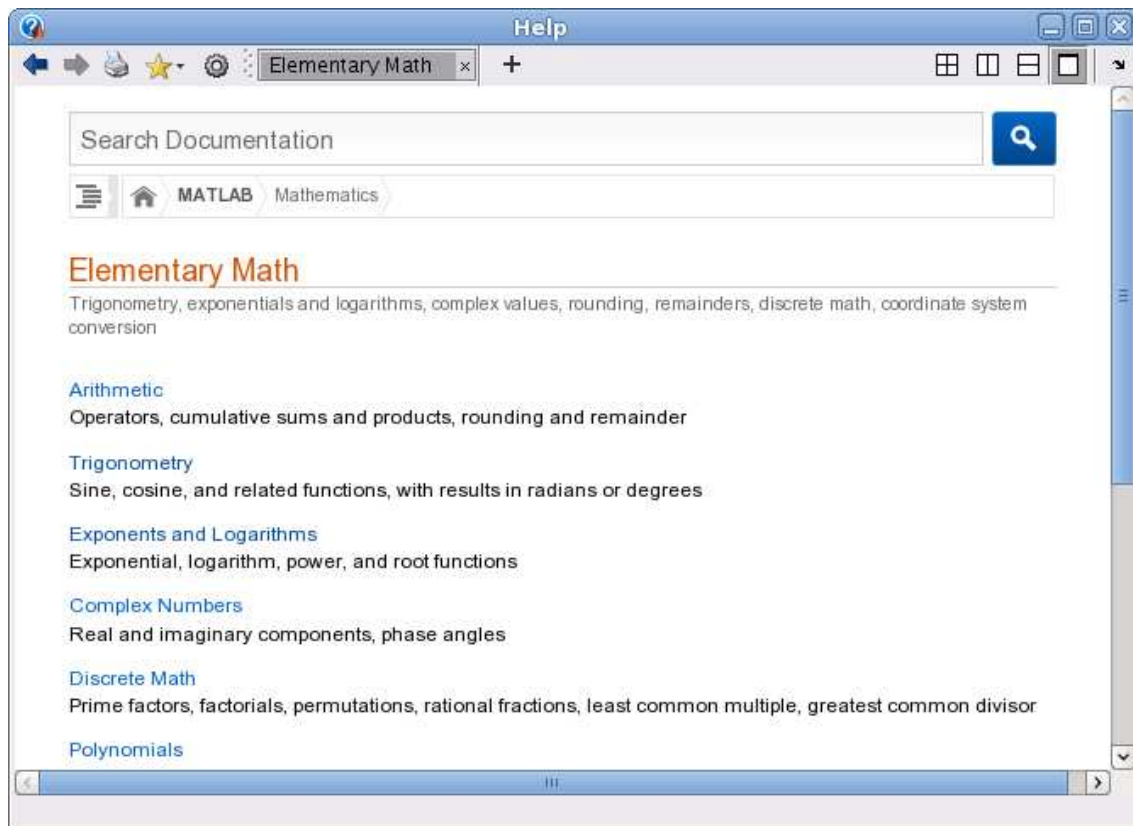
Att använda **linspace** är det effektivaste sättet vid grafitning då vi har ett intervall vi är intresserade av (start- och slutvärde) och vill bara ha tillräckligt många punkter (antal) i intervallet för att kunna rita en jämn och snygg graf. Exempelvis

```
>> 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 hjälptexterna 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 typ, 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  $\sin(\mathbf{x})$  en radvektor eftersom  $\mathbf{x}$  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. Varför ritar man grafen över intervallet  $-\frac{\pi}{2} + s \leq x \leq \frac{\pi}{2} - s$ , där  $s$  är ett litet positivt tal?

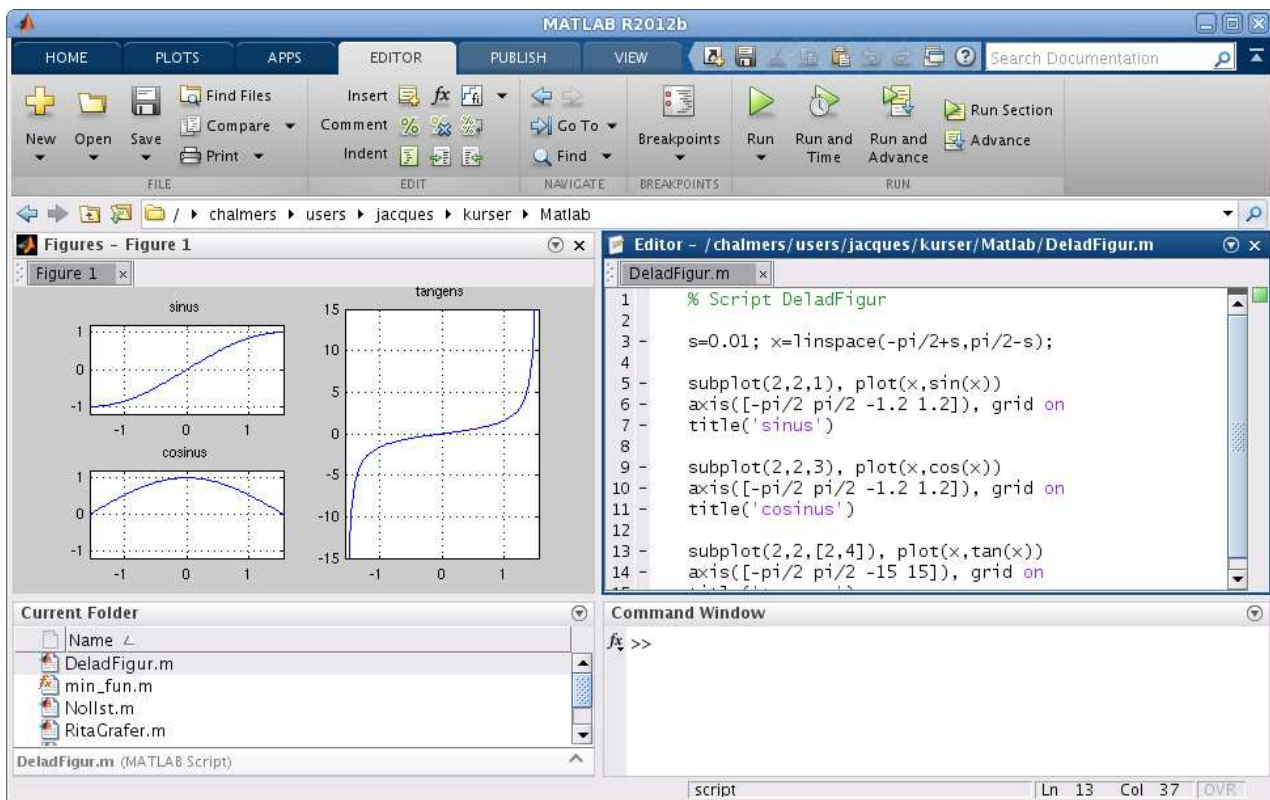
## 4 Funktionsgrafer

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.

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))
```

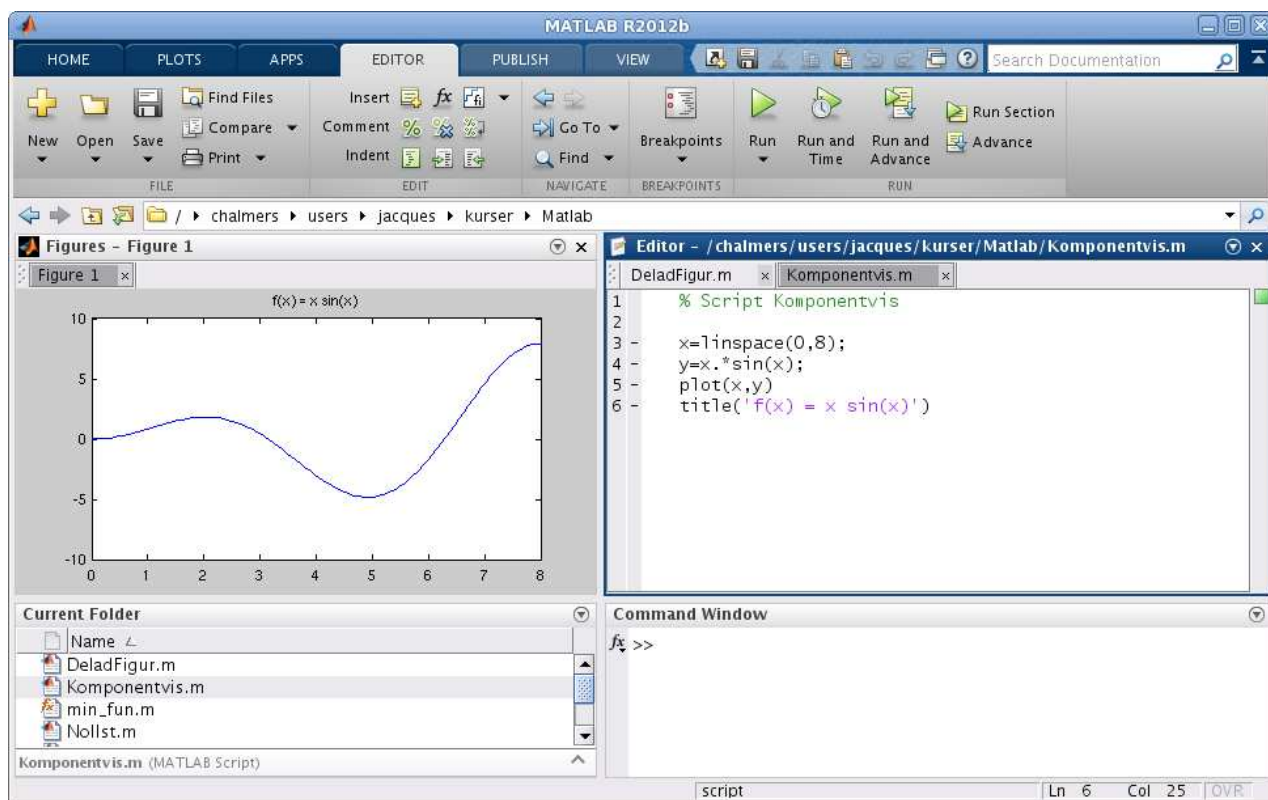
```
>> axis([-pi/2 pi/2 -1.2 1.2]), grid on, title('cosinus')
>> subplot(2,2,[2,4])          % samma indelning men gör 2:a och 4:e aktiva
>> plot(x,tan(x))
>> axis([-pi/2 pi/2 -15 15]), grid on, title('tangens')
```

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 av tangens.

Vi använder `axis` när vi inte vill ha 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  $-15 \leq y \leq 15$  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  $0 \leq x \leq 8$ .

Här ser vi grafen



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

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

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

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

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

## 5 Egna funktioner

Man har ofta behov av att definiera egna funktioner i sina program. I MATLAB kan man göra detta på olika sätt. Ett sätt är att beskriva funktionen med en anonym funktion (**anonymous function**).

**Exempel 3.** Låt oss söka nollställena till funktionen  $f(x) = 0.5(x-2)^2 - 2\cos(2x) - 1.5$ , dvs. vi vill bestämma de  $x$  som löser ekvationen

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

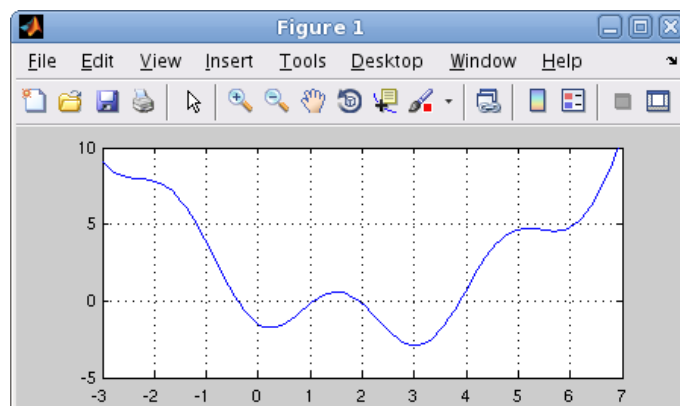
För att beskriva vår funktion i MATLAB skriver vi

```
>> f=@(x)0.5*(x-2).^2-2*cos(2*x)-1.5;
```

Detta kallas för en anonym funktion och `@(x)` är det som anger att `f` ska vara en funktion (och inte en vanlig variabel). Raden ovan är en definition, vi talar om hur funktionen ska beräknas men ingen beräkning av funktionen görs. Det är först när funktionen anropas som funktionsvärdena beräknas.

Vi ritar grafen till  $f(x)$  över intervallet  $-3 \leq x \leq 7$  för att få en uppfattning om hur många nollställena som finns och ungefär var de ligger.

```
>> x=linspace(-3,7);  
>> plot(x,f(x))  
>> axis([-3 7 -5 10]), grid on
```



Det finns ett kommando, `fzero`, i MATLAB som beräknar nollställena till en given funktion och som används enligt `x=fzero(fun,x0)`. Här beskriver `fun` funktionen vi skall finna nollstället till, `x0` är en första approximation av nollstället.

Vi använder `fzero` för att beräkna ett av nollställena till funktionen ovan enligt

```
>> x0=4;  
>> x=fzero(f,x0)  
x =  
    3.8664
```

Vi beskrev vår funktion genom att i MATLAB införa 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**). Vi ser på det i följande exempel.

**Exempel 4.** 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,  $\theta$  är utkastvinkeln och  $g$  är tyngdaccelerationen.

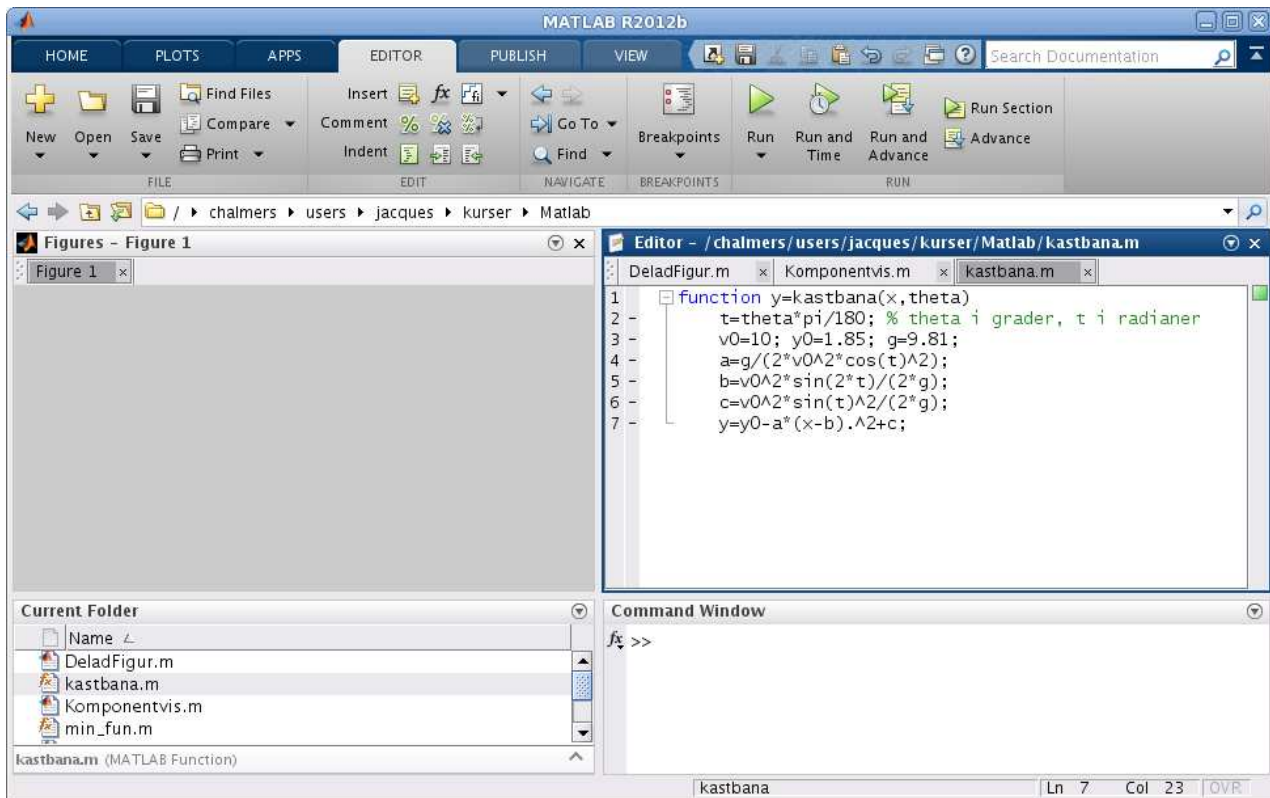
Vi gör en **function** med namnet `kastbana` som beskriver kastbanan för olika utkastvinklar.

```
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`, 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.

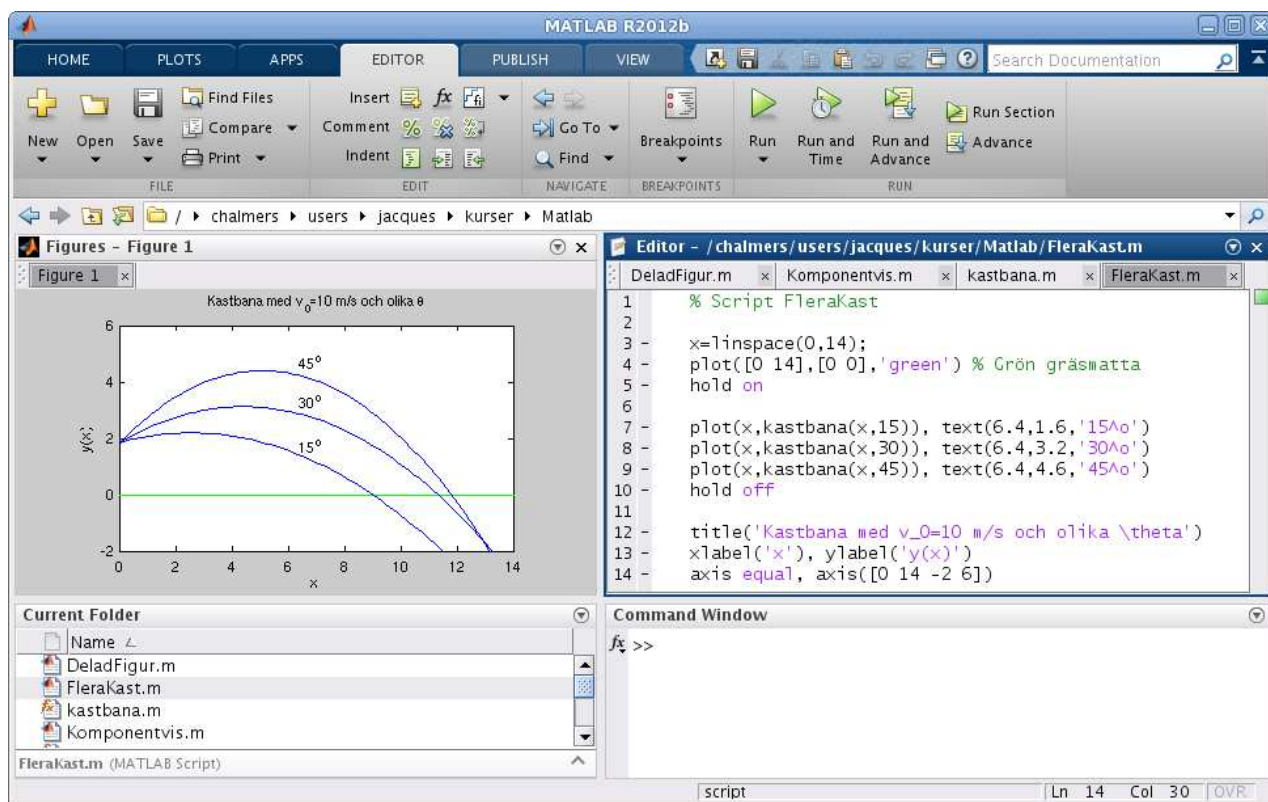


En funktion (function) är alltså en textfil med följande struktur

```
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`.

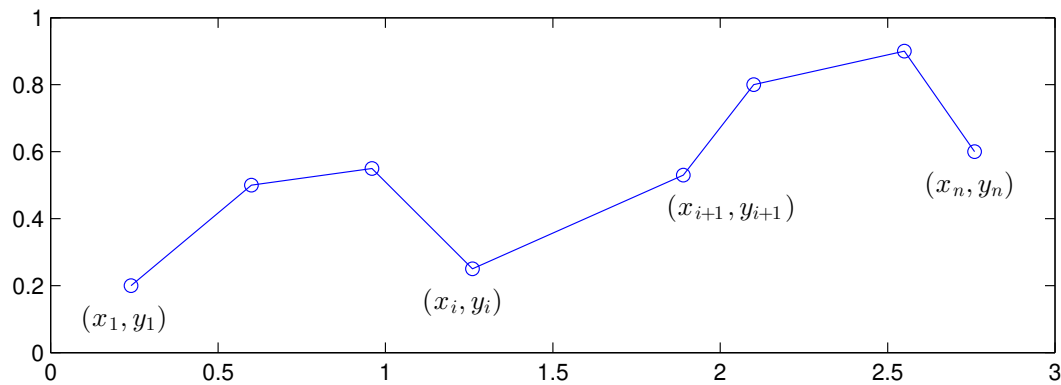


**Uppgift 4.** Skriv den function och det script för kastbanan som vi pratar om i exemplet. Rita graferna. Vad blir höjden vid  $x = 6$  och  $\theta = 30^\circ$ ? Rita ut med en liten ring eller stjärna. Läs i hjälptexten för `plot` hur du ritar t.ex. en ring. Varför delar vi upp funktionsuttrycket för  $y(x)$  i flera delar?

Avsnittet "Egna funktioner" får man repetera flera gånger, 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 följd av punkter  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , som vi successivt förbinder med rätta linjer.



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

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

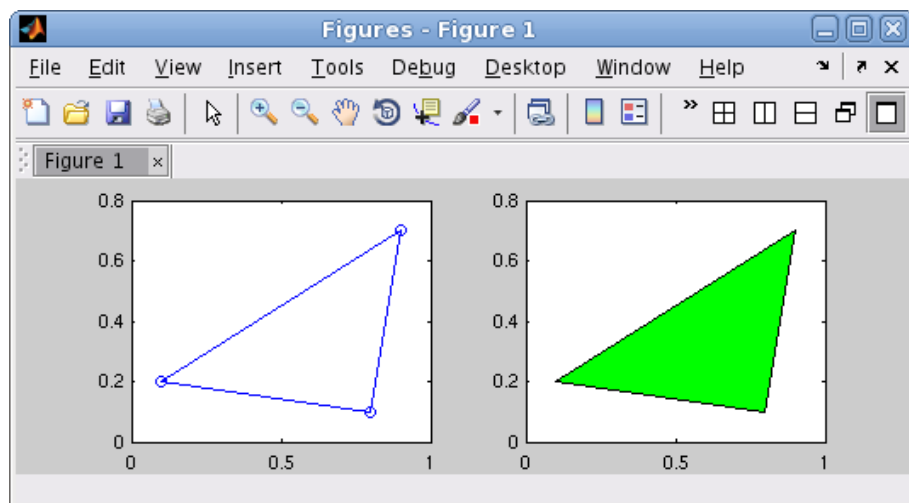
Vi ritas upp polygontåg som ges av punkterna  $(0.1, 0.2), (0.8, 0.1), (0.9, 0.7), (0.1, 0.2)$ , dvs. en triangel.

```
>> 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 rätta 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])
```



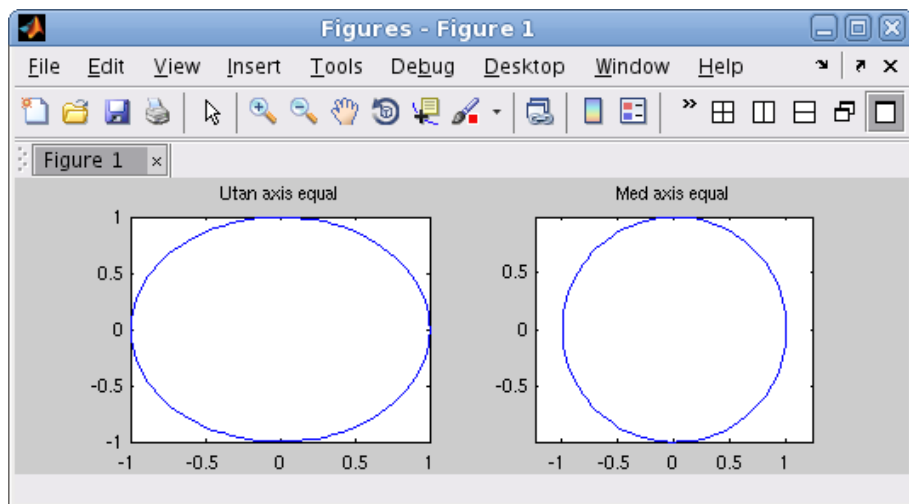
Nu skall vi rita s.k. parameterframställda kurvor. Som exempel tar vi enhetscirkeln

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

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

**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.