Kort orientering om MATLAB

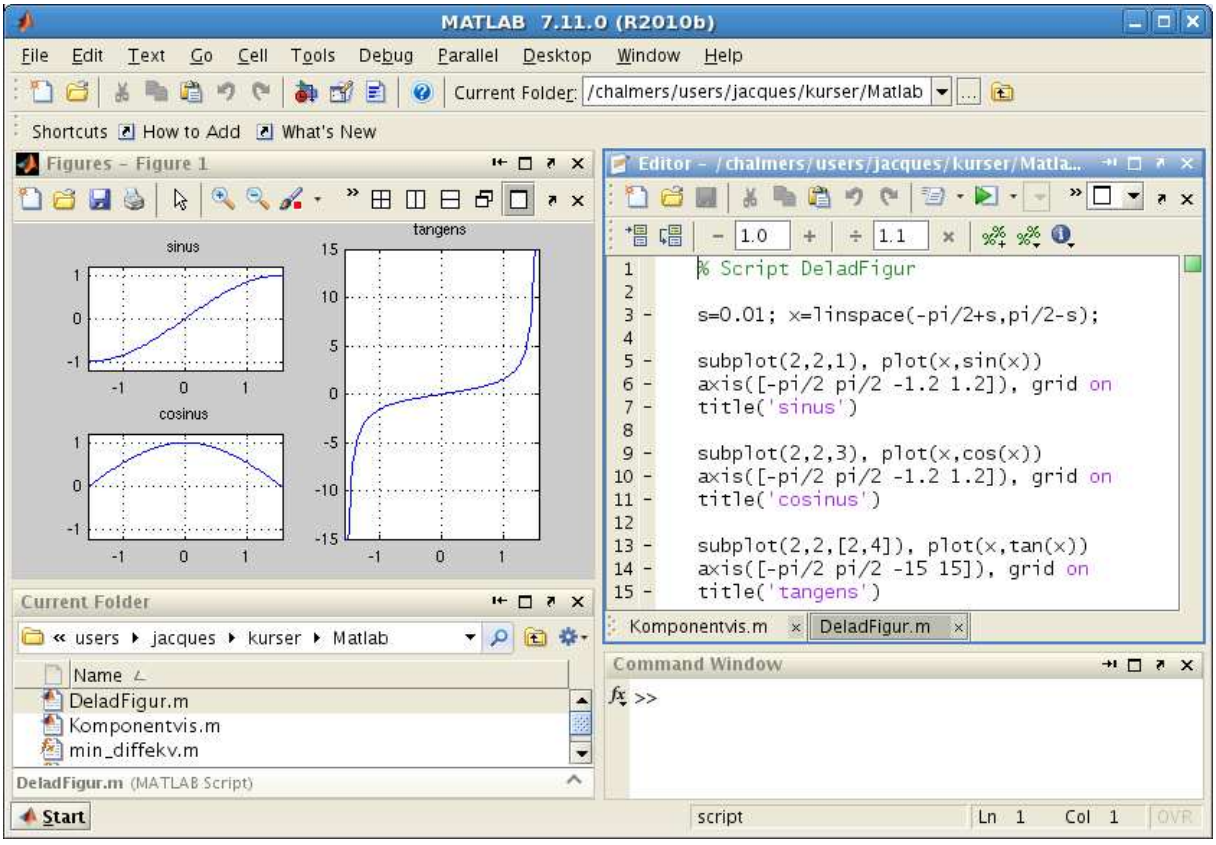
1 Inledning

MATLAB är både en interaktiv matematikmiljö och ett programspråk, som används på de flesta tekniska högskolor runt om i världen, och har stor användning även inom industrin.

En av styrkorna med MATLAB är att systemet är utbyggbart med bibliotek eller verktygslådor, toolboxes, för olika tillämpningsområden.

Ni kommer använda MATLAB i många kurser i utbildningen. Bl.a. kommer ni göra laborationsuppgifter i matematikkurserna.

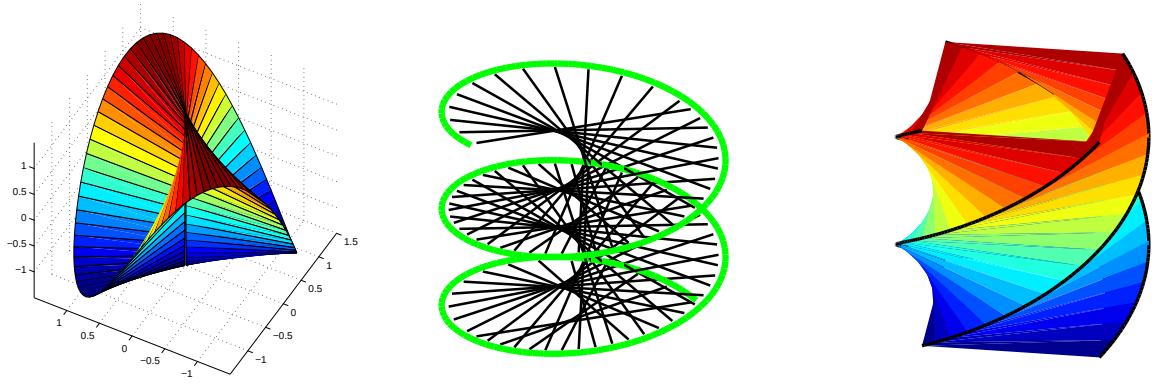
Så här kan det typiskt se ut i MATLAB när vi ritat graferna till några funktioner.



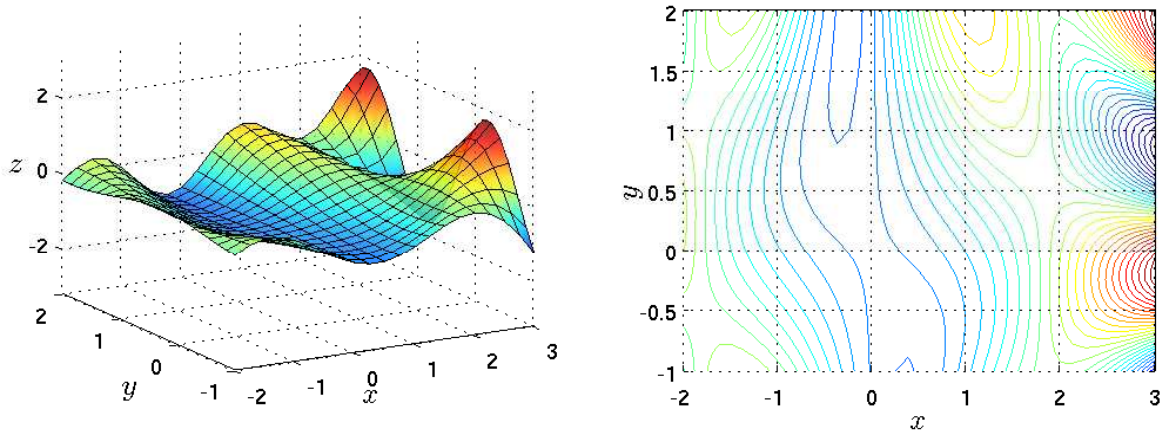
Nu i första läsperioden hinner ni bara bekanta er lite med MATLAB, mest för att man skall kunna ge en del exempel i kursen i inledande matematik.

Under andra läsperioden får ni möjlighet att lära er betydligt mycket mer MATLAB, och göra laborationsuppgifter i matematikkursen i envariabel analys.

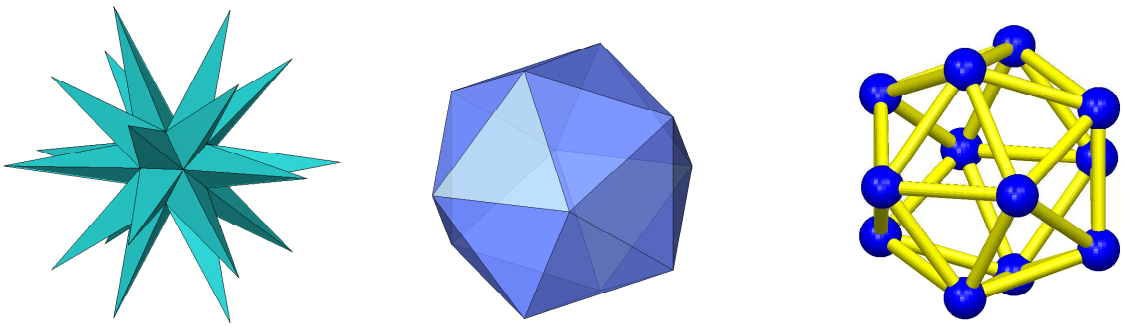
En liten matematisk lek med former och färger med dragning mot arkitektur, kommer vi också hinna med.



Till våren i läsperiod 3 kommer ni i matematikkursen i flervariabel analys använda MATLAB för att illustrera den matematik ni lär er. Här nedan ser vi en typisk bild av en funktion i två variabler, en s.k. funktionsyta till vänster med tillhörande s.k. nivåkurvor till höger.



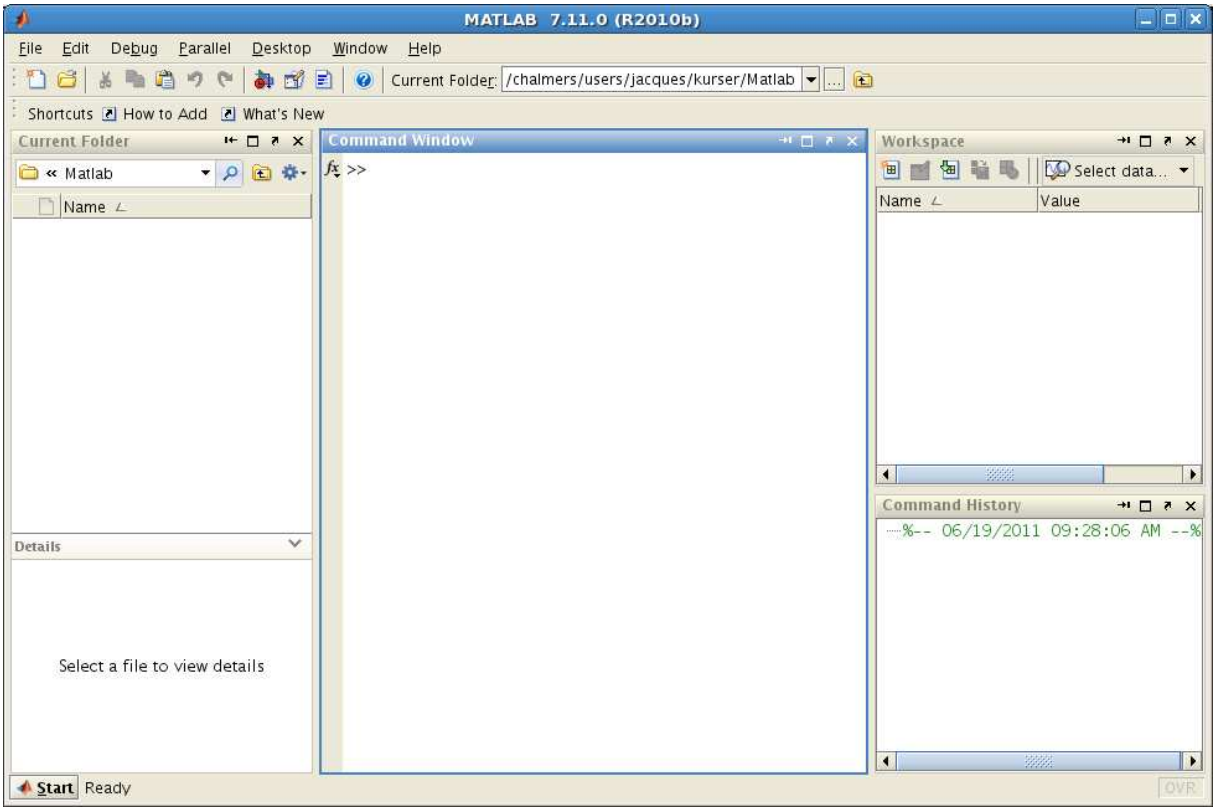
Vi kommer också arbeta med geometri, bl.a. med polygoner i rummet.



Från och med våren kommer även andra kurser än matematikkurserna ta vid och ge er fortsatt utveckling i MATLAB-kunnande. I själva verket kommer MATLAB följa er som ett arbetsredskap genom hela utbildningen.

2 Starta MATLAB

Vid en WINDOWS-dator startas MATLAB genom att man går in under Start-symbolen och väljer All Programs och därunder MATLAB.



Man avslutar MATLAB genom att gå in under File och välja Exit MATLAB (längst ned i menyn).

MATLAB-fönstret man får upp kallas Desktop och dess utseende eller konfiguration kallas Desktop Layout. Den standard Desktop Layout vi får då vi startar MATLAB för första gången ser vi ovan.

Vi kommer vid senare tillfälle göra en layout som ser ut ungefär som på bilden på första sidan och som är lämplig för det fortsatta arbetet med MATLAB, men just nu nöjer vi oss med standard layouten.

Den stora delen av Desktop som har Command Window som rubrik är den plats där vi ger kommandon (eller instruktioner) till MATLAB och det är där resultat av beräkningar skrivs ut.

3 En enkel beräkning och några grafer

Här följer några exempel så att vi snabbt kommer igång och ser lite resultat. Följ gärna med vid datorn (om möjligt) och knappa in efter hand i Command Window och se vad som händer.

Exempel 1. Beräkna volymen av ett klot med radien $r = 3$ cm. Volymen ges av $V = \frac{4}{3}\pi r^3$.

Först inför vi en variabel `r`, för radien, som vi ger värdet 3.

```
>> r=3
```

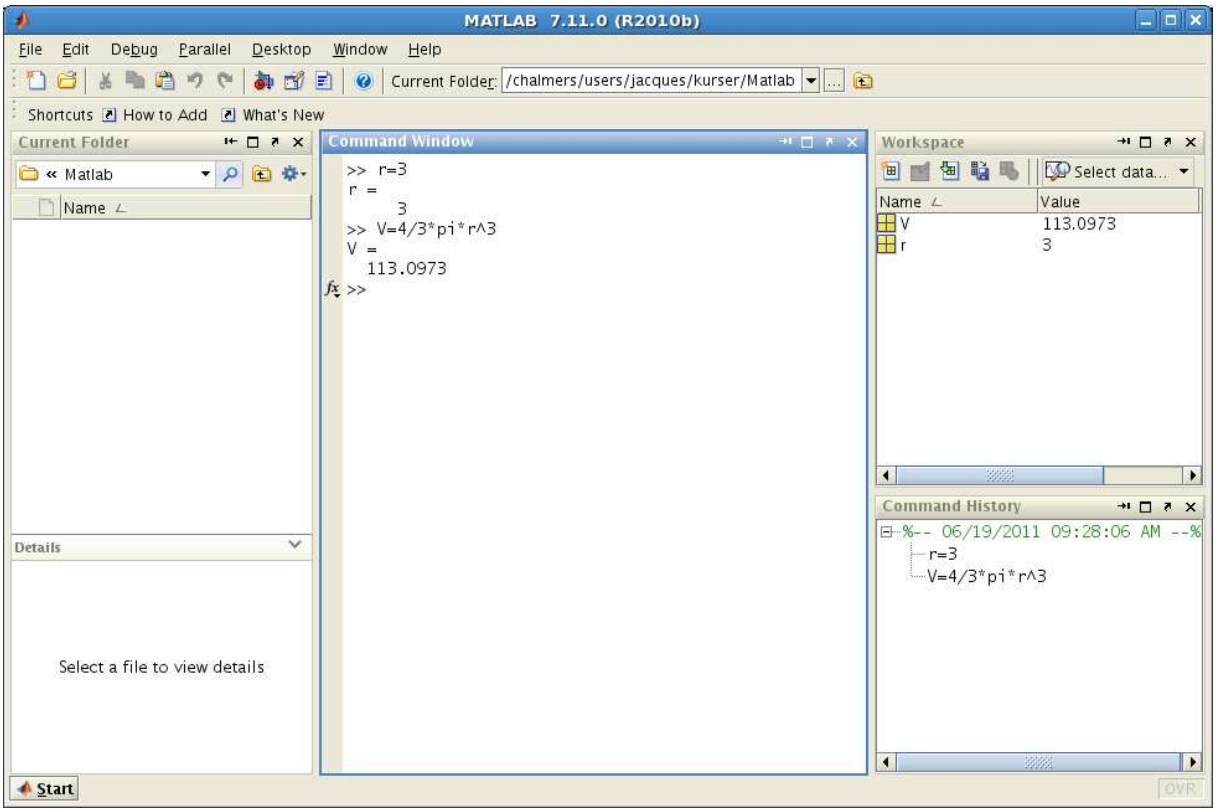
Ett variabelnamn skall börja med en bokstav (`a-z`, `A-Z`), därefter får vi ha bokstäver (`a-z`, `A-Z`), siffror (`0-9`) och understrykningstecken (`_`). Vidare skiljer MATLAB på stora och små bokstäver.

Den s.k. promptern (>>) skriver vi inte. Tecknet finns i Command Window på raden där vi skall skriva vårt kommando och visar att MATLAB är redo.

Därefter beräknar vi volymen enligt formeln och låter variabeln `V` få detta värde.

```
>> V=4/3*pi*r^3
```

Konstanten `pi` är en approximation av den matematiska konstanten π.



Övning 1. Beräkna arean av en cirkelskiva med radien $r = 4$ cm. Arean ges av $A = \pi r^2$.

Vi kan använda uppåtpil ($\uparrow$) för att komma till ett kommando vi givit tidigare. Om vi vill kan vi gå längs raden med vänster- och högerpilarna ($\leftarrow$), ($\rightarrow$) och redigera kommandot. När kommandot ser ut som vi vill trycker vi på enter ($\hookleftarrow$). Vi kan rensa Command Window med kommandot `clc`.

Exempel 2. Rita grafen av $f(x) = \sin(x) + 0.3\sin(4x)$ för $0 \leq x \leq 4\pi$.

Först gör vi en lista eller radvektor `x` av x-värden mellan 0 och 4π, med

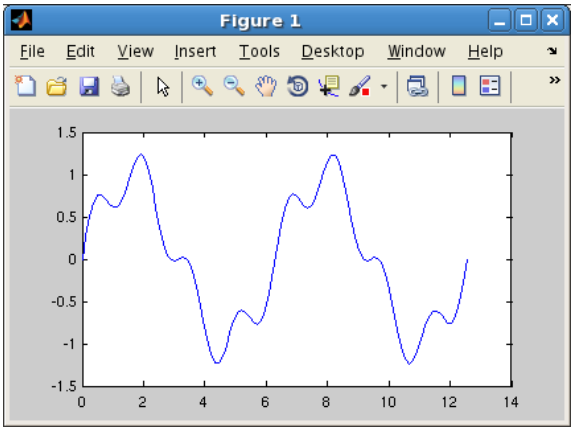
```
>> x=0:0.1:4*pi;
```

Närmare bestämt får vi värdena 0, 0.1, 0.2, 0.3, $\cdots$, 12.5, dvs. värden med start i 0, steget 0.1 och slut så nära upp mot 4π som möjligt.

Därefter gör vi en lista eller radvektor `f` med $f(x)$-värden för varje x-värde i `x` och ritar upp grafen med `plot`.

```
>> f=sin(x)+0.3*sin(4*x);
>> plot(x,f)
```

Dessa kommandon skriver vi i Command Window och ett Figure-fönster kommer upp



Om vi hade inte skrivit ett semikolon (;) sist i uttrycket för `x` och `f`, hade alla värden skrivits ut i Command Window och det vill vi nog inte. Vill vi rensa Figure-fönstret ger vi kommandot `clf`.

Det skulle fungera lika bra att räkna ut funktionsvärdena på plats i `plot` kommandot enligt

```
>> x=0:0.1:4*pi;
>> plot(x,sin(x)+0.3*sin(4*x))
```

Övning 2. Rita grafen till $f(x) = \sin(x) + 0.3\sin(5x)$ över samma intervall.

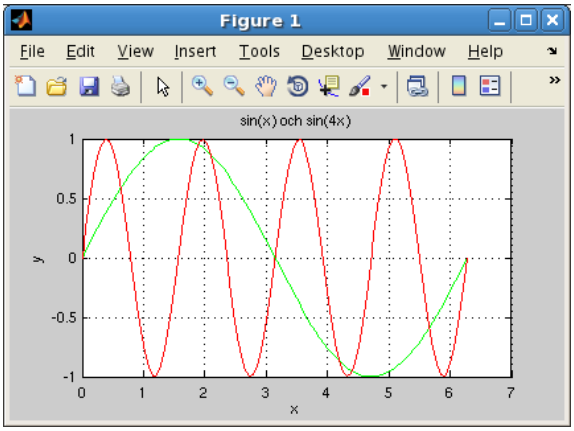
Exempel 3. Rita graferna av $f(x) = \sin(x)$ och $g(x) = \sin(4x)$ för $0 \leq x \leq 2\pi$. Sätt rubrik och text på axlarna.

Vi använder funktionen `linspace` för att få 100 punkter jämnt fördelade mellan 0 och 2π, då blir graferna jämna och snygga.

```
>> x=linspace(0,2*pi);
>> f=sin(x);
>> g=sin(4*x);
```

Vi ritar båda graferna samtidigt med `plot`, både paret `x`, `f` och paret `x`, `g`.

```
>> plot(x,f,'green',x,g,'red')
```



För att skilja graferna åt gjorde vi $\sin(x)$-grafen grön `'green'` och $\sin(4x)$-grafen röd `'red'`.

Vi sätter text på axlarna och rubrik samt lägger på ett rutnät med

```
>> xlabel('x')
>> ylabel('y')
>> title('sin(x) och sin(4x)')
>> grid on
```

Texterna inom apostrofer (' ') är s.k. textsträngar. Exempelvis är `'green'`, `'x'` och `'sin(x) och sin(4x)'` textsträngar.

4 Något om matriser

Grundstenen i linjär algebra är matrisbegreppet. Detta är även den grundläggande datatypen i MATLAB.

En matris är ett rektangulärt talschema

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

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

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

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 skriver in detta i MATLAB enligt

```
>> A=[1 4 7 10; 2 5 8 11; 3 6 9 12]
>> b=[1; 3; 5]
>> c=[0 2 4 6 8]
```

Övning 3. Skriv in matriserna i MATLAB och skriv sedan ut matriselementen a_{23}, b_2, c_3. Prova `size` och `length`. Ändra a_{23} genom att skriva `A(2,3)=15`.

Vi har redan i exempel 2 och 3 haft nytta av radvektorer eller radmatriser. Nu skall vi se exempel där ni har nytta av matriser.

5 Linjärt ekvationssystem

Linjära ekvationssystem kan vi lösa med MATLAB om vi först skriver dem på matrisform. Vi tar som exempel: Ekvationssystemet

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 14 \\ 3x_1 + 2x_2 + x_3 = 10 \\ 7x_1 + 8x_2 = 23 \end{cases}$$

skrivs på matrisform

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 14 \\ 10 \\ 23 \end{bmatrix}$$

dvs.

$$\mathbf{Ax} = \mathbf{b}, \quad \text{med } \mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 14 \\ 10 \\ 23 \end{bmatrix}$$

Med backslash-kommandot (\) alternativt kommandot `rref` (row-reduced-echelon form) löser vi systemet, $\mathbf{Ax} = \mathbf{b}$

```
>> x=A\b
>> rref([A b])
```

I det första fallet fungerar det bra om lösningen är entydig men sämre om det finns fria variabler eller inga lösningar alls. I det andra fallet reducerar MATLAB den utökade matrisen [**A b**] till reducerad trappstegsform.

Övning 4. Skriv följande ekvationssystem på matrisform och lös dem sedan med \ respektive `rref`. Beskriv också hur högerledet beror av kolonnerna i koefficientmatrisen.

$$\begin{cases} x_1 + 5x_2 + 9x_3 = 29 \\ 2x_1 + 5x_3 = 26 \\ 3x_1 + 7x_2 + 11x_3 = 39 \end{cases} \qquad \begin{cases} x_1 + x_2 + 3x_3 + 4x_4 = 2 \\ -2x_1 + 2x_2 + 2x_3 = -4 \\ x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - x_2 - 2x_3 - x_4 = 1 \end{cases}$$

6 Script

För att slippa skriva om sina kommandon, eller bläddra med uppåt- och nedåtpilar (↑), (↓) i kommandofönstrets historik, så brukar man oftast skriva sin kod i en script eller *skriptfil*.

En script är en textfil som innehåller det man skulle kunna skriva direkt vid promptern (>>) i Command Window, och som utförs i MATLAB genom att man ger textfilens namn som kommando. För att MATLAB skall hitta filen, förutsätter det att katalogen där filen ligger är aktuell katalog.

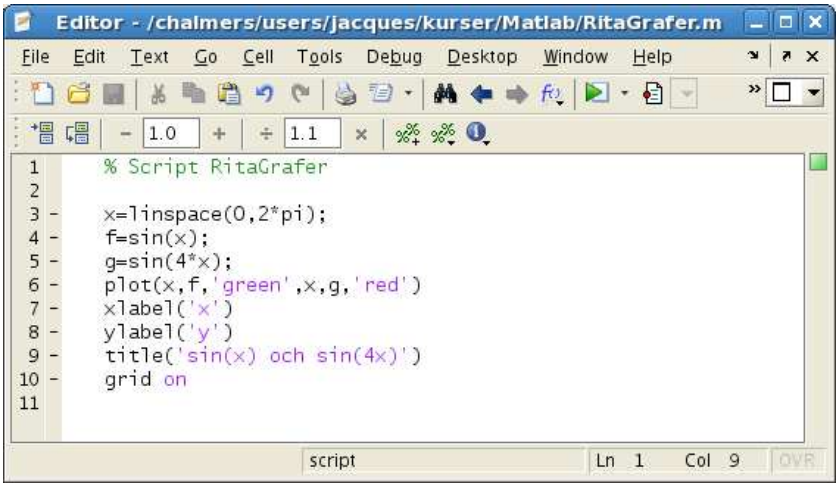
Man kan byta katalog med kommandot `cd` i Command Window, klicka sig fram i Current Folder eller använda Browse for folder i verktygsfältet i Desktop.

Utanför MATLAB får namnet på en script tillägget `.m` för att skilja den från andra filer.

MATLAB har en inbyggd editor som är det bästa verktyget att göra en script med. Om man inte redan har editorn uppe i Desktop så startas den genom att gå till File, sedan New och välja

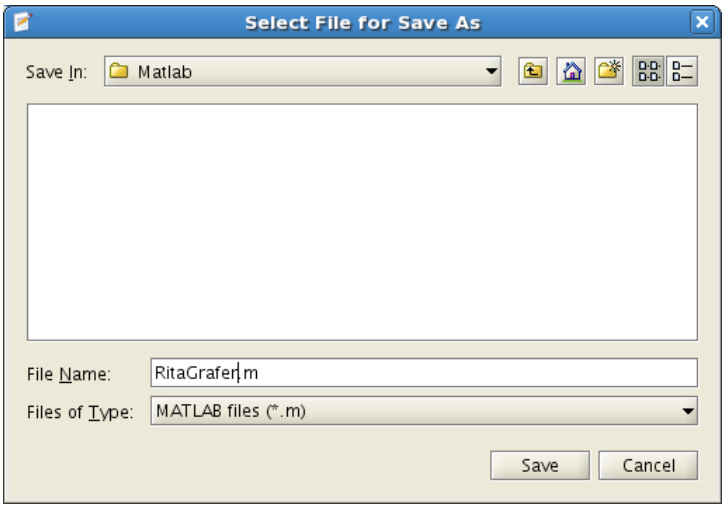
Script. Editorn markerar koden med olika färger för att visa vad som är kommentarer, nyckelord, textsträngar, etc. (Kommentarer inleds med procenttecken.)

Vi gör en script med namnet `RitaGrafer.m` som ritar graferna från förra exemplet.

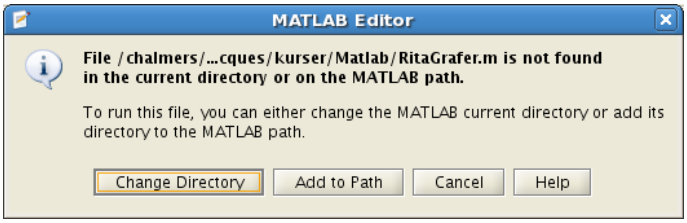


Spara kan vi göra under File och köra under Debug. Enklast är dock att trycka på  som finns i verktygsfältet. Då sparas vår script och utförs som om vi gav den som ett kommando.

Vi får givetvis upp samma grafer som tidigare. Så här ser dialogrutan ut som kommer upp då vi skall namnge skriptfilen.



Om man försöker köra en skritfil som ligger i en annan katalog än den aktuella, så får man upp en fråga om att byta till den katalogen:



Välj Change Directory så byter MATLAB katalog.

7 Lite programmering

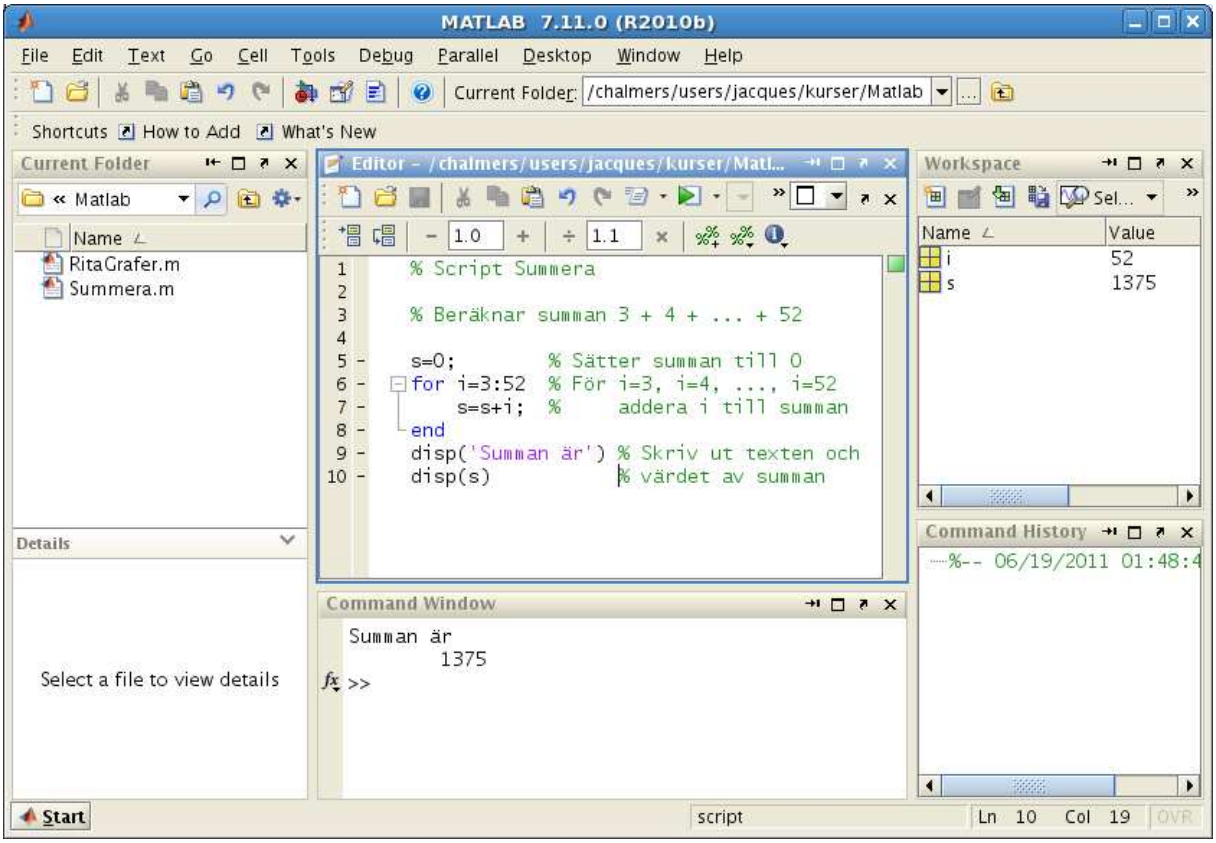
I MATLAB finns repetitions- och villkorssatser som påminner om motsvarande i programspråk som C och Java.

Vi nöjer oss för tillfället med att se på en repetitionssats, en `for`-sats, som vi använder för att beräkna en summa i följande exempel.

Exempel 4. Beräkna summan $s = 3 + 4 + 5 + \cdots + 52$

Vi gör en script med programkoden

```
s=0;
for i=3:52
    s=s+i;
end
```



Vi skriver lämpliga kommentarer (grön text) i programkoden och gör lämplig utskrift, först textsträngen `Summan är` och sedan summans värde.

I matematik skriver man gärna summan $3 + 4 + 5 + \cdots + 52$ med beteckningen

$$\sum_{i=3}^{52} i$$

Övning 5. Skriv en script som beräknar summan

$$\sum_{i=1}^{5} i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

8 Function

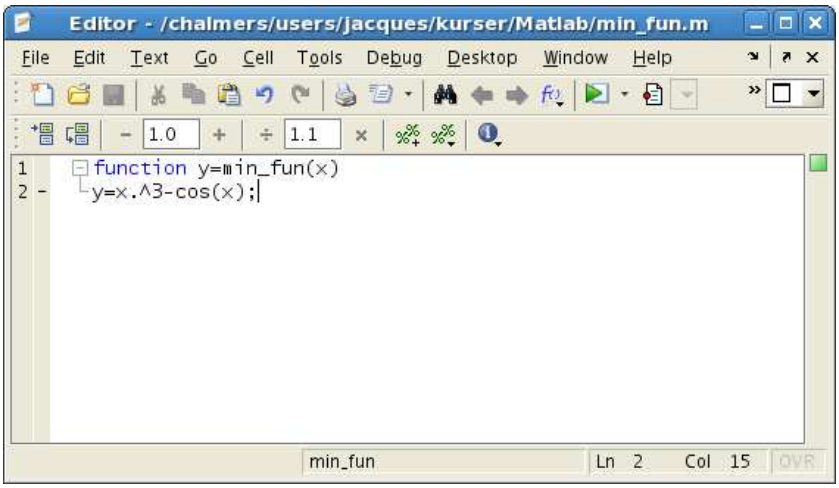
En function eller en *funktionsfil* är en textfil med samma namn som funktionen och som inleds med en funktionsdeklaration. Just nu ser vi bara på ett exempel där vi behöver göra en funktion.

Exempel 5. Vi vill hitta ett nollställe till funktionen $f(x) = x^3 - \cos(x)$.

Det finns en funktion `fzero` i MATLAB som hittar nollställen. För att använda `fzero` måste vi beskriva vår funktion och det gör vi som en function enligt

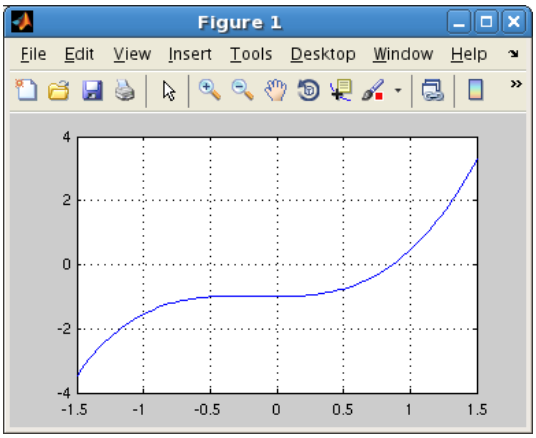
```
function y=min_fun(x)
y=x.^3-cos(x);
```

där `y` är funktionens värde (utdata), `x` är funktionens argument (indata) och `min_fun` är funktionens namn. Vi skriver in i editorn



Vi ritar grafen direkt i Command Window enligt

```
>> x=linspace(-1.5,1.5);
>> plot(x,min_fun(x)), grid on
```



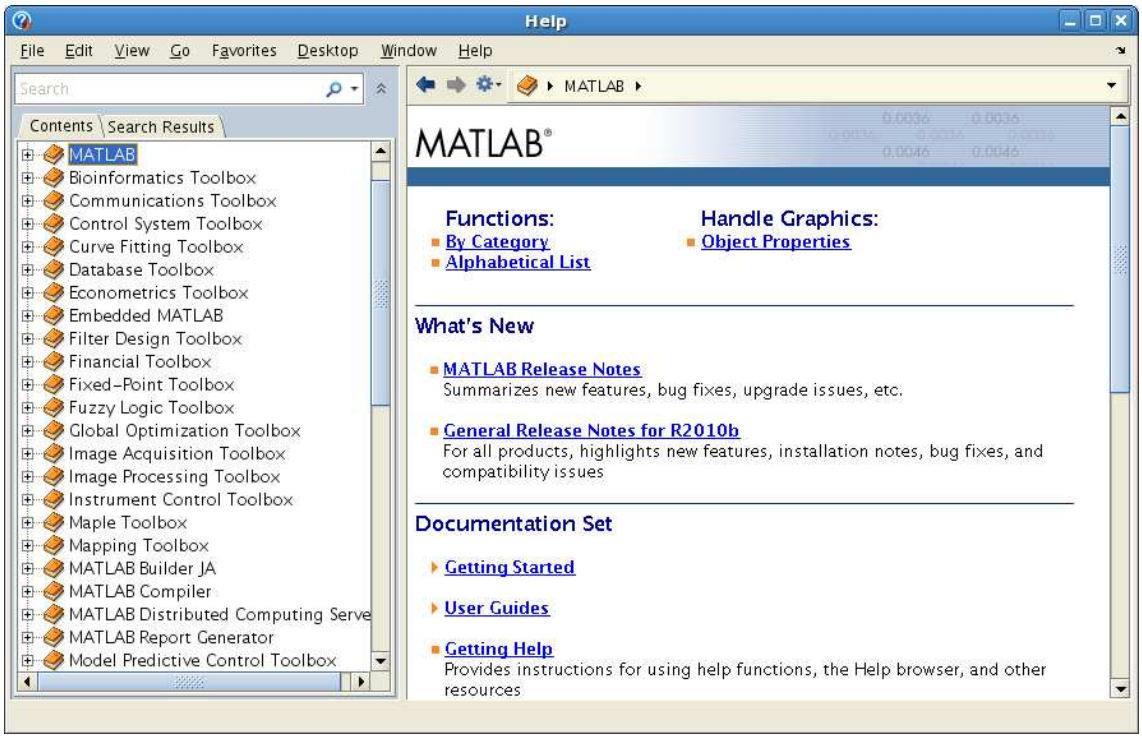
Vi ser att vi har ett nollställe nära $x = 1$ och låter `fzero` söka nollstället genom

```
>> x=fzero(@min_fun,1)
x =
    0.8655
```

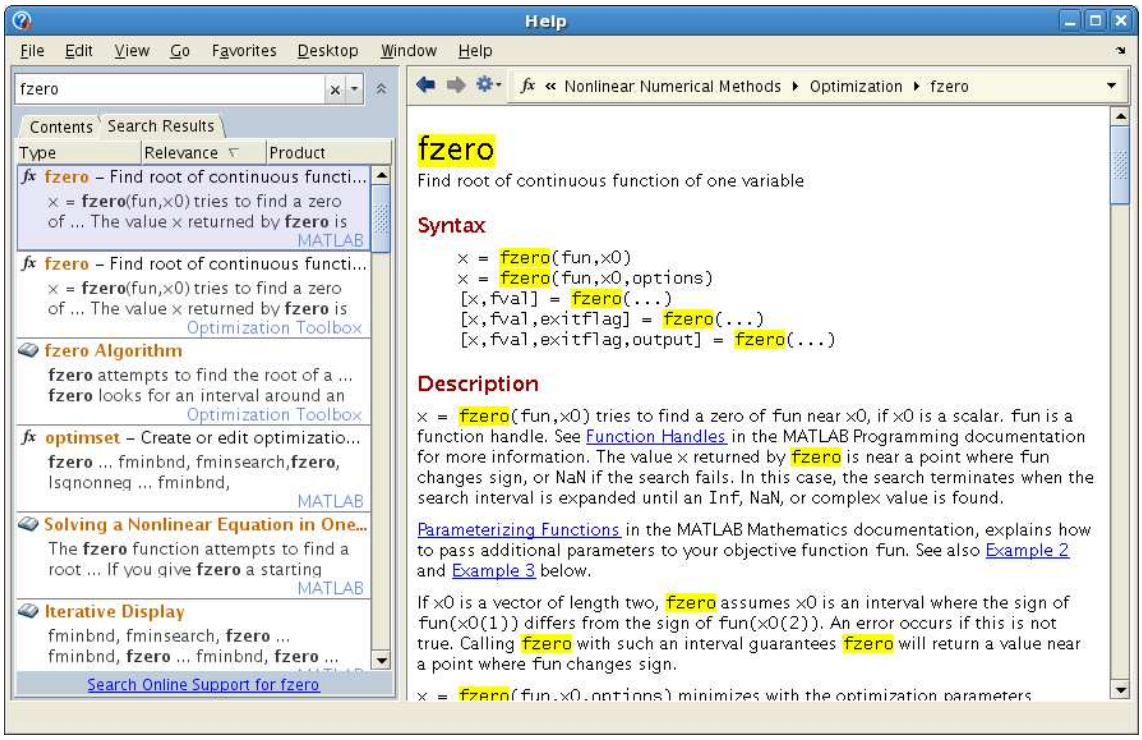
Med `@min_fun` talar vi om för `fzero` vilken funktion den skall arbeta med.

9 Helpdesk i MATLAB

Tryck på  i verktygsfältet, eller på MATLAB Help under Window, och Help Navigator öppnas.



Vi ser den stora uppsättningen av verktygslådor, för olika tillämpningsområden, som följer med. Här ser vi hjäptexten för funktionen `fzero`. Vi har skrivit `fzero` i sökrutan och tryckt på enter.



Läs gärna lite i texten och titta tillbaka på exempel 5 där vi använde `fzero`.