

# Introduktion till 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.

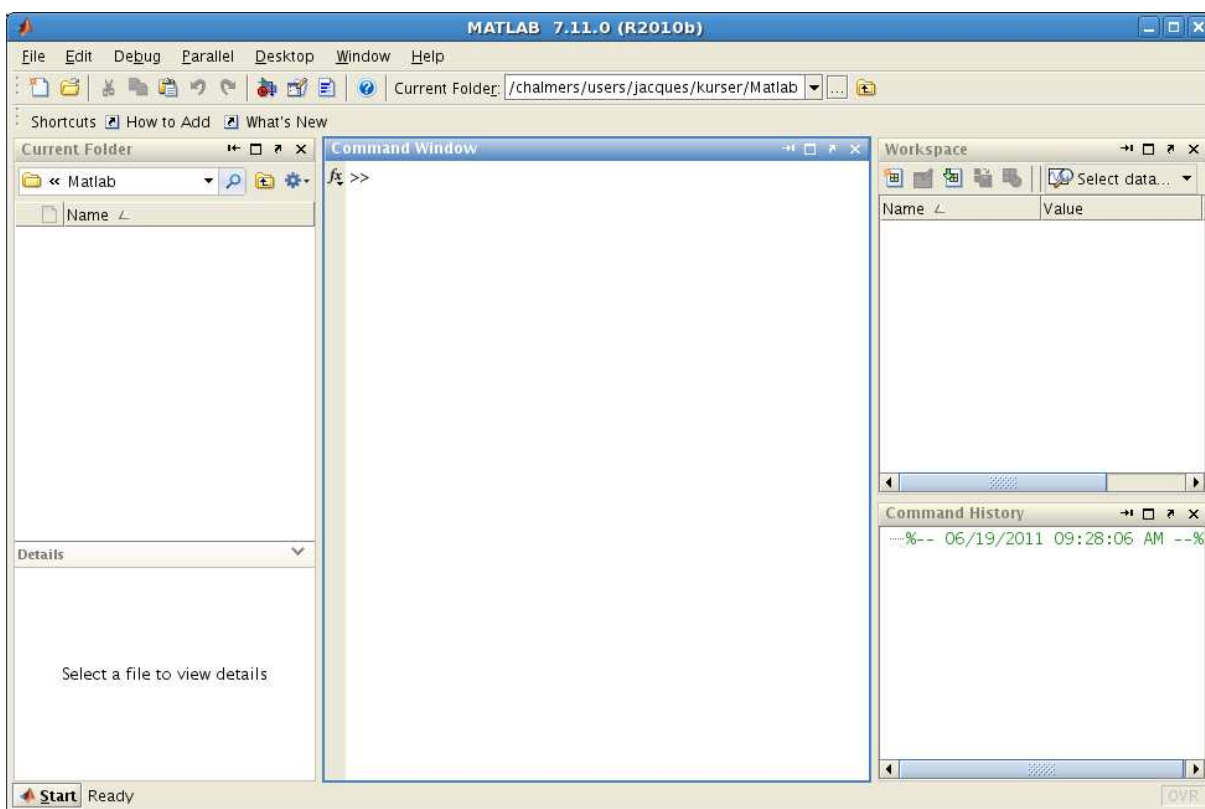
En av styrkorna med MATLAB är att systemet är utbyggbart med bibliotek eller verktyglådor, toolboxes, för olika tillämpningsområden.

Ni kommer använda MATLAB i många kurser i utbildningen. Det är viktigt att komma igång tidigt så att man blir en tillräckligt erfaren användare.

## 2 Starta MATLAB

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

Vid en LINUX-dator går man in under Applications och väljer Chalmers Applications och sedan Matlab.



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

MATLAB-fönstret som kommer upp kallas **Desktop** och dess konfiguration eller uppdelning kallas **Desktop Layout**. Vi kommer lite senare se hur man kan ändra den standard layout vi får då vi startar MATLAB första gången, genom att "klicka och dra", så att vi får en layout som vi trivs med. Man kan sedan spara sin layout och ge den ett namn. Om vi vill kan ha många olika layouter och vi kan alltid återvända till standard layouten.

### 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 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.

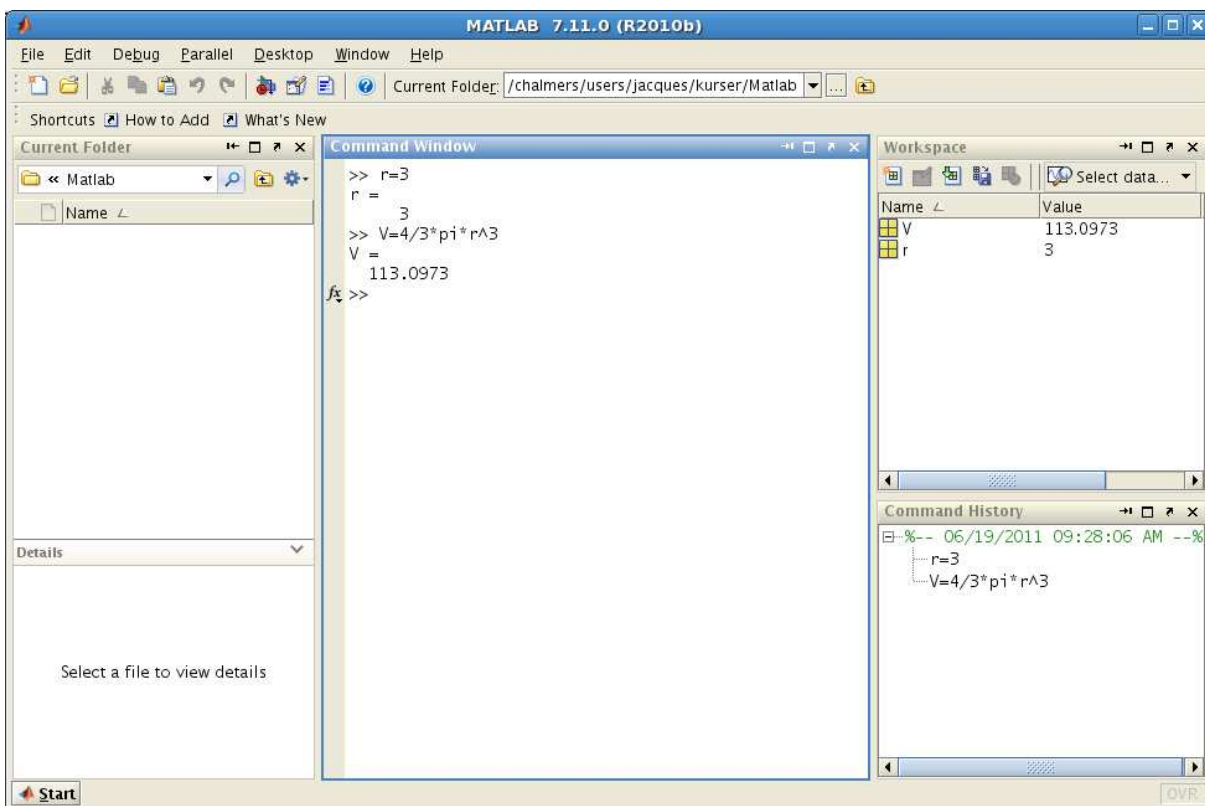
```
>> r=3
```

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 (\_). MATLAB skiljer 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 (konstanten  $\pi$  ger en approximation av  $\pi$ ) och låter variabeln  $V$  få detta värde.

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



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

**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  $\mathbf{x}$  av  $x$ -värden mellan 0 och  $4\pi$ , med kommandot

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

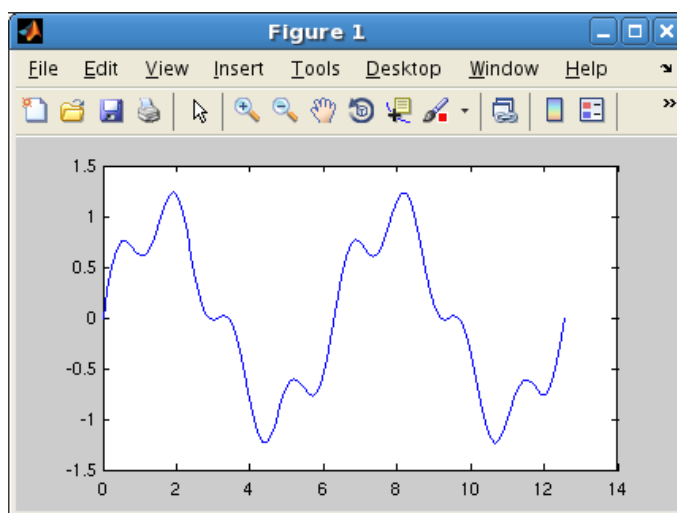
som vi skriver i **Command Window**.

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

Därefter gör vi en lista eller radvektor  $\mathbf{f}$  med  $f(x)$ -värden för varje  $x$ -värde i  $\mathbf{x}$  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  $\mathbf{x}$  och  $\mathbf{f}$ , hade alla värden skrivits ut i **Command Window** och det vill vi nog inte.

**Uppgift 2.** Rita grafen till  $f(x) = \sin(x) + 0.3 \sin(5x)$  över intervallet  $0 \leq x \leq 4\pi$ .

Vi kan använda uppåtpil ( $\uparrow$ ) för att komma till ett kommando vi givit tidigare, eller dra kommandot från **Command History**. 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 ( $\leftrightarrow$ ).

Vill vi rensa **Command Window** så ger vi kommandot `clc` och vill vi rensa **Figure 1** ger vi kommandot `clf`.

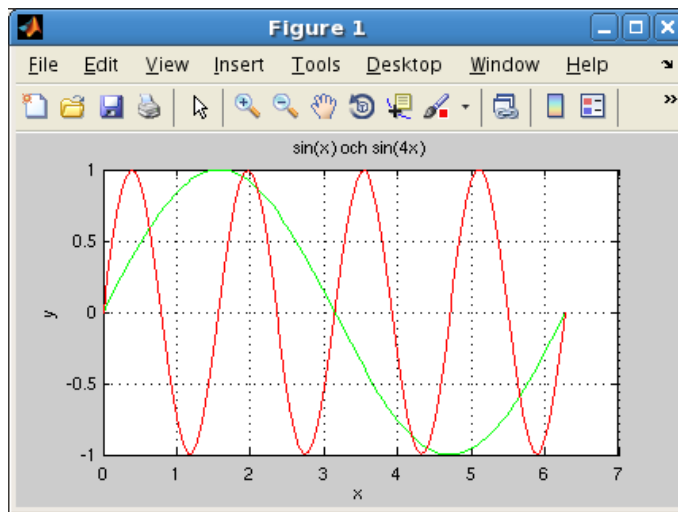
**Exempel 3.** Rita graferna av  $f(x) = \sin(x)$  och  $g(x) = \sin(4x)$  för  $0 \leq x \leq 2\pi$ . Sätt rubrik och text på axlarna.

Vi använder funktionen `linspace` för att få 100 punkter jämnt fördelade mellan 0 och  $2\pi$ , 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  $\sin(x)$ -grafens gröna `'green'` och  $\sin(4x)$ -grafens röda `'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
```

Vill vi ta bort rutnätet, gör vi det med `grid off`.

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

## 4 Script

För att slippa skriva om sina kommandon, eller bläddra med uppåt- och nedåtpilar ( $\uparrow$ ), ( $\downarrow$ ) i kommandofönstrets historik eller dra från **Command History**, så brukar man oftast skriva sin MATLAB-kod i en **script** eller *skriptfil* som vi ofta kommer säga.

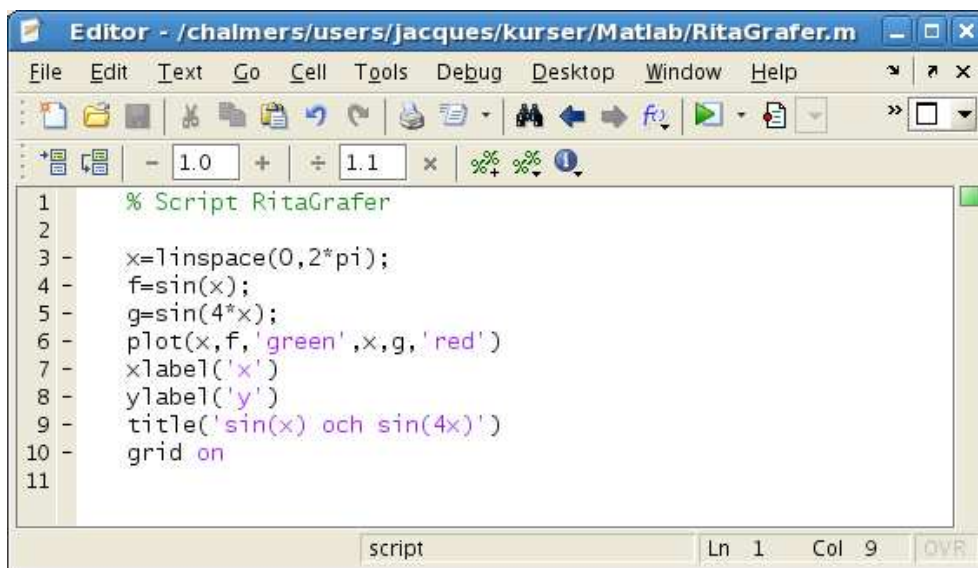
En **script** eller skriptfil är en textfil som innehåller det man skulle kunna skriva direkt vid promptern (`>>`) i **Command Window**, och som utförs i MATLAB genom att man ger textfilens namn som kommando. För att MATLAB skall hitta filen, förutsätter det att katalogen där filen ligger är aktuell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

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


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** eller skriptfil 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.)

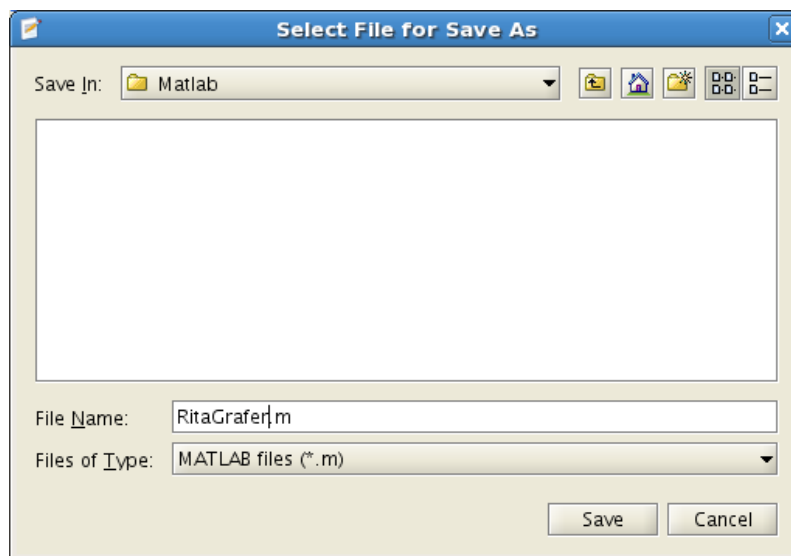
Vi gör en script som ritar graferna från exempel 3.



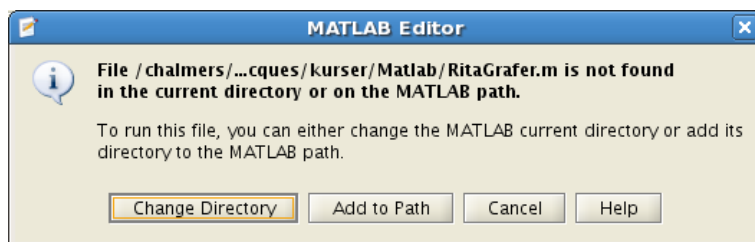
```
1 % Script RitaGrafer
2
3 x=linspace(0,2*pi);
4 f=sin(x);
5 g=sin(4*x);
6 plot(x,f,'green',x,g,'red')
7 xlabel('x')
8 ylabel('y')
9 title('sin(x) och sin(4x)')
10 grid on
11
```

Spara kan vi göra under File eller med diskett-symbolen, spara och köra kan vi göra under Debug. Enklast är dock att trycka på  som finns i verktygsfältet. Då sparas vår script, under ett namn vi väljer, 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 skriptfil 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.

Editor i MATLAB har något som kallas **Cell Mode** (cell-läge). Skriver man en kommentar som börjar med två procent-tecken, så avgränsar det en cell. Poängen är att man kan exekvera koden från en cell, istället för hela filen. På så sätt kan man dela upp en stor skriptfil (för ett helt övnings-tillfälle) i flera delar (varje deluppgift).

I cell-läge kan man evaluera aktuell cell genom att klicka på , evaluera aktuell cell och gå till nästa genom att klicka på . Samtliga val finns även under **Cell** i verktygsfältet.

## 5 Lite programmering

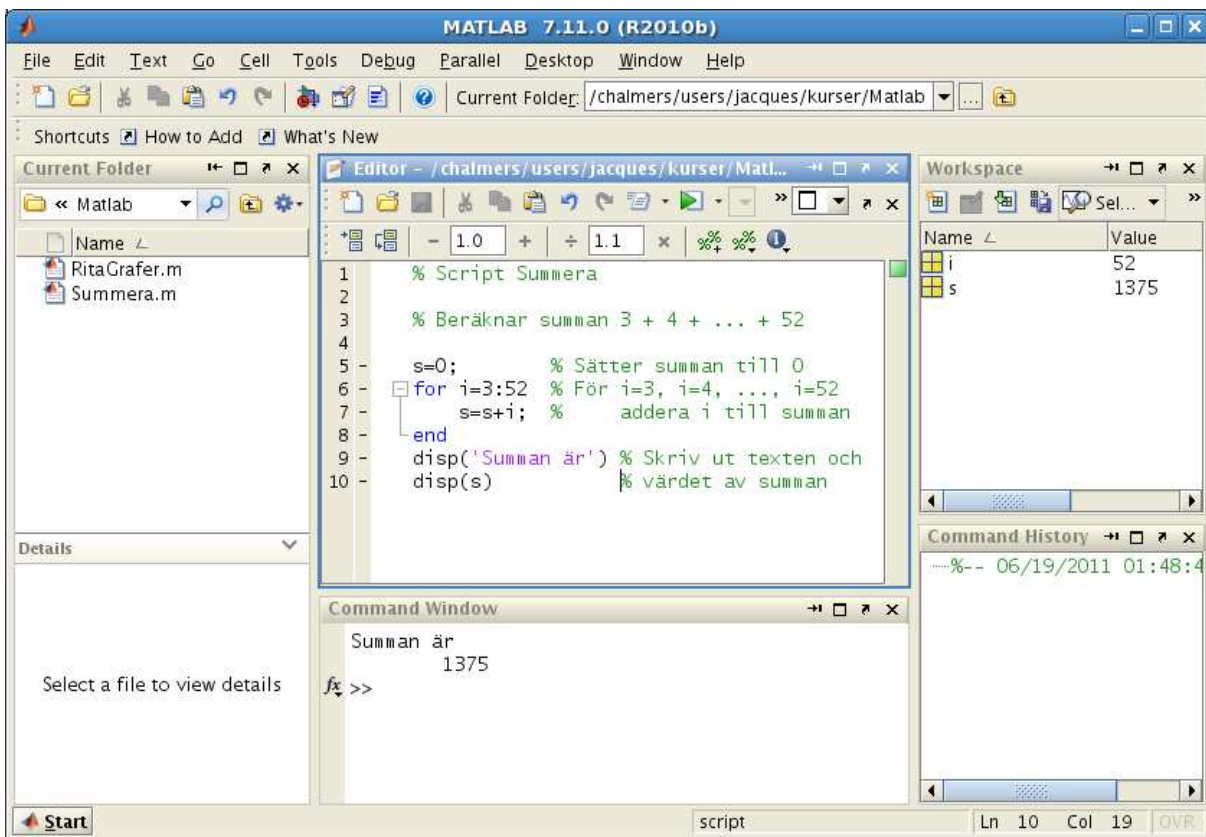
I MATLAB finns repetitions- och villkorssatser som påminner om motsvarande i programspråk som C och Java.

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  $s = 3 + 4 + 5 + \dots + 52$

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  $3 + 4 + 5 + \dots + 52$  med beteckningen

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

**Uppgift 5.** Skriv en script som beräknar summan

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

## 6 Function

Det finns olika sätt att göra egna funktioner i MATLAB. Om funktionen innehåller flera uttryck eller satser måste man göra en **function**, dvs. skapa en textfil med funktionsbeskrivningen. Består funktionen av ett enda uttryck så kan vi göra ett s.k. funktionshandtag (**function handle**) eller en s.k. anonym funktion.

En **function** är en textfil med samma namn som funktionen och som inleds med en funktionsdeklaration. I fortsättningen kommer vi ofta kalla en **function** för en *funktionsfil*.

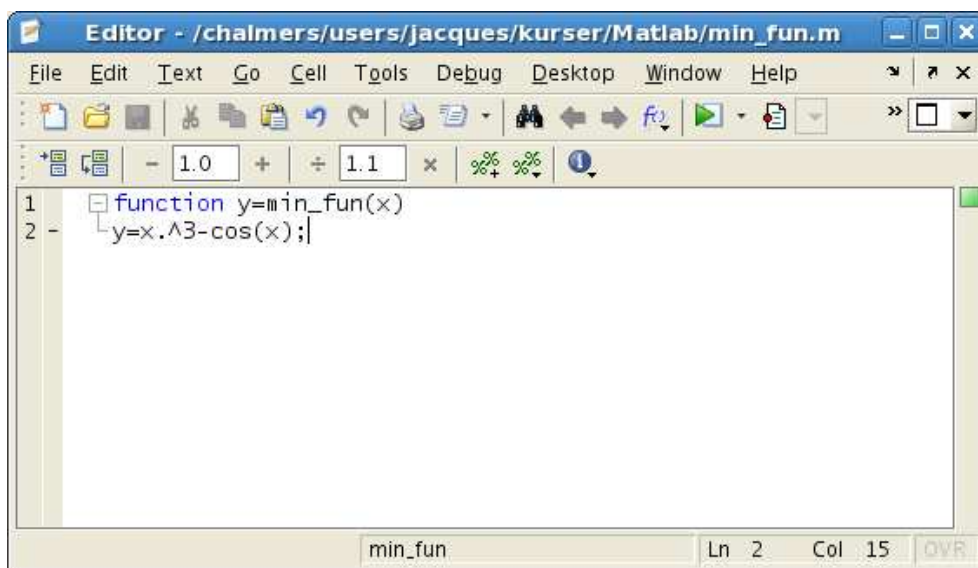
**Exempel 5.** Vi vill hitta ett nollställe till funktionen  $f(x) = x^3 - \cos(x)$ .

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

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

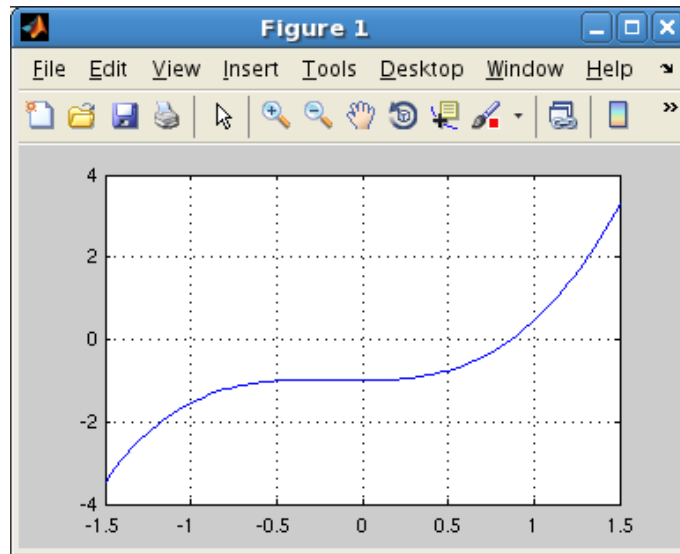
där **y** är funktionens värde (utdata), **x** är funktionens argument (indata) och **min\_fun** är funktionens namn.

Vi skriver in i editorn



Vi ritar grafen och använder **fzero** direkt i Command Window

```
>> x=linspace(-1.5,1.5);
>> y=min_fun(x);
>> plot(x,y)
>> 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 funktionsfil den skall arbeta med.

Alternativt använder vi en skriptfil, vilket vi vanligtvis kommer göra

```
1 %% Script Nollst
2
3 x=linspace(-1.5,1.5);
4 y=min_fun(x);
5 plot(x,y)
6 grid on
7 %%
8
9 x=fzero(@min_fun,1)
```

Lägg märke till att vi använde cell-läge i skriptfilen.

Skulle man råka försöka köra en funktionsfil (vilket händer ibland), så får man ett felmeddelande av typen `Input argument x is undefined`, eftersom inget argument gavs.



## 7 Desktop Layout

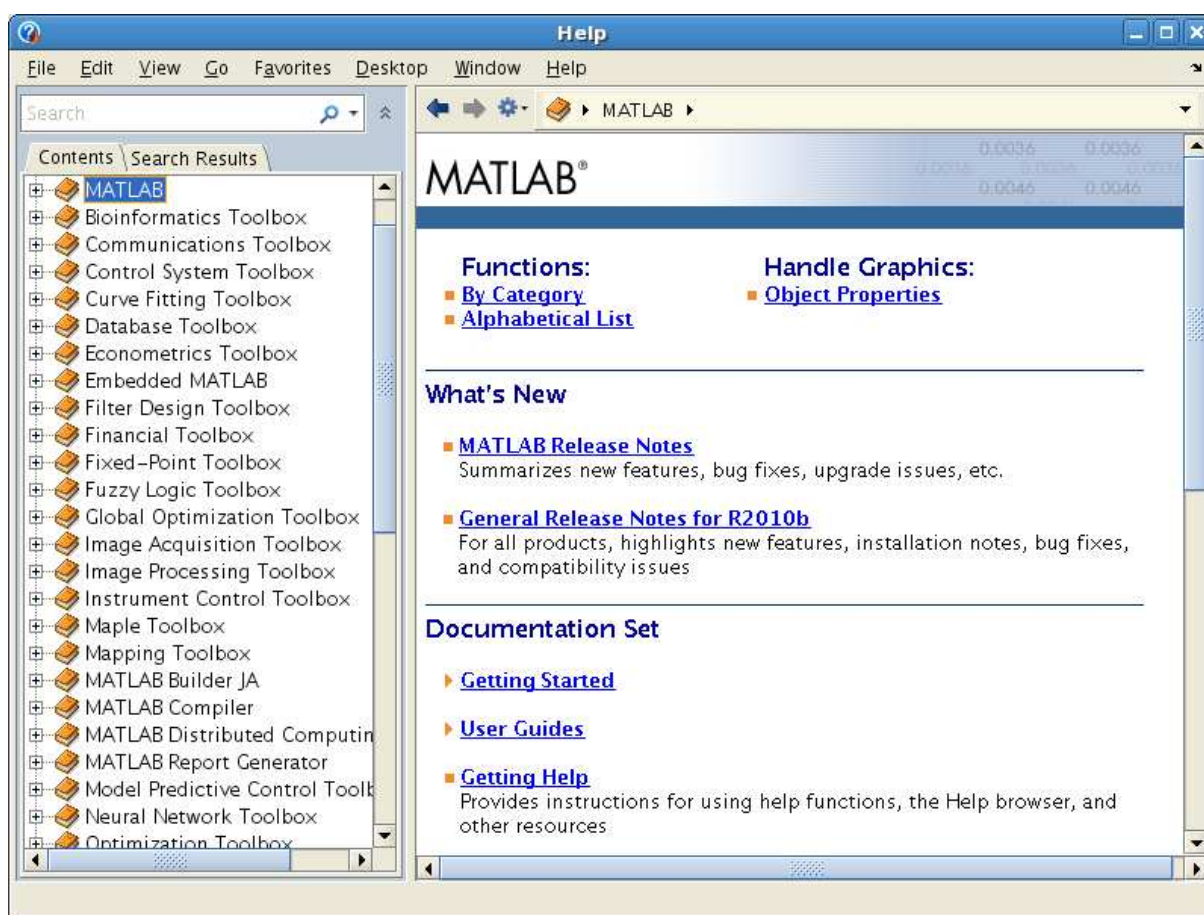
Som vi nämnde tidigare kallas MATLAB-fönstret för Desktop och dess konfiguration eller uppdelning för Desktop Layout. Man kan ändra standard layouten genom att ”klicka och dra” i delfönster, så att vi får en layout som vi trivs med.

Vi kan spara denna layout med ett lämpligt namn, vilket görs genom att välja **Save Layout ...** under **Desktop** i verktygsfältet. Om vi vill kan ha många olika layouter och vi kan alltid återvända till standard layouten.

Man kan ”docka” in eller ut delfönster, t.ex. MATLAB-editorn eller figurfönstret, och sedan ”dra” dem till rätt plats (om det behövs!). Att ”docka” in eller ut ett delfönster görs med de små pilar som finns uppe till höger i fönstren (strax intill ”krysset”).

## 8 Helpdesk i MATLAB

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



Vi ser den stora uppsättningen av verktygsfält, för olika tillämpningsområden, som följer med. Man kan söka sig fram för att hitta referenssidor (hjälpertexter) för olika kommandon och funktioner. Nästan alla inbyggda kommandon och funktioner har en referenssida.

Man kan också titta på dessa hjälpertexter med kommandon som ges i **Command Window**: `help` som ger texten i **Command Window** och `doc` som plockar fram aktuell referenssida i webläsaren. Texten

är ungefär samma, men för vissa kommandon (speciellt de för grafik så innehåller doc-sidan bilder, vilket kan vara till hjälp, medan `help` enbart visar ren text.

Det är viktigt att lära sig att läsa dokumentationen. Den är inte skriven för att lära ut till nybörjare hur man löser ett problem med MATLAB, utan för att visa den något vane användaren exakt hur en funktion eller ett kommando används. Det är inte lättläst, och man måste lära sig att plocka fram den information som är av intresse för tillfället, dvs. man måste lära sig att "skumma" texterna.

Nedan ser vi hjälptexten 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`.

