Introduktion till MATLAB

1 Inledning

MATLAB är både en interaktiv matematikmiljö och ett programspråk, som används på många tekniska högskolor runt om i världen. Med tiden har MATLAB blivit ett viktigt ingenjörsverktyg och har stor användning även inom industrin.

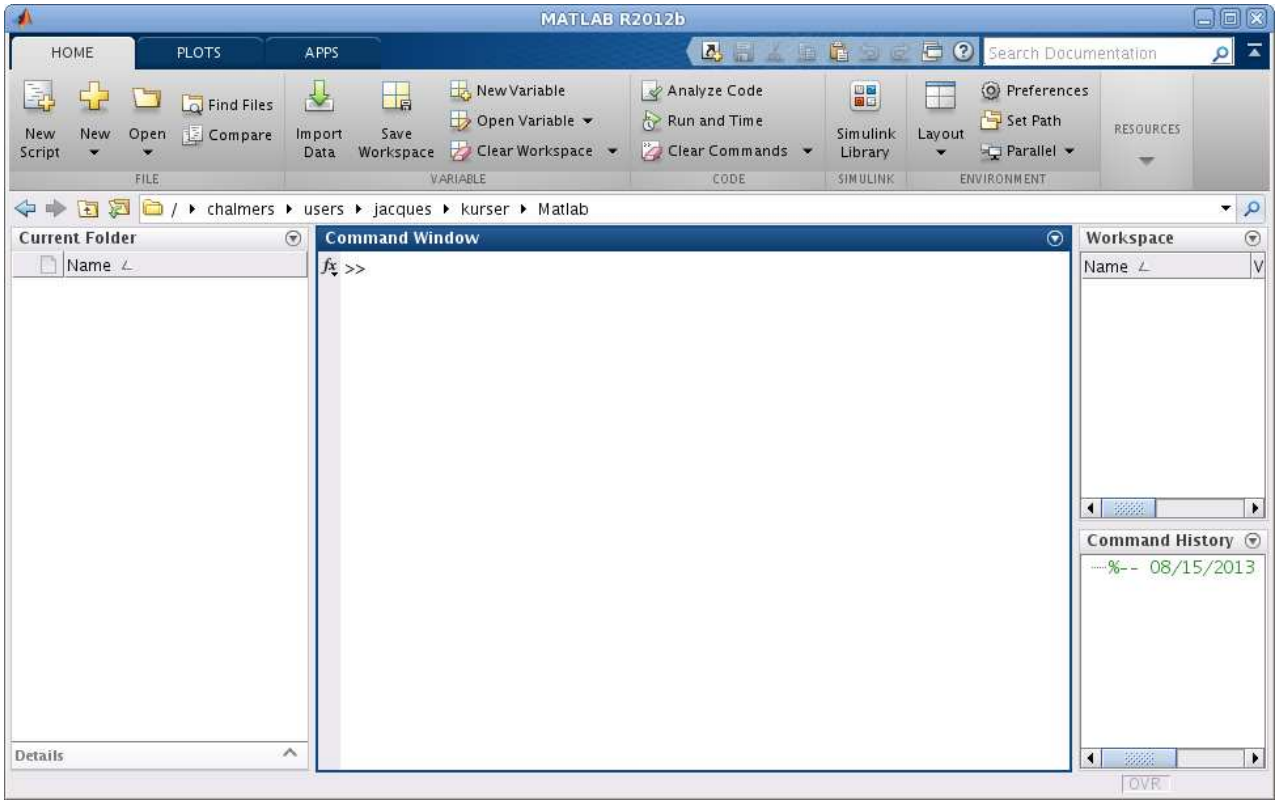
En av styrkorna med MATLAB är att systemet är utbyggbart med bibliotek eller verktygslådor, toolboxes, för olika tillämpningsområden.

Många studenter på de olika civilingenjörsprogrammen på Chalmers lär sig MATLAB och ni kommer använda MATLAB i många kurser i utbildningen. Det är viktigt att komma igång tidigt så att man hinner bli en tillräckligt erfaren användare.

2 Starta MATLAB

Vid en WINDOWS-dator startar man MATLAB genom att under WINDOWS-ikonen scrolla ned och välja MATLAB i listan av program. Sedan kommer MATLABs Desktop upp på skärmen.

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



Högst upp ser vi flikarna HOME, PLOTS och APPS. Det kommer bli fler när vi börjar arbeta, men mer om det senare.

De olika fönstren i Desktopen har namn (namnet står överst i fönstret). Det stora fönstret i mitten kallas Command Window och där kommer vi ge kommandon, till höger ser vi Workspace och Command History där vi ser vilka variabler vi har respektive vilka kommandon vi givit tidigare, slutligen till vänster ser vi Current Folder som visar innehållet i aktuell mapp eller katalog.

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.

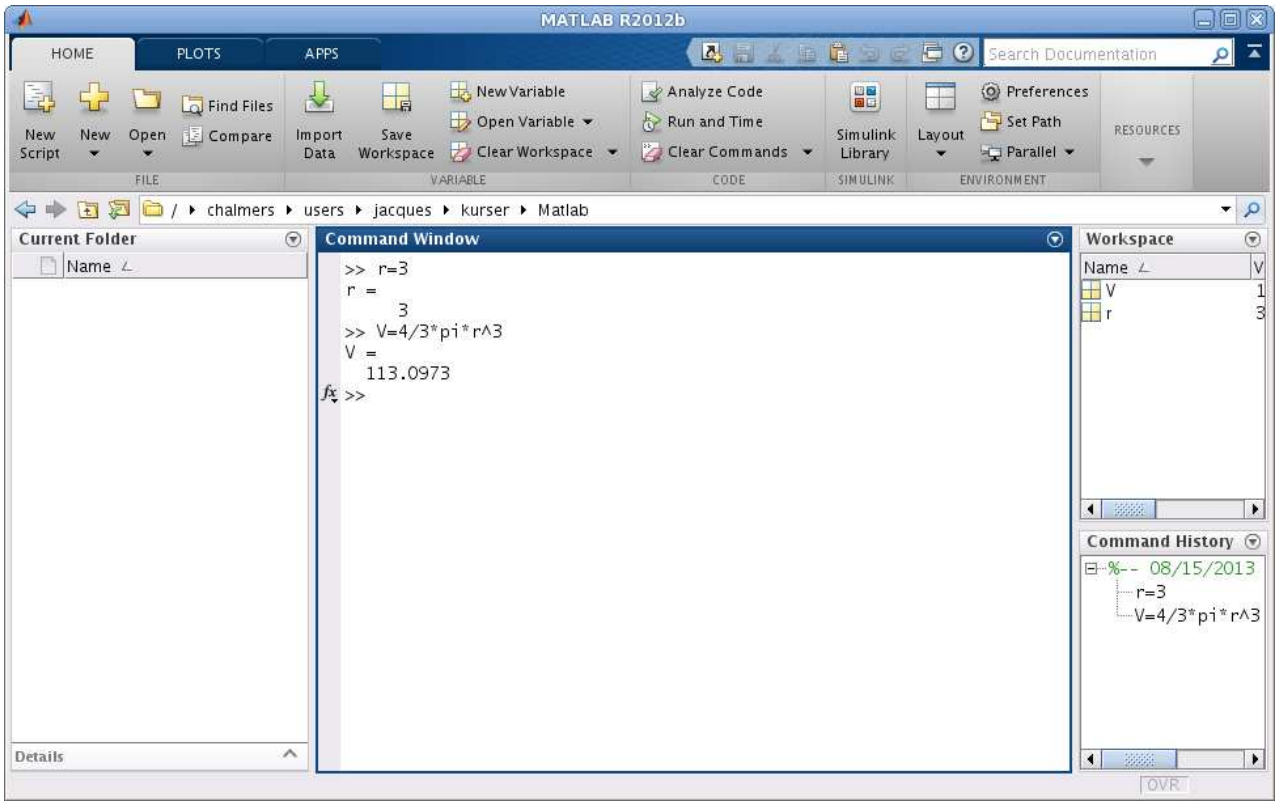
Exempel 1. Beräkna volymen av ett klot med radien $r = 3$ cm. Volymen ges av $V = \frac{4}{3}\pi r^3$.

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

```
>> r=3
```

Därefter beräknar vi volymen enligt formeln (`pi` ger en approximation av konstanten π) och låter variabeln `V` få detta värde.

```
>> V=4/3*pi*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. Vi ser våra variabler och deras värden i Workspace och i Command History ser vi kommandona vi givit så långt.

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 `x` av x-värden mellan 0 och 4π, med kommandot

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

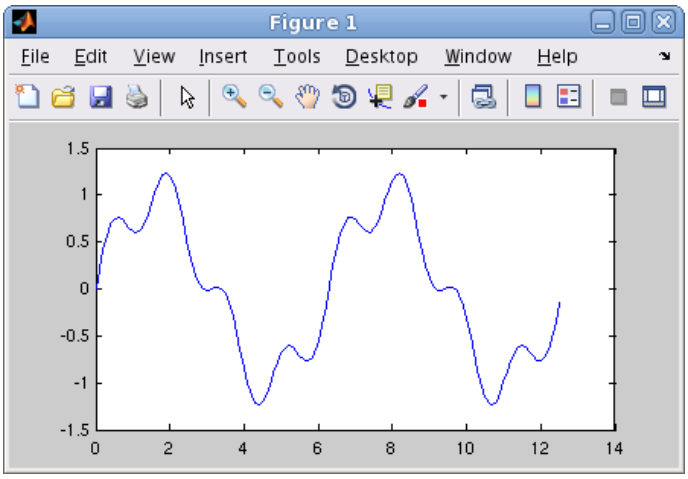
som vi skriver i Command Window.

Närmare bestämt får vi värdena 0, 0.1, 0.2, 0.3, $\cdots$, 12.5, dvs. värden med start i 0, steget 0.1 och slut så nära upp mot 4π som möjligt. Om vi hade inte skrivit ett semikolon (`;`) sist i uttrycket för `x`, hade alla x-värden skrivits ut i Command Window.

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 grafikfönster Figure kommer upp



Vi kan använda uppåtpil (↑) 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 (←), (→) och redigera kommandot. När kommandot ser ut som vi vill trycker vi på enter (↩).

Vill vi rensa Command Window så ger vi kommandot `clc` och med kommandot `clf` rensar vi Figure 1.

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

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);
```

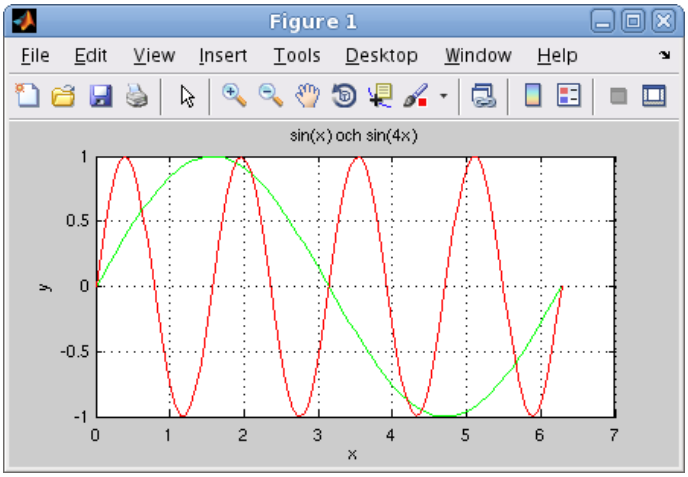
Vi ritar båda graferna samtidigt med `plot`, både paret `x`, `f` och paret `x`, `g`.

```
>> 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 (' '), t.ex. `'green'` och `'x'`, är s.k. textsträngar.

4 Något om matriser

Den grundläggande datatypen i MATLAB är matriser. En matris är ett rektangulärt talschema

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

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):

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

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

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

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]
```

Uppgift 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`.

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.

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

Vi bildar koefficientmatrisen $\boldsymbol{A}$ och högerledsvektorn $\boldsymbol{b}$ med

```
>> A=[1 2 3;3 2 1;7 8 0]
>> b=[14;10;23]
```

Med kommandot `rref` kommer vi till radreducerad trappstegsform (row-reduced-echelon form) så att vi kan läsa av lösningen till $\boldsymbol{Ax} = \boldsymbol{b}$.

```
>> R=rref([A b])
R =
     1     0     0     1
     0     1     0     2
     0     0     1     3
```

Lösningen ser vi i sista kolonnen i `R` och vi har

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Eftersom detta ekvationssystem har entydig lösning kan vi även lösa det med backslash-kommandot (\) så här enkelt `x=A\b`.

Som ytterligare ett exempel ser vi på följande ekvationssystem med oändligt många lösningar

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 10 \\ 3x_1 + 2x_2 + x_3 = 14 \\ 7x_1 + 8x_2 + 9x_3 = 46 \end{cases}$$

eller på matrisform

$$\boldsymbol{Ax} = \boldsymbol{b} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \\ 46 \end{bmatrix}$$

Vi bildar koefficientmatrisen $\boldsymbol{A}$ och högerledsvektorn $\boldsymbol{b}$ med

```
>> A=[1 2 3;3 2 1;7 8 9]
>> b=[10;14;46]
```

Vi reducerar utökande matrisen med

```
>> R=rref([A b])
R =
     1     0    -1     2
     0     1     2     4
     0     0     0     0
```

Vi har en fri variabel. Om vi sätter $x_3 = t$ får vi

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2+t \\ 4-2t \\ t \end{bmatrix}$$

där t är ett godtyckligt reellt tal.

Uppgift 4. Skriv följande ekvationssystem på matrisform och lös dem sedan med `rref`.

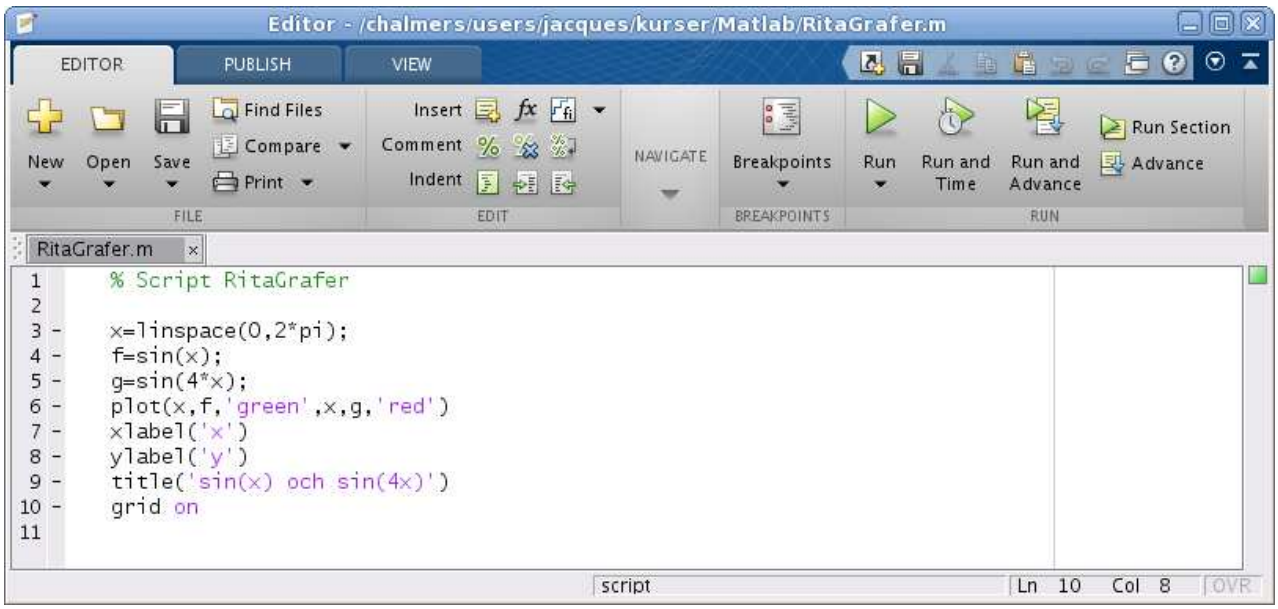
$$\begin{cases} x_1 + 5x_2 + 9x_3 = 29 \\ 2x_1 + 5x_3 = 26 \\ 3x_1 + 7x_2 + 11x_3 = 39 \end{cases} \qquad \begin{cases} x_1 + x_2 + 3x_3 + 4x_4 = 2 \\ -2x_1 + 2x_2 + 2x_3 = -4 \\ x_1 + x_2 + 2x_3 + 3x_4 = 1 \\ x_1 - x_2 - 2x_3 - x_4 = 1 \end{cases}$$

Skulle det finnas oändligt många lösningar skriv upp en formel för samtliga lösningar.

6 Script

För att slippa skriva om sina kommandon, eller bläddra med uppåt- och nedåtpilar (↑), (↓) i kommandofönstrets historik eller dra från Command History, så brukar man skriva ett script. Ett script är en textfil som innehåller det man skulle kunna skriva direkt vid promptern (>>) i Command Window, och som utförs i MATLAB då man ger textfilens namn som kommando.

Som exempel ser vi på ett script för exempel 3 gjort med den i MATLAB inbyggda editorn.

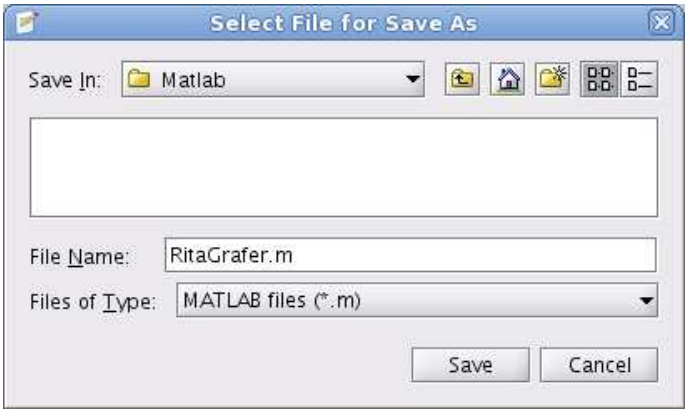


Editorn i MATLAB startas genom att man trycker på New Script eller det stor plustecknet på HOME-fliken (se Desktopen i avsnitt 2 eller 3).

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

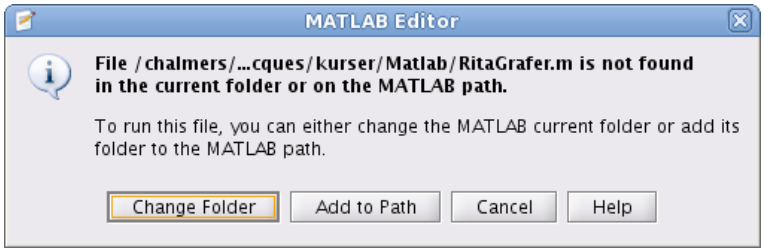
Spara kan vi göra under Save på EDITOR-fliken eller med diskett-symbolen i verktygsfältet och köra kan vi göra genom att trycka på  som finns på EDITOR-fliken. Då sparas vårt script, under ett namn vi väljer, och utförs som om vi gav namnet som ett kommando. När scriptet körs kommer MATLAB att utföra rad för rad (med start från första raden), och vi kommer få samma grafer som i exempel 3.

Så här ser dialogrutan ut som kommer upp då vi skall namnge vårt script.



Utanför MATLAB får namnet på ett script tillägget `.m` för att skilja denna typ av fil från andra filer.

För att MATLAB skall hitta filen, krävs det att katalogen där filen ligger är aktuell katalog. Om man försöker köra ett script som ligger i en annan katalog än den aktuella, så får man upp en fråga om att byta till den katalogen:



Välj Change Folder så byter MATLAB katalog.

Man kan byta katalog genom att antingen klicka sig fram i Current Folder eller använda navigeringsfältet precis under flikarna.

Editor i MATLAB har något som kallas Cell Mode (cell-läge). Inleder man en rad med två procenttecken följt av ett blanktecken (`%% `), så avgränsar det en cell. Poängen är att man kan köra koden från en cell, istället för hela filen. På så sätt kan man dela upp en stort script (för en hel datorövning) 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  eller bara gå till nästa med  . Samtliga val finns till höger på EDITOR-fliken.

I fortsättningen kommer vi ofta kalla ett script för en *skriptfil*.

7 Lite programmering

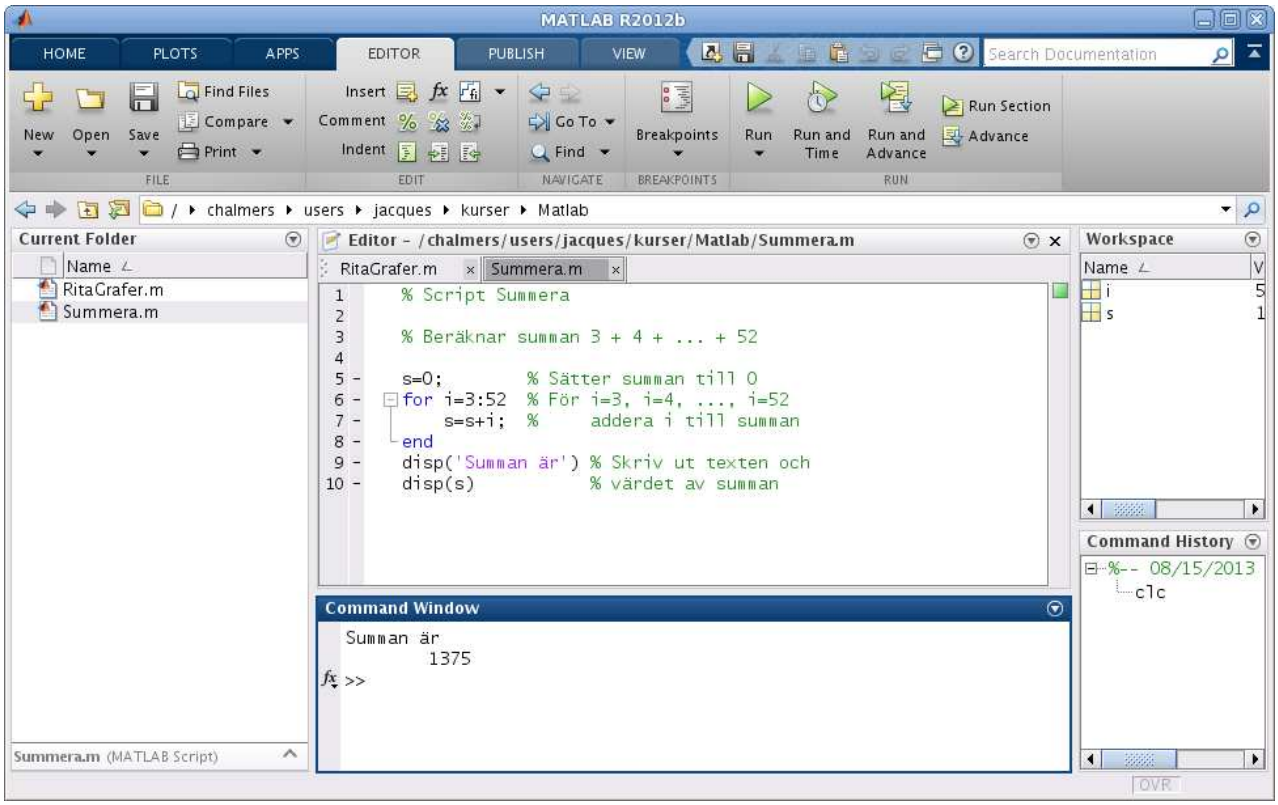
I MATLAB finns repetitions- och villkorssatser som påminner om motsvarande i programspråk som C och Java.

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 ett 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

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

Uppgift 5. Skriv ett script som beräknar summan

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

8 Function

Det finns flera 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 kan vi göra en anonym funktion (anonymous function) med ett funktionshandtag (function handle).

För större program kan man även vilja använda andra sätt att skriva funktioner, exempelvis underfunktioner (subfunction) eller nästlade funktioner (nested function), men vi lämnar det så länge.

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 $f(x) = x^3 - \cos(x)$.

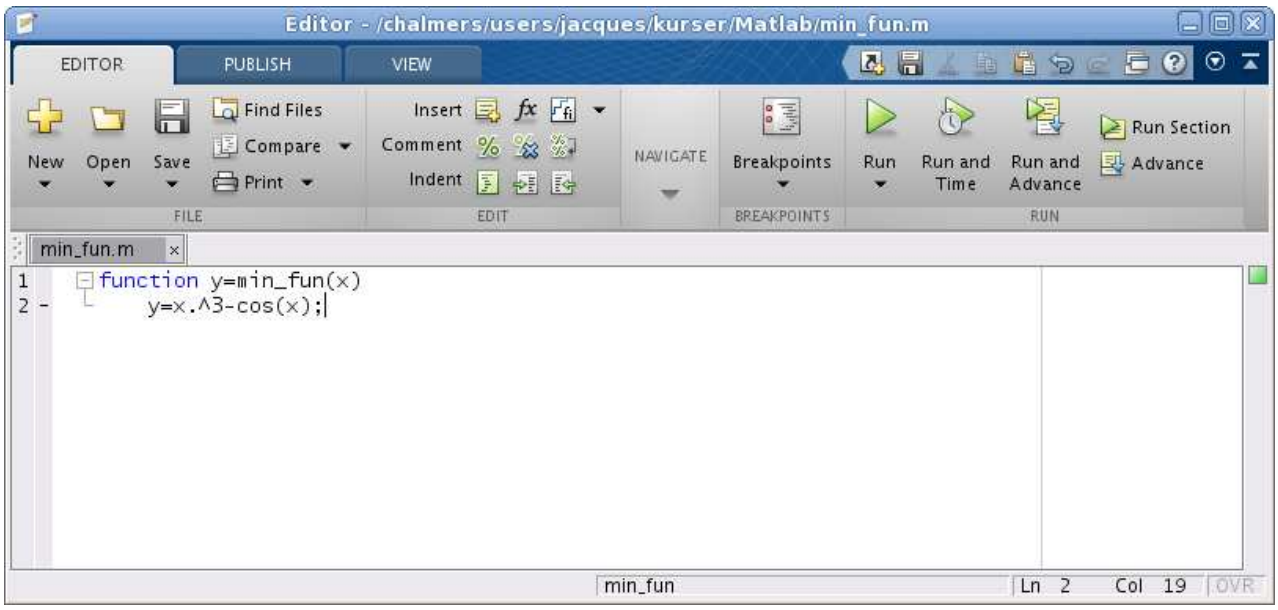
Det finns en funktion `fzero` i MATLAB som hittar nollställen. För att använda `fzero` måste vi först beskriva vår funktion och det gör vi som en function enligt

```
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 (som vi själva valt).

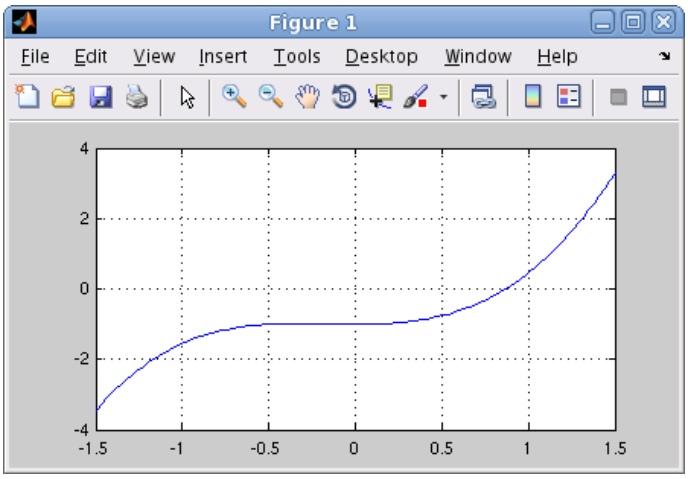
Vi skriver x^3 som `x.^3` i MATLAB eftersom vi vill att `x` skall kunna vara en lista eller radvektor med många x-värden och vill då att varje enskilt x-värde, dvs. varje element eller komponent, skall upphöjas till 3. Detta är en s.k. *elementvis* operation.

Vi skriver in funktionen i editorn och sparar den under namnet `min_fun` på samma sätt som för ett `script`, dvs. textfilen skall heta `min_fun.m` i katalogen.



Vi ritar grafen genom att direkt i Command Window skriva

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

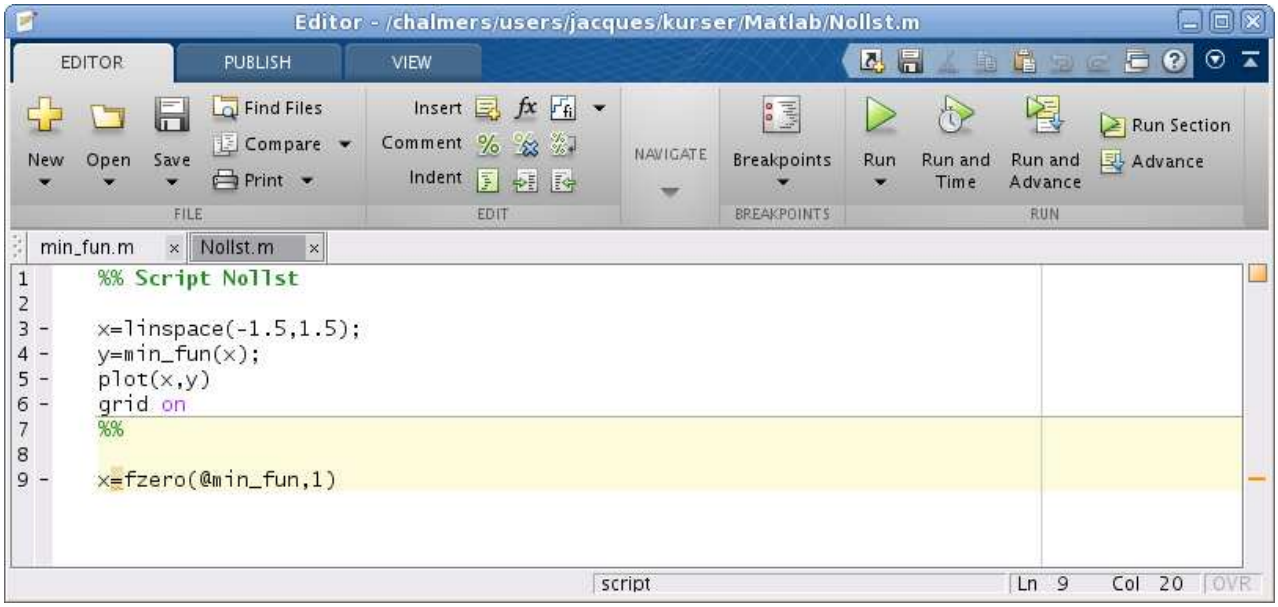


Vi ser att vi har ett nollställe nära $x = 1$ och låter `fzero` beräkna nollstället noggrant med

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

Med `@min_fun` talar vi om för `fzero` vilken function som skall användas, dvs. vilken funktion det skall sökas nollställe till.

Vanligtvis kommer vi använda vi ett script. Lägg märke till att vi använder cell-läge.



Utskriften av beräkningsresultatet ovan gjordes med 4 decimaler. Vill vi få fler decimaler utskrivna kan vi ge kommandot `format long` innan utskriften och får då 15 decimaler i resultatet. Med `format short e` och `format long e` får vi s.k. scientific notation och med `format short` får vi tillbaka den korta varianten.

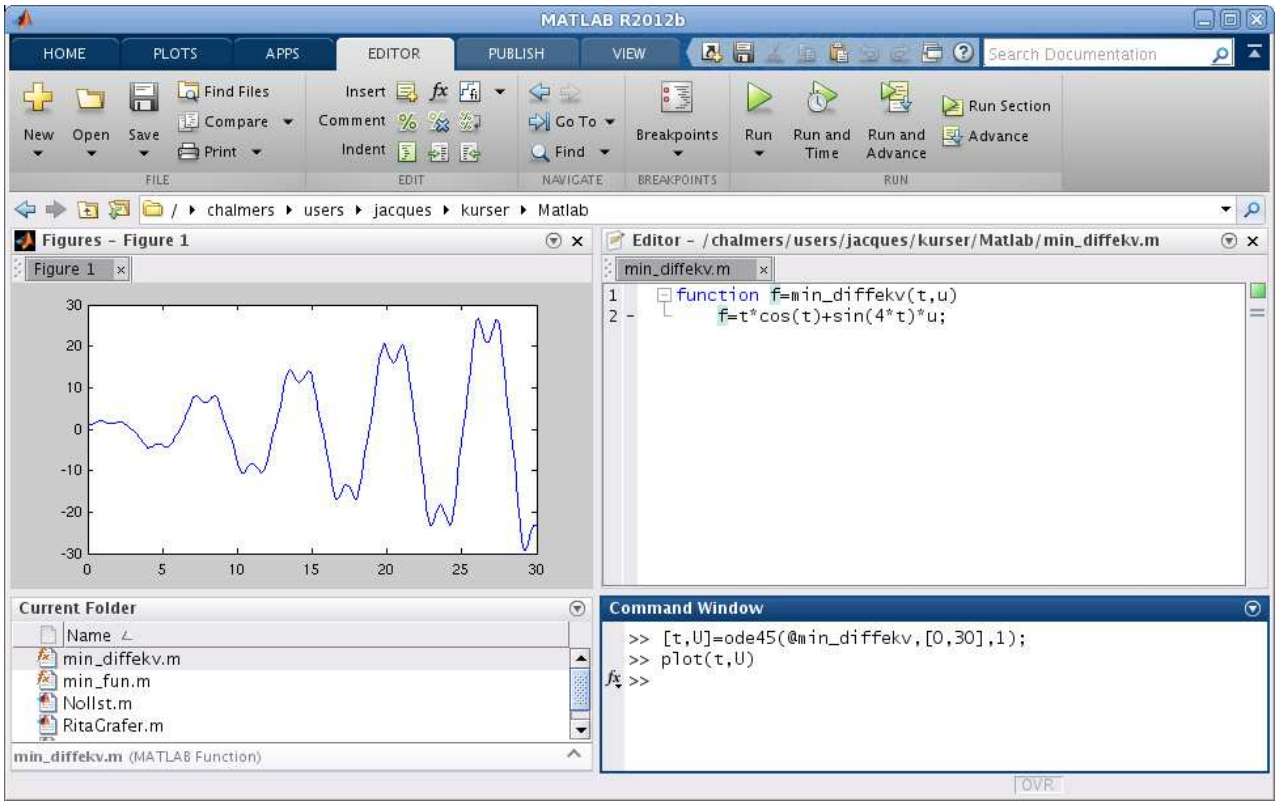
Uppgift 6. Hitta alla nollställen till funktionen $f(x) = x^2 - \cos(x)$. Gör en function som beskriver vår funktion och rita en graf. Använd sedan `fzero` för att beräkna varje nollställe, ett i taget.

Glöm inte att skriva x^2 som `x.^2` i MATLAB om `x` är en lista eller radvektor. Tänk på att funktionen måste skrivas i en egen textfil.

Exempel 6. Vi vill beräkna och rita lösningen till differentialekvationen

$$\begin{cases} u' = t\cos(t) + \sin(4t)u, \ 0 \le t \le 30 \\ u(0) = 1 \end{cases}$$

Vi beskriver högerledet i differentialekvationen med en funktion och sedan beräknar vi lösningen med funktionen `ode45` enligt



Lägg märke till hur vi anger intervallet för t och hur vi ger det s.k. begynnelsevärdet $u(0) = 1$.

9 Desktop Layout

När man startar MATLAB får man en standard desktop layout. Man kan ändra denna layout genom att "docka" in de fönster man vill ha på sin desktop och sedan "dra" dem till rätt plats (om det behövs). Att "docka" in eller ut ett MATLAB-fönster görs med de små pilar som finns uppe till höger i fönstren (strax intill "krysset"). Man kan sedan spara sin layout med ett lämpligt namn, genom att välja Save Layout ... under Layout på HOME-fliken.

I texterna till kommande datorövningar kommer vi använda den layout som ni ser i exempel 6 ovan. Den passar bättre för oss än den som är standard.

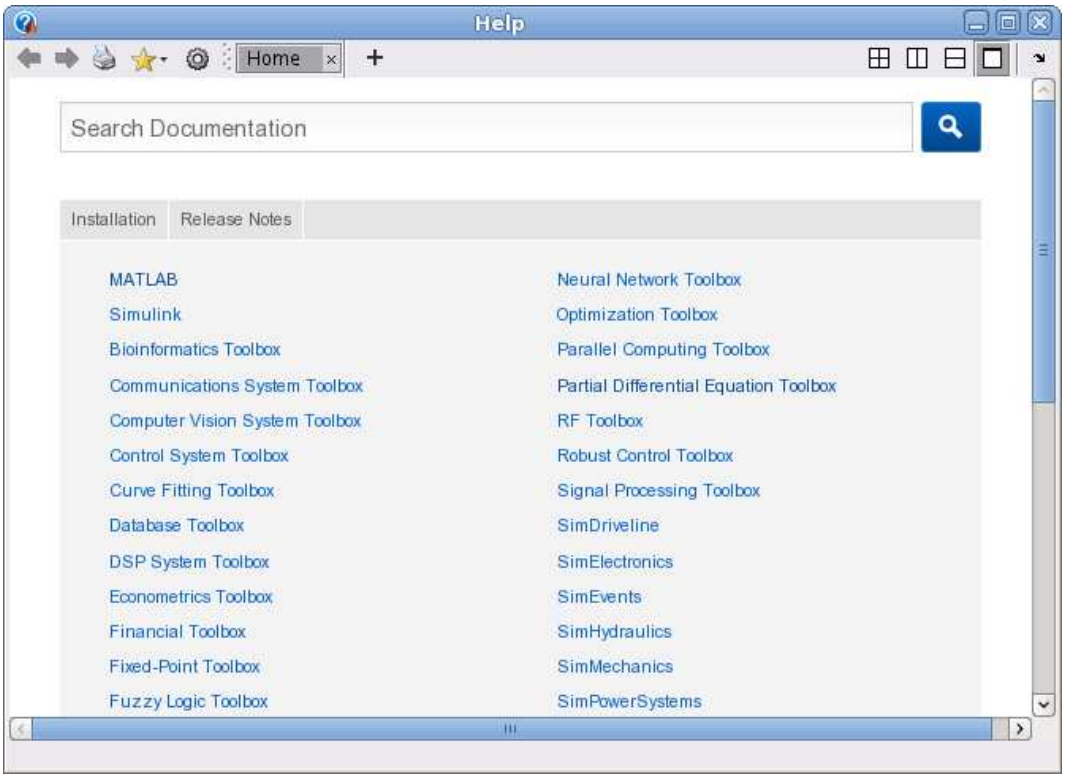
För att handledning och redovisning skall fungera effektivt kräver vi att all redovisning görs via en sammanhållande skriptfil i cell-läge (Cell Mode) tillsammans med nödvändiga funktionsfiler. Vi ser gärna att ni har en MATLAB desktop layout av det slag som visas i exempel 6.

Fördelarna med detta är att man får en bra (översiktlig och effektiv) interaktiv miljö för att utveckla program och för tolkning av resultat. En bieffekt blir dessutom att både handledning och redovisningar blir effektivare.

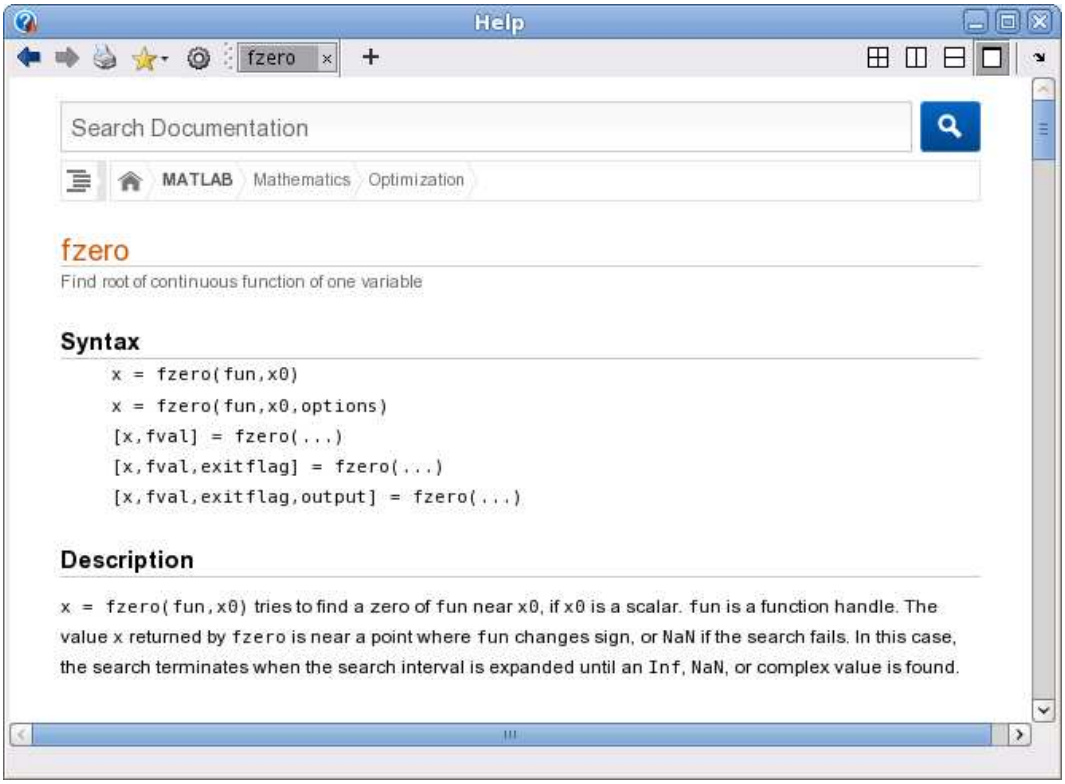
Uppgift 7. Gör en desktop layout som ser ut ungefär som den i exempel 6. Spara denna layout med ett lämpligt namn. Denna layout skall sedan helst användas vid redovisningar.

10 Help i MATLAB

Den mest utförliga och aktuella beskrivningen av MATLAB finns i det inbyggda hjälptextsystemet. Tryck på i verktygsfältet eller på HOME-fliken och Help-fönstret öppnas.



Vi ser den stora uppsättningen av verktygslådor, för olika tillämpningsområden, som följer med. Man kan söka sig fram för att hitta hjälptexter (referenssidor) för olika kommandon och funktioner. Här ser vi t.ex. hjäptexten för funktionen `fzero`.



Läs gärna i texten (skriv `fzero` i sökrutan och tryck på enter) och titta tillbaka på exempel 5 där vi använde `fzero`, och leta gärna upp hjälptexten för `ode45` som vi använde i exempel 6.

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 informationen som är av intresse för tillfället, dvs. man måste lära sig att "skumma" texterna.

11 Import av data

För att hantera mätdata behöver man bl.a. kunna hantera olika fil-format. MATLAB har en Import Wizard som man kan läsa om i Help.