# Introduktion till MATLAB

## 1 Inledning

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

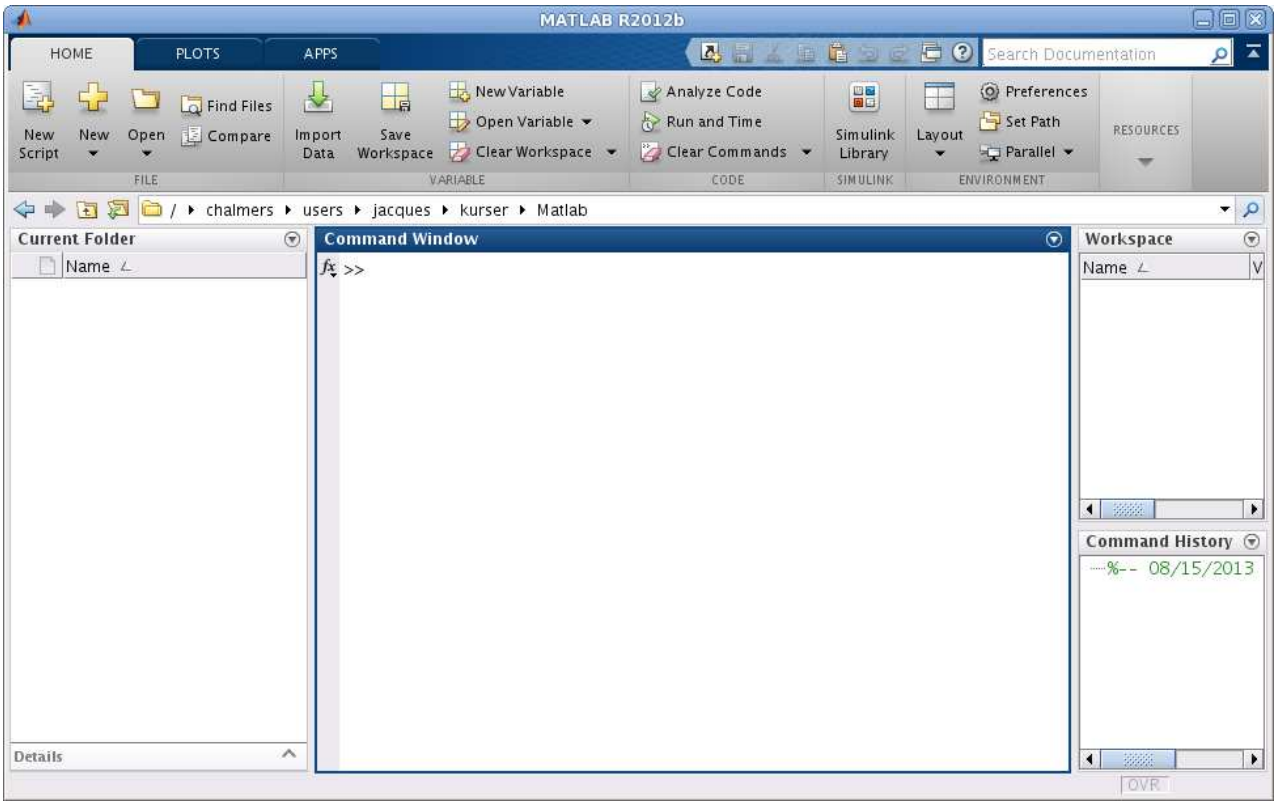
En av styrkorna med MATLAB är att systemet är utbyggbart med bibliotek eller verktygslådor, toolboxes, för olika tillämpningsområden.

Alla studenter på de olika civilingenjörsprogrammen på Chalmers lär sig MATLAB och ni kommer använda MATLAB i flera 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 funktionsgrafer

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.

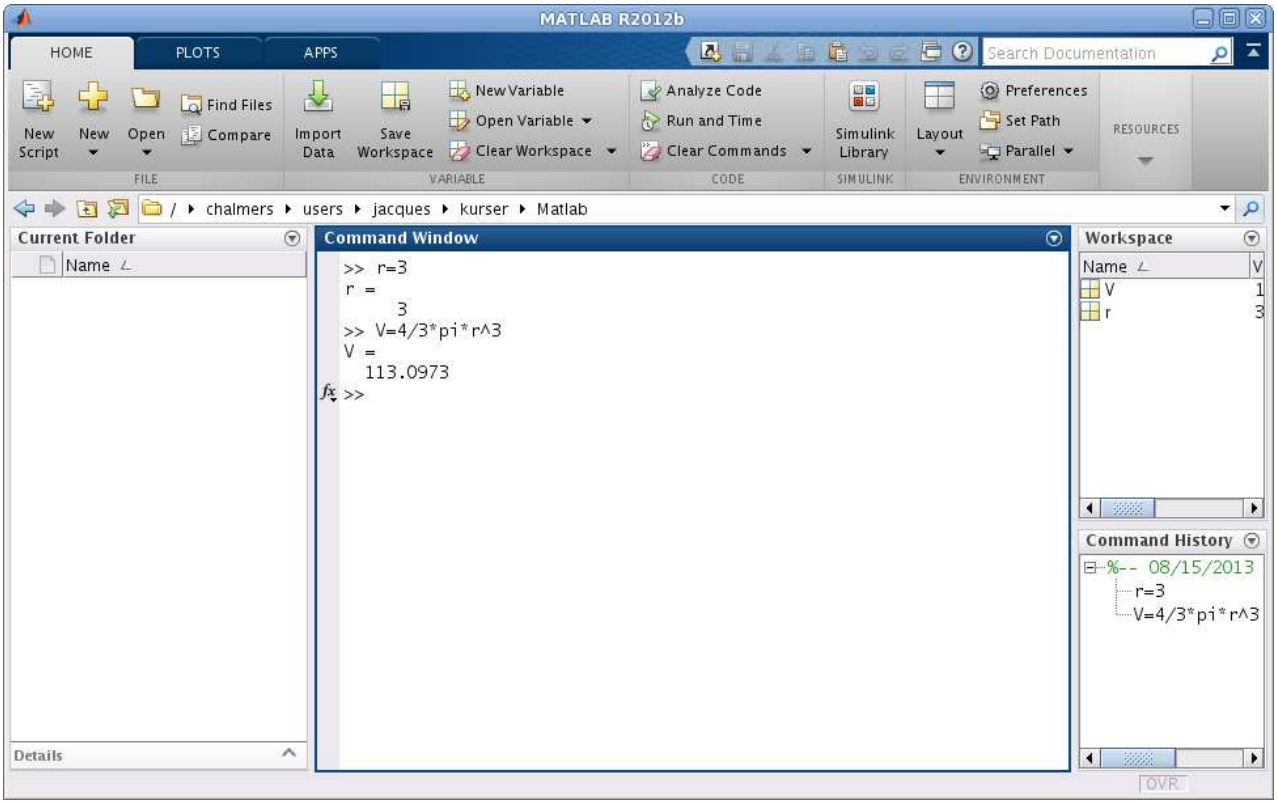
**Exempel 1.** Beräkna volymen av ett klot med radien $r = 3$ cm. Volymen ges av $V = \frac{4}{3}\pi r^3$.

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 $\pi$) 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 $4\pi$, med kommandot

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

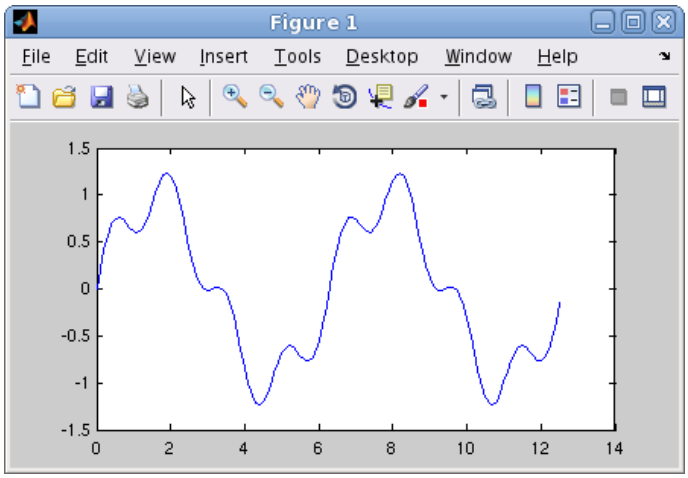
som vi skriver i Command Window.

Närmare bestämt får vi värdena 0, 0.1, 0.2, 0.3, $\cdots$ , 12.5, dvs. värden med start i 0, steget 0.1 och slut så nära upp mot $4\pi$ 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 gör videt med 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 $2\pi$, då blir graferna jämna och snygga.

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

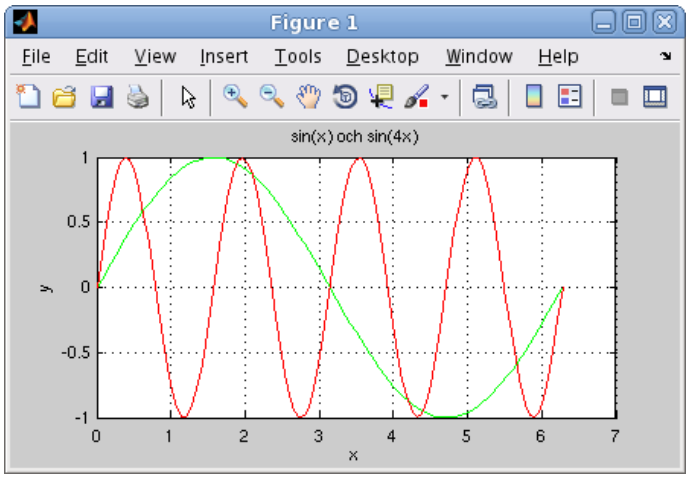
Vi ritar båda graferna samtidigt med `plot`, både paret `x`, `f` och paret `x`, `g`.

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

För att skilja graferna åt gjorde vi $\sin(x)$-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
```

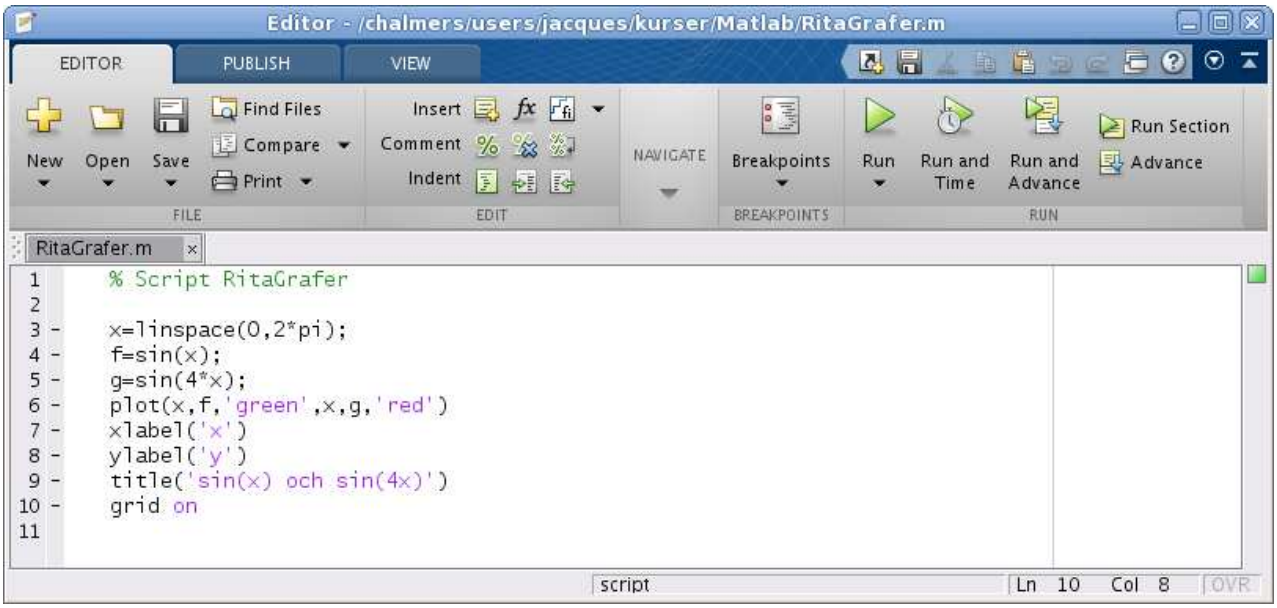



Texterna inom apostrofer (' '), t.ex. `'green'` och `'x'`, är s.k. textsträngar.

# 4 Skriptfiler

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 eller skriptfil. 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.

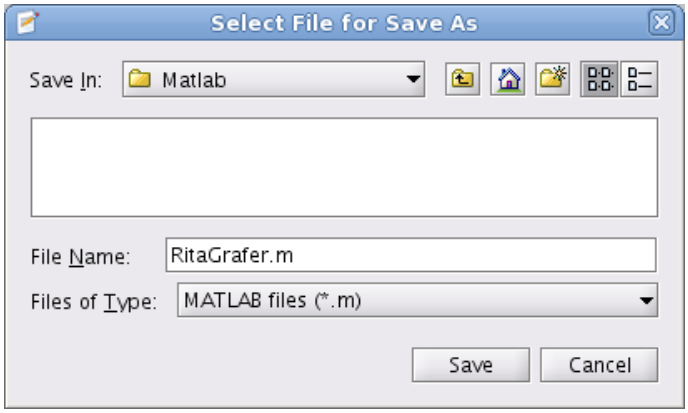



Editorn i MATLAB startas genom att man trycker på New Script eller det stor plustecknet på HOME-fliken (se Desktopen i avsnitt 2 eller 3).

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

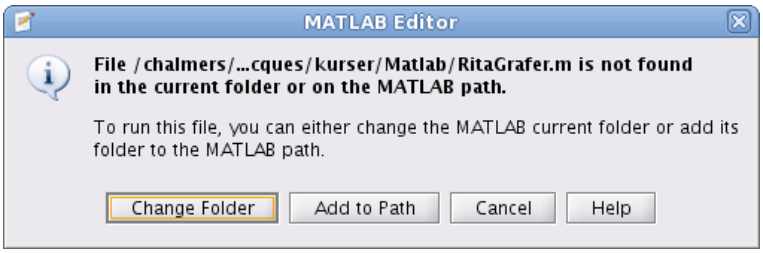
Spara kan vi göra under Save på EDITOR-fliken eller med diskett-symbolen i verktygsfältet och köra kan vi göra genom att trycka på  som finns på EDITOR-fliken. Då sparas vårt script, under ett namn vi väljer, och utförs som om vi gav namnet som ett kommando. När scriptet körs kommer MATLAB att utföra 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 studioövning) 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.

# 5 Lite programmering

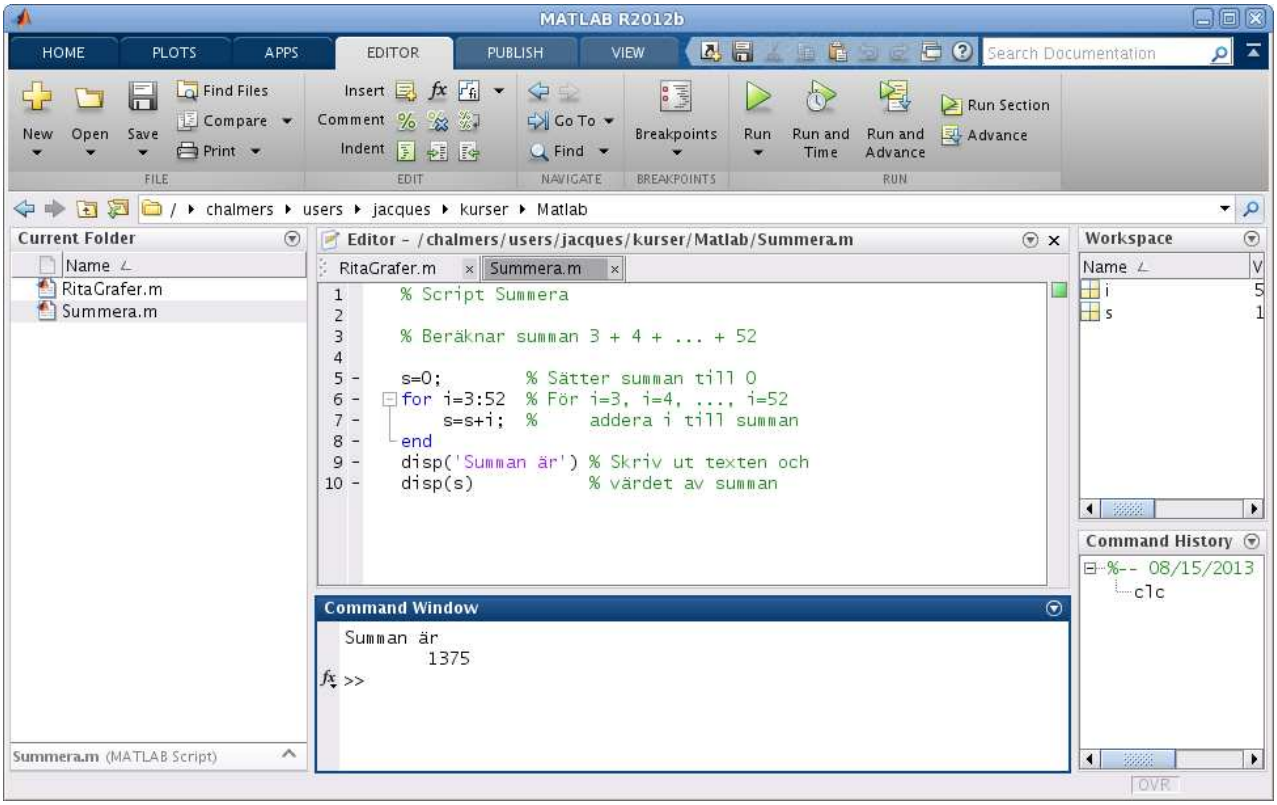
I MATLAB finns repetitions- och villkorssatser som påminner om motsvarande i programspråk som C och Java.

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 $s = 3 + 4 + 5 + \cdots + 52$.

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 3.** Skriv ett script som beräknar summan

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

# 6 Funktioner

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 en s.k. anonym funktion (anonymous function).

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.

Vi ser på ett exempel där vi gör en function vilket är grundtypen för egna funktioner.

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

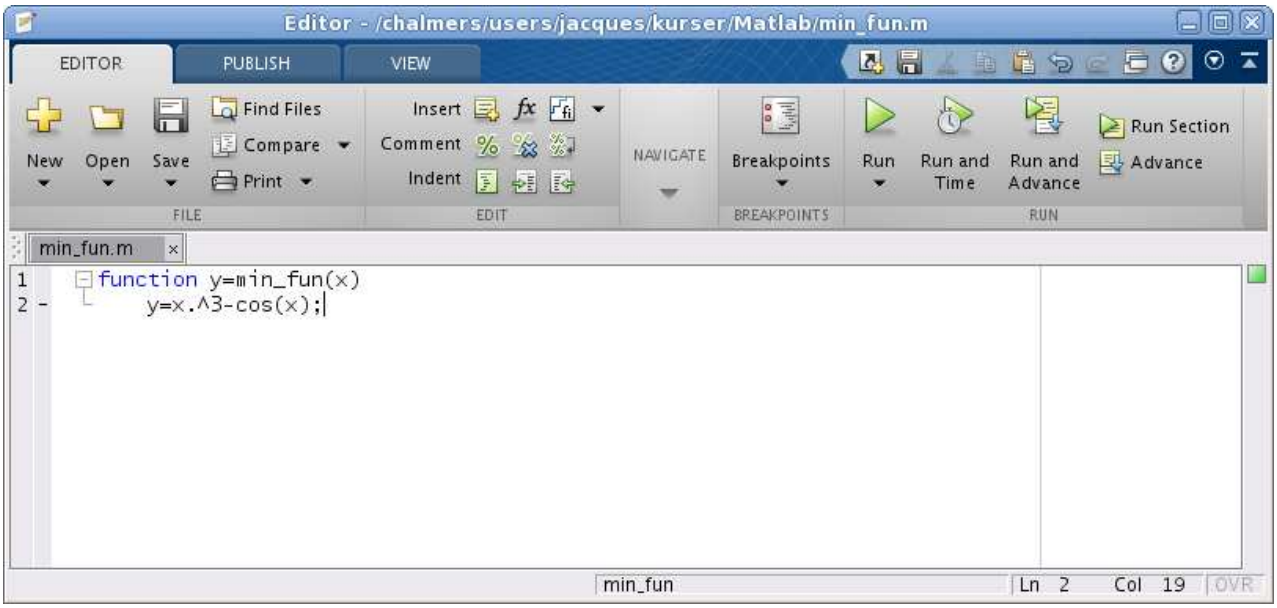
Det finns en funktion `fzero` i MATLAB som hittar nollställen. För att använda `fzero` måste vi först beskriva vår funktion och det gör vi som en function enligt

```
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).

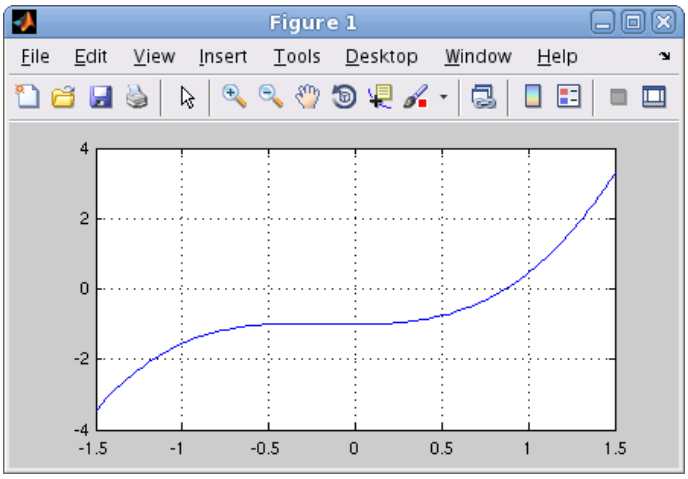
Vi skriver $x^3$ som `x.^3` i MATLAB eftersom vi vill att `x` skall kunna vara en lista eller radvektor med många $x$-värden och vill då att varje enskilt $x$-värde, dvs. varje element eller komponent, skall upphöjas till 3. Detta är en s.k. *komponentvis* operation.

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
```

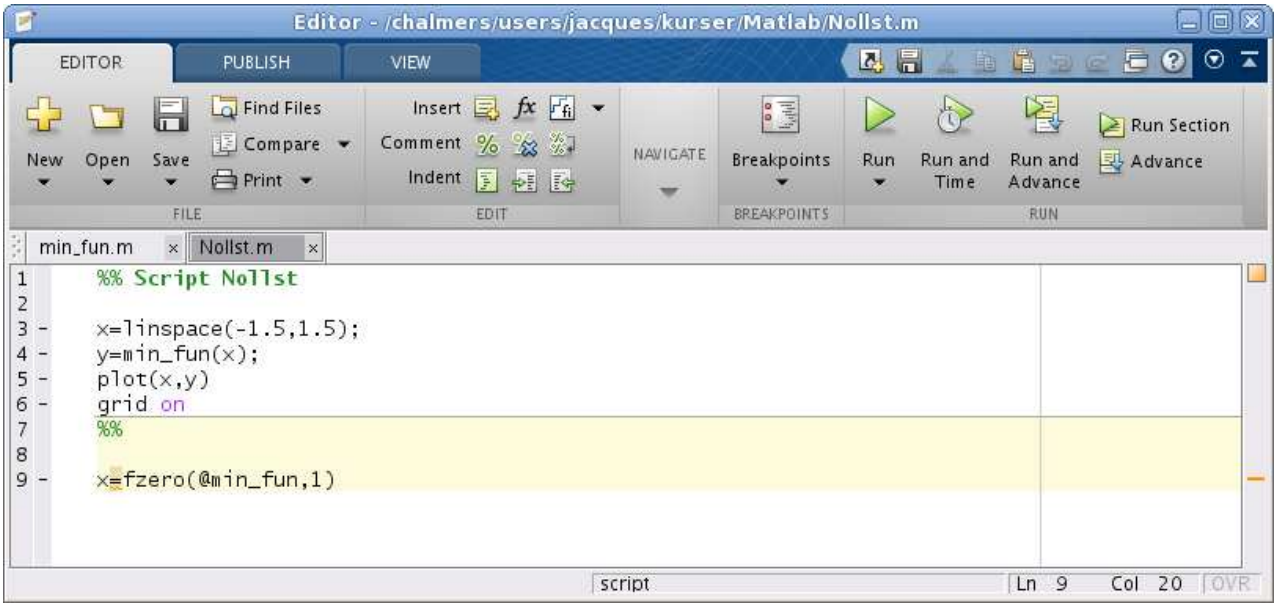



Vi ser att vi har ett nollställe nära $x = 1$ och låter `fzero` beräkna nollstället noggrant med

```
>> 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.




**Uppgift 4.** 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.

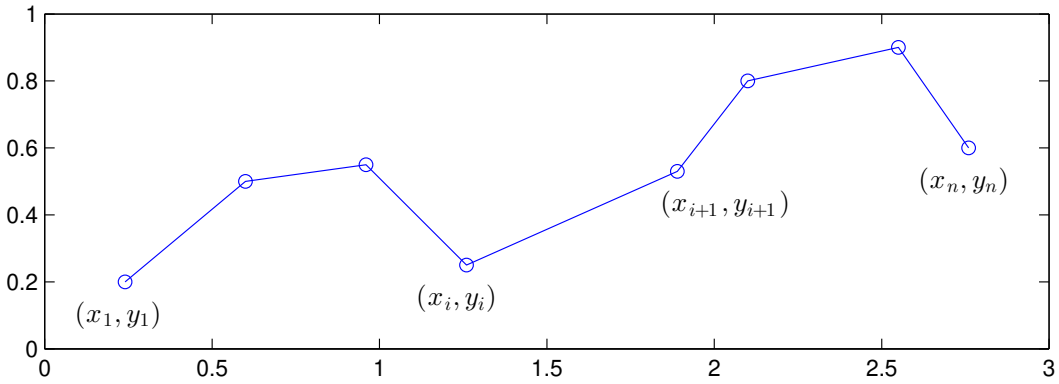
# 7 Lite mer om grafritning

Ibland vill man rita flera funktionsgrafer 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 kan lägga på ett rutnät med `grid on` (vi gjorde det i exempel 3) och ta bort det igen med `grid off`, om vi vill det.

Man kan ibland vilja ha flera koordinatsystem i samma figur-fönster (Figure). Då använder man kommandot `subplot`, som vi snart skall se i ett exempel.

En funktionsgraf är ett polygontåg, dvs. en följ av punkter $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, som vi successivt förbinder med räta linjer.

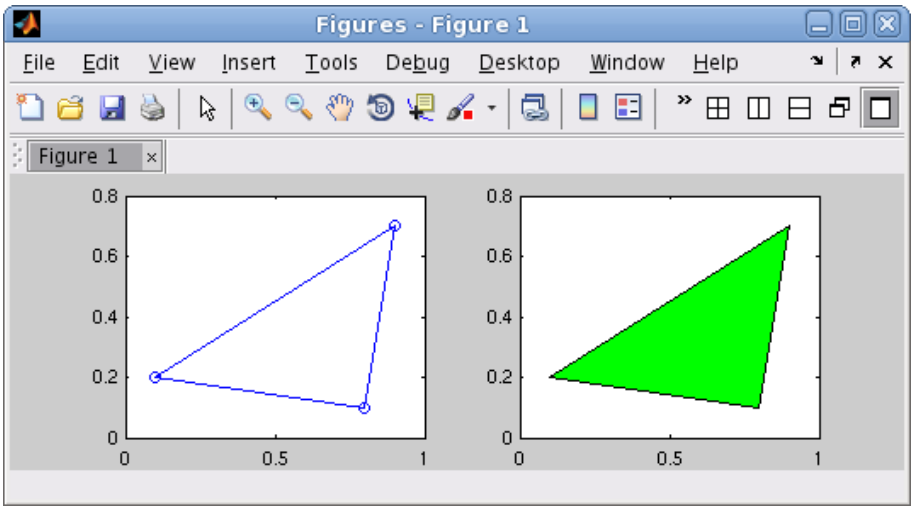



Vi kan även rita ett polygontåget som inte är en funktionsgraf, t.ex. en triangel, en rektangel eller en cirkel.

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.

**Exempel 6.** Vi ritar triangeln som ges av polygontåget $(0.1, 0.2), (0.8, 0.1), (0.9, 0.7), (0.1, 0.2)$.

```
>> x=[0.1 0.8 0.9 0.1];
>> y=[0.2 0.1 0.7 0.2];
>> subplot(1,2,1)      % Delar in Figure i 1x2 delar och gör 1:a aktivt
>> plot(x,y,'-o')      % '-o' förbinder med räta linjer och markerar med ringar
>> axis([0 1 0 0.8])   % Ger lite "luft' runt triangeln
```

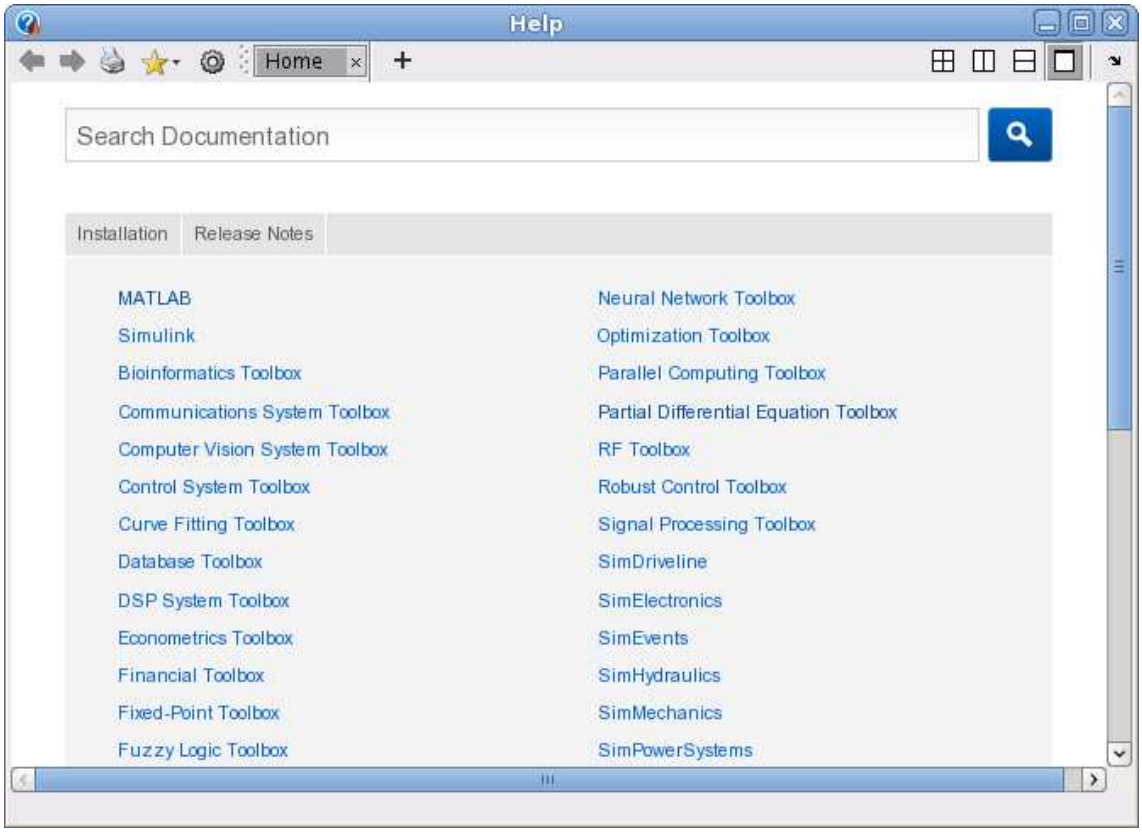



```
>> subplot(1,2,2)      % Delar in Figure i 1x2 delar och gör 2:a aktivt
>> fill(x,y,'g')       % Fyller triangeln med grön färg
>> axis([0 1 0 0.8])
```

# 8 Hjälpsystemet

Den mest utförliga och aktuella beskrivning som finns av MATLAB hittar man i det inbyggda hjälpsystemet Help.

Tryck på  i verktygsfältet eller på fliken HOME och ett fönster Help öppnas.




Vi ser då den stora uppsättningen av verktygslådor, för olika tillämpningsområden, som följer med. Man kan söka sig fram för att hitta hjälptexter för olika kommandon och funktioner.

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 exakt hur en funktion eller ett kommando används. Man måste lära sig att ”skumma” texterna.

**Uppgift 5(a).** Leta upp och läs hjälptexten för `linspace` som vi använde i samband med grafritning. Hur anger man antal punkter man vill ha? Hur många punkter får man som standard om man inte anger något antal?

**(b).** Leta upp hjälptexten för `fzero` som vi använde i exempel 5.

# 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 spara sin layout med ett lämpligt namn, genom att välja Save Layout ... under Layout på HOME-fliken.

# 1 Målsättning

Avsikten eller målsättningen med laborationen är i stort sett att vi skall komma igång med MATLAB och rita några funktionsgrafer. Vidare skall vi bekanta oss med script, hur vi skapar och namnger sådana. Vi skall också känna till function något, hur vi skapar och använder sådana för att t.ex. beskriva en matematisk funktion som är uppbyggd av flera elementära funktioner.

# 2 Kommentarer och förklaringar

I avsnittet "En enkel beräkning och några grafer" bildade vi några vektorer för att rita en funktionsgraf. Man kan bilda en vektor på tre sätt och vi tar vektorn $\mathbf{c} = (0, 2, 4, 6, 8)$ som exempel. Detta är en samling av talen $0, 2, 4, 6$ samt $8$ och vi låter $\mathbf{c}$ var dess namn. Med kommandot

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

kan vi bilda vektorn i MATLAB. Vi räknar helt enkelt upp talen mellan hakparanteser (`[ ]`) och låter `c` få detta värde. Eftersom vi har samma avstånd (steg) mellan talen kan vi även bilda vektorn med

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

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. Det tredje sättet att bilda en vektor är att använda den inbyggda funktionen `linspace`. Den fungerar ungefär som kolon-notationen 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)
```

För vårt exempel $\mathbf{c} = (0, 2, 4, 6, 8)$ skulle detta bli

```
>> c=linspace(0,8,5)
```

Här måste vi alltså ange hur många element vi skall ha. Kanske inte så praktiskt för vårt exempel. Detta är dock det effektivaste sättet vid grafritning då vi har ett intervall vi är intressede av och vill bara ha tillräckligt många punkter i intervallet för att kunna rita en jämn och snygg graf.

I avsnittet "Skriptfiler" såg vi hur vi kunde samla ihop eller packetera flera sammanhörande kommandon för att lösa en uppgift, och i avsnittet "Lite programmering" såg vi hur vi kunde automatisera en upprepning med en `for`-sats, en s.k. kontrollstruktur. Kontrollstrukturer ser vi

mer på i nästa laboration och script kommer vi använda varje laboration. Det är nog förståndigt att flera gånger återvända till avsnittet om script, bl.a. se på hur man använder cell-läge.

Även avsnittet "Funktioner" handlar på sätt och vis om packetering, men på ett lite annorlunda sätt. I exempel 5 beskrev vi funktionen $f(x) = x^3 - \cos(x)$ som vi skulle söka nollställe till. Så här såg det ut

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

Vi använde två elementära funktioner för att bygga upp vår funktion, som vi sedan gav namnet `min_fun`. Självklart kunde vi givit funktionen ett annat namn, vi bestämmer själva namnet bara det är ett tillåtet namn (samma regler som för variabelnamn, se sid. 2).

Vi skall åter vi se på varför vi beskrev $x^3$ som `x.^3` i MATLAB. Detta kallas för en komponentvis operation och används för vi vill utföra operationen på en hel vektor med $x$-värden på en gång.

Om vi har två vektorer $\mathbf{u} = (2, 3, 5)$ och $\mathbf{v} = (1, 2, 3)$ av samma storlek och vill bilda summan $\mathbf{a} = \mathbf{u} + \mathbf{v}$ och skillnaden $\mathbf{b} = \mathbf{u} - \mathbf{v}$, så gör vi det i MATLAB med `a=u+v` respektive `b=u-v`. Operationerna sker komponentvis (elementvis)

$$\mathbf{a} = \mathbf{u} + \mathbf{v} = (2, 3, 5) + (1, 2, 3) = (2 + 1, 3 + 2, 5 + 3) = (3, 5, 8)$$
$$\mathbf{b} = \mathbf{u} - \mathbf{v} = (2, 3, 5) - (1, 2, 3) = (2 - 1, 3 - 2, 5 - 3) = (1, 1, 2)$$

eller med andra ord $a_i = u_i + v_i$ och $b_i = u_i - v_i$.

Bl.a. vid grafritning behövs de komponentvisa motsvarigheterna till multiplikation, division och upphöjt till, t.ex. kvadrering

$$\mathbf{u}\ \texttt{.*}\ \mathbf{v} = (2, 3, 5)\ \texttt{.*}\ (1, 2, 3) = (2 \cdot 1, 3 \cdot 2, 5 \cdot 3) = (2, 6, 15)$$
$$\mathbf{u}\ \texttt{./}\ \mathbf{v} = (2, 3, 5)\ \texttt{./}\ (1, 2, 3) = (2/1, 3/2, 5/3) = (2, 1.5, 1.666\ldots)$$
$$\mathbf{u}\ \texttt{.\^{}}\ 2 = (2, 3, 5)\ \texttt{.\^{}}\ 2 = (2^2, 3^2, 5^2) = (4, 9, 25)$$

Här har vi lånat beteckningar från MATLAB där vi skriver `u.*v`, `u./v` och `u.^2` för att utföra beräkningarna.

Avslutningvis så handlar avsnitten "Desktop layout" och "Hjälpsystemet" om sådant vi behöver i det fortsatta arbetet. Hur man hittar kommandon eller funktioner för olika uppgifter i samband med problemlösning och hur man kan göra om utseendet på Desktop i MATLAB för att kunna arbeta på ett man trivs med.

# 3 Lärandemål

Efter denna laboration skall du i MATLAB

- kunna tilldela en variabel ett värde och utföra en enklare beräkning samt veta vad som är ett tillåtet variabelnamn
- kunna rita en graf av en funktion och förse den med text på axlar och rubrik samt eventuellt lägga på rutnät med `plot`, `xlabel`, `ylabel`, `title` och `grid`
- kunna bygga upp en vektor med `[ ]`, `:` (kolon-notation) och `linspace`
- kunna skapa och använda script för att samla ihop en följd av kommandon
- ha kännedom om hur man söker hjälptexter och letar efter kommandon och funktioner