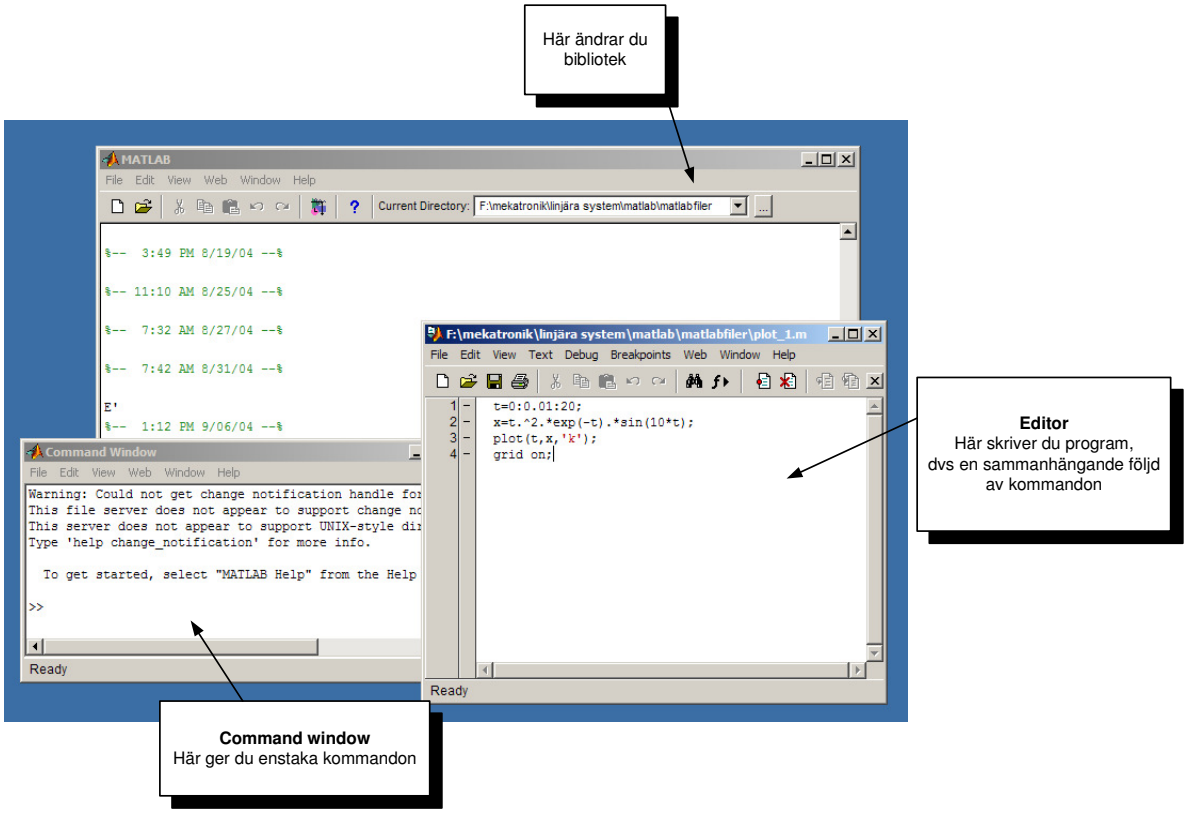
Något om Matlab

Bill Karlström
September 2015

Innehåll

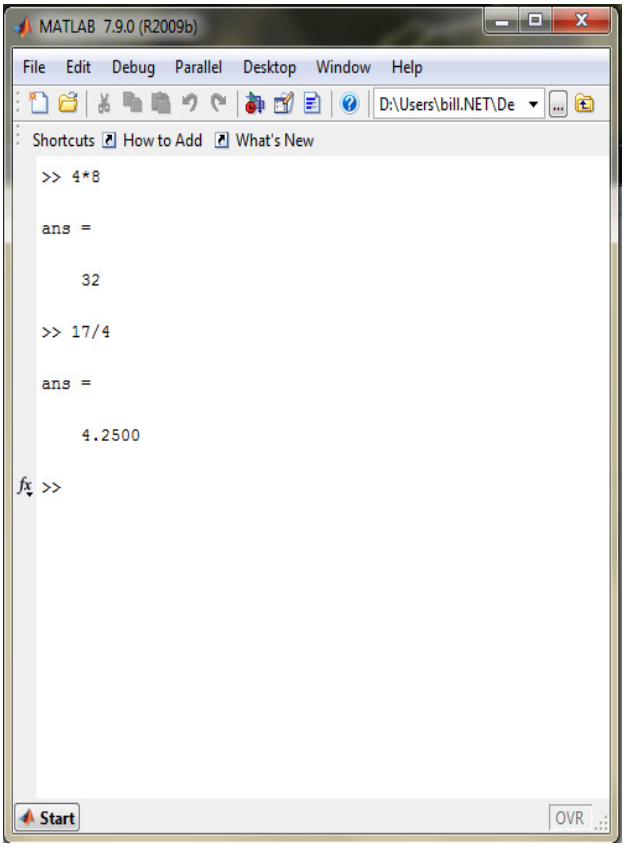
Något om Matlab

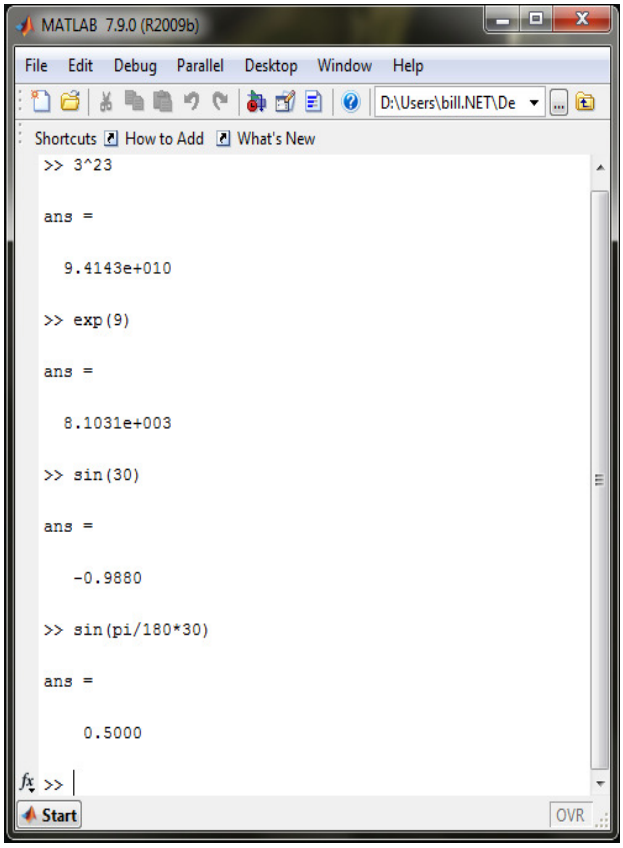
Matlab är ett interaktivt programpaket för numerisk beräkning, simulering och presentation av data. Matlab är en förkortning av MATrix LABoratory. Detta namn antyder att Matlab arbetar med matriser och vektorer, vilket framgår senare.
När du öppnar Matlab ser du ett antal fönster på bildskärmen. Nedan visas de som vi kommer att använda.
MATLAB-fönstret använder vi för att ställa in det bibliotek i vilket vi kommer att lägga våra program. I **Command window** ger vi kommandon, t.ex. för att köra ett visst program. Här får vi också felmeddelanden vid programkörning.



Räknesätt och funktioner

I kommandofönstret kan du göra beräkningar med de fyra räknesätten och matematiska funktioner.





Kommandona ges genom att skriva dem vid promptern och därefter trycka på RETURN.

I Matlab används följande räkneoperationer

+	**addition**
-	**subtraktion**
*	**multiplikation med konstant**
/	**division med konstant**
^	**upphöjt till**
.*	**elementvis multiplikation mellan vektorer**
./	**elementvis division mellan vektorer**
.^	**elementvis ”upphöjt till” av vektor.**

Elementvisa operationer som nämns ovan kommer vi till något senare.

De matematiska funktionerna har följande utseende (ett litet urval av de vanligaste)

$\sqrt{t}$	`sqrt(t)`	$\lvert t\rvert$	`abs(t)`	$\sin t$	`sin(t)`
e^{at}	`exp(a*t)`	$\lvert z\rvert$	`abs(z)`	$\cos t$	`cos(t)`
$\ln t$	`log(t)`	$\arg(z)$	`angle(z)`	$\tan t$	`tan(t)`
$^{10}\log t$	`log10(t)`	$\mathrm{Re}\, z$	`real(z)`		
$\arctan t$	`atan(t)`	$\mathrm{Im}\, z$	`imag(z)`		
$\arcsin t$	`asin(t)`	$\overline{z}$	`conj(z)`		
$\arccos t$	`acos(t)`				

Vektorer och matriser

Exempel

Om vi vill bilda den komponentvisa produkten mellan vektorerna A=[2 3 4 5 6] och B=[1 2 3 4 5] ger vi följande kommandon där semikolon betyder return och >> är promptern i kommandofönstret.
Här separeras vektorkomponenterna med **mellanslag!!**

```
>> A=[2 3 4 5 6];
>> B=[1 2 3 4 5];
>> C=A.*B;
>> C;

C =

    2    6   12   20   30

>>
```

Detta betyder att vektorn $C = [2 \quad 6 \quad 12 \quad 20 \quad 30]$

C=A.*B betyder komponentvis multiplikation

Exempel

Antag att vi vill lösa ekvationssystemet $\begin{cases} 2x_1 + 3x_2 - 5x_3 &= 6 \\ 8x_1 - x_2 - 4x_3 &= -2 \\ x_1 + 7x_2 - 3x_3 &= 1 \end{cases}$,

Då tänker vi oss detta på matrisform enligt $\begin{pmatrix} 2 & 3 & -5 \\ 8 & -1 & -4 \\ 1 & 7 & -3 \end{pmatrix} \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} 6 \\ -2 \\ 1 \end{pmatrix}$ eller $A \cdot x = b$

A, *x* och *b* är s.k. matriser. Dessa är talscheman för vilka räkneregler kan formuleras

Vi går inte in på dessa utan visar bara hur lösningen till ekvationssystemet kan formuleras, nämligen som

$x = A^{-1} \cdot b$,

Där A^{-1} är den ***inversa*** matrisen till *A*.
I matlab löser vi ekvationssystemet i kommandofönstret enligt följande.

Först matar vi in matriserna enligt

```
>> A=[2 3 -5;8 -1 -4;1 7 -3]; %Mellanslag mellan elementen ger 3x3-matris!
b=[6;-2;1];                   %Semikolon mellan elementen ger kolonnvektor, 1x3-
                              %matris!
```

```
x=A\b;                          % Här beräknas lösningen
```

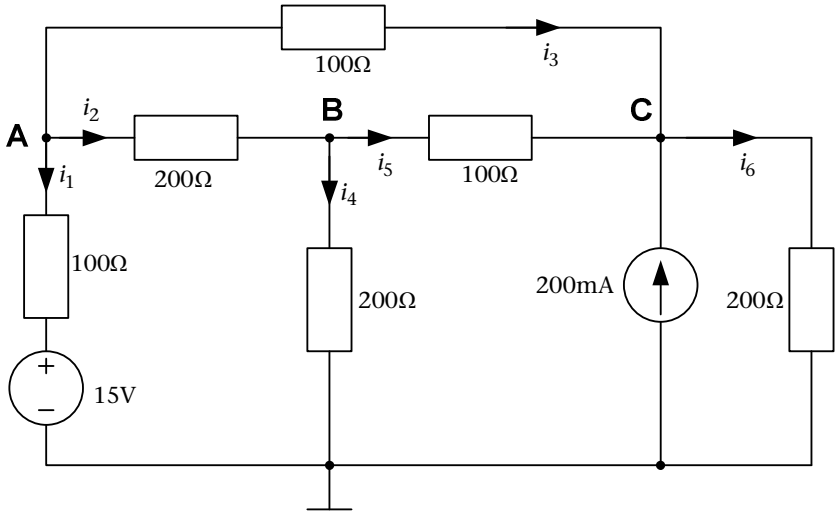
Därefter skriver vi *x* (som vi ju söker) och trycker vi på RETURN och får följande utskrift

```
>> x

x =

            -1.3558
            -0.5521
            -2.0736

>>
```

Detta betyder $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1{,}3558 \\ -0{,}5521 \\ -2{,}0736 \end{pmatrix}$

Exempel

Bestäm nodpotentialerna v_A, v_B och v_C i nedanstående krets.



Lösning

KCL i noderna ger

A $\quad i_1 + i_2 + i_3 = 0$

B $\quad -i_2 + i_4 + i_5 = 0 \quad \Rightarrow$

C $\quad -i_3 - i_5 + i_6 = 0{,}2\text{A}$

Uttryckt i nodpotentialerna blir detta

$$\frac{v_A - 15\text{V}}{100\Omega} + \frac{v_A - v_B}{200\Omega} + \frac{v_A - v_C}{100\Omega} = 0$$
$$-\frac{v_A - v_B}{200\Omega} + \frac{v_B}{200\Omega} + \frac{v_B - v_C}{100\Omega} = 0 \quad \Rightarrow$$
$$-\frac{v_A - v_C}{100\Omega} - \frac{v_B - v_C}{100\Omega} + \frac{v_C}{200\Omega} = 0{,}2\text{A}$$

$$\begin{cases} 5v_A - v_B - 2v_C = 30\text{V} \\ -v_A + 4v_B - 2v_C = 0 \\ -2v_A - 2v_B + 5v_C = 40\text{V} \end{cases} \quad \Rightarrow \quad A \cdot V = E$$

$$A = \begin{bmatrix} 5 & -1 & -2 \\ -1 & 4 & -2 \\ -2 & -2 & 5 \end{bmatrix} \qquad V = \begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} \qquad E = \begin{bmatrix} 30 \\ 0 \\ 40 \end{bmatrix}, \text{ så att}$$

$$\begin{bmatrix} 5 & -1 & -2 \\ -1 & 4 & -2 \\ -2 & -2 & 5 \end{bmatrix} \cdot \begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = \begin{bmatrix} 30 \\ 0 \\ 40 \end{bmatrix}$$

A, *V* och *E* är s.k. matriser. Dessa är talscheman för vilka räkneregler kan formuleras

Vi går inte in på dessa utan visar bara hur lösningen till ekvationssystemet kan formuleras, nämligen som

$V = A^{-1} \cdot E$,

Där A^{-1} är den ***inversa*** matrisen till A.
I matlab löser vi ekvationssystemet i kommandofönstret enligt följande.

Först matar vi in matriserna enligt

```
>> A=[5 -1 -2;
-1 4 -2;
-2 -2 5]

RETURN

A =

     5    -1    -2
    -1     4    -2
    -2    -2     5

>> E=[30;0;40]

RETURN

E =

    30
     0
    40

>> V=A\E

RETURN

V =

   17.2549
   14.7059
   20.7843

>>
```

Detta ger alltså att

$$\underline{\underline{v_A = 17{,}3\text{V}}}$$

$$\underline{\underline{v_B = 14{,}7\text{V}}}$$

$$\underline{\underline{v_C = 20{,}8\text{V}}}$$

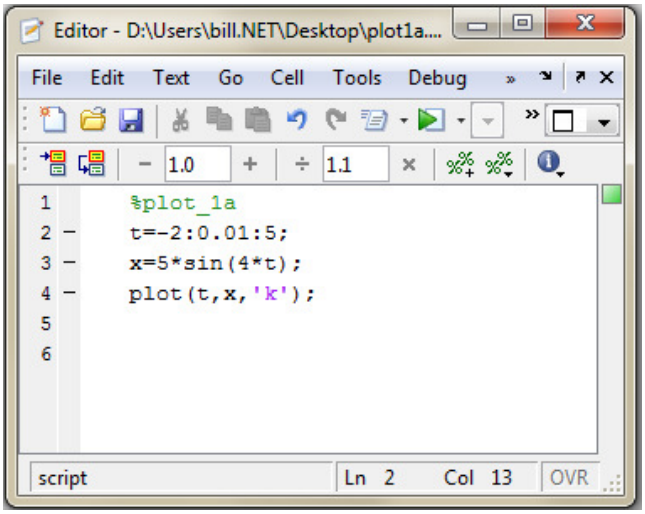
Program

Ofta vill man utföra ett antal kommandon flera gånger. Då skriver man detta som ett antal instruktioner i ett **program**.
Ett datorprogram är en rad instruktioner som datorn skall utföra. De sparas i en fil som kan exekveras genom att man ger datorn ett kommando.
Programmet skrivs i en **editor**.
I Matlab kallas programfilerna m-files.
Editorn öppnas genom att ge kommandot **`edit`** i kommandofönstret. Den kan även öppnas genom att man trycker på den vita rutan högst upp till höger i kommandofönstret eller via
`File> New> Blank M-File`.
Tidigare sparade program öppnas med öppna-symbolen (gul) i editorns menyrad.

Exempel

Vi skall skriva ett program som ritar grafen för $x(t) = 5 \cdot \sin(t)$ i intervallet $-2 \le t \le 5$.
Detta program innehåller ett antal vanliga instruktioner.

```
%p1 plot_1a
t=-2:0.01:5;
x=5*sin(4*t);
plot(t,x,'k');
```



På rad 1 anges programmets namn **`plot1a`**. Denna instruktion skall inte utföras av datorn utan är en kommentar för användaren. Sådana kommentarer föregås av ett procenttecken, %, och får grön text.

På rad 2 anges en vektor med den oberoende variabeln *t*. Här har vi valt att rita kurvan för värden med intervallet 0,01 mellan -2 och 5.

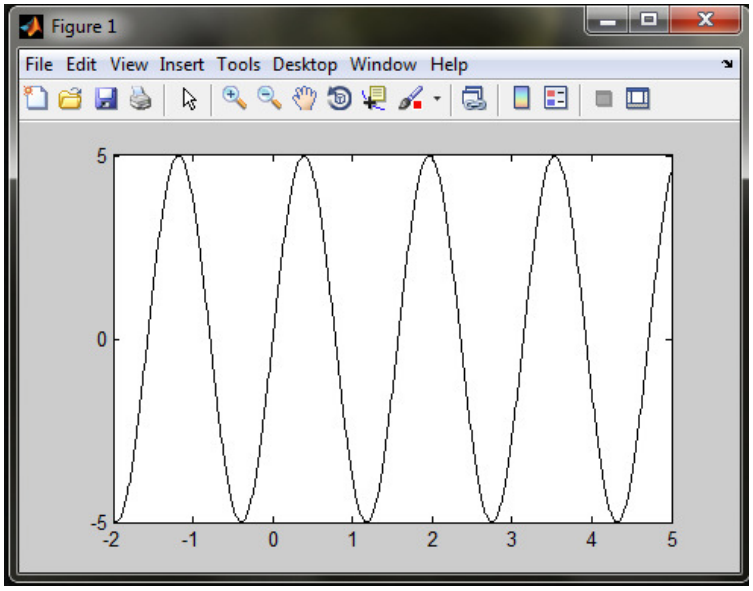
Vektorn anges alltså på formen [***Begynnelsevärde*** : ***Steg*** : ***Slutvärde***].

På rad 3 skrivs den funktion in, vars graf vi vill rita. Vi kallar funktionen *x*.

På rad 4 ges den instruktion som innebär att datorn skall rita grafen. Plot-instruktionen har ett antal argument. Det första är den vektor som anger den oberoende variabeln. Det andra är den vektor som består av funktionsvärdet i de punkter vi valt på rad 1. Det tredje argumentet är 'k', och anger att vi vill rita kurvan svart. Om vi inte anger detta ritas kurvan blå.

Spara nu programmet i en katalog som vi anger enligt ovan (**`Current Folder`**).

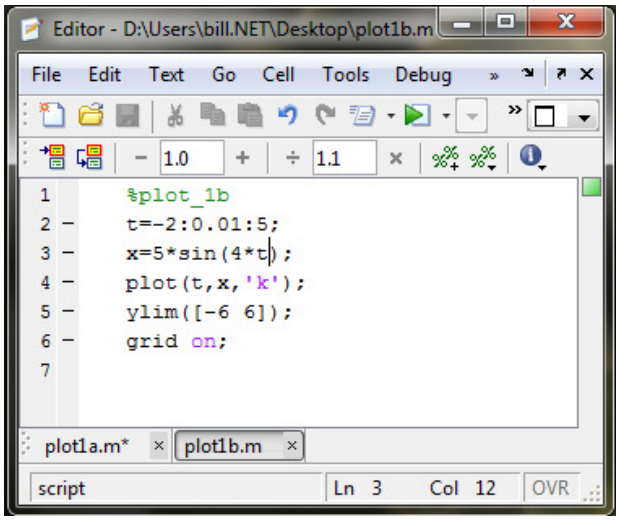
Programmet kan nu köras genom att trycka på den gröna pilen i editorns menyrad.
Detta ger följande utskrift.



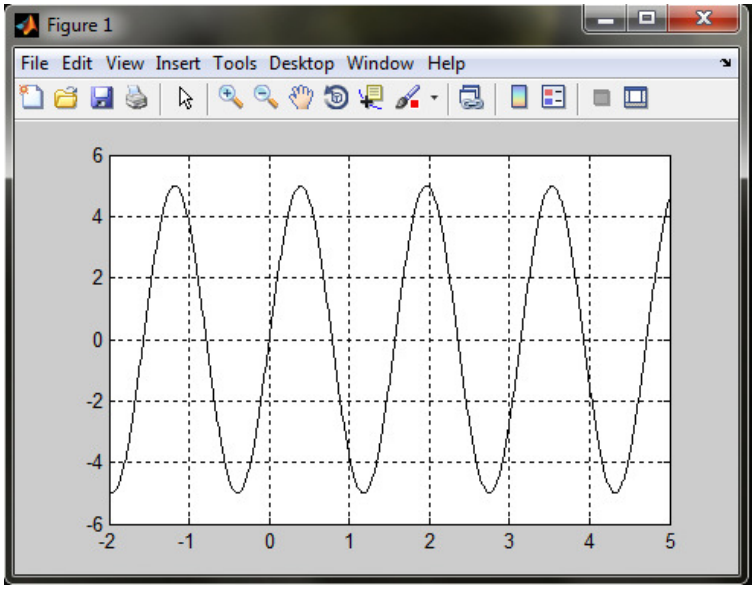
Som synes den sinuskurva vi önskade. Utskriften kan förbättras genom att lägga in ett koordinatsystem och låta den vertikala axeln löpa mellan -6 och 6.

Programmet kompletteras enligt följande

```
%p2 plot_1b
t=-2:0.01:5;
x=5*sin(4*t);
plot(t,x,'k');
ylim([-6 6]);
grid on;
```



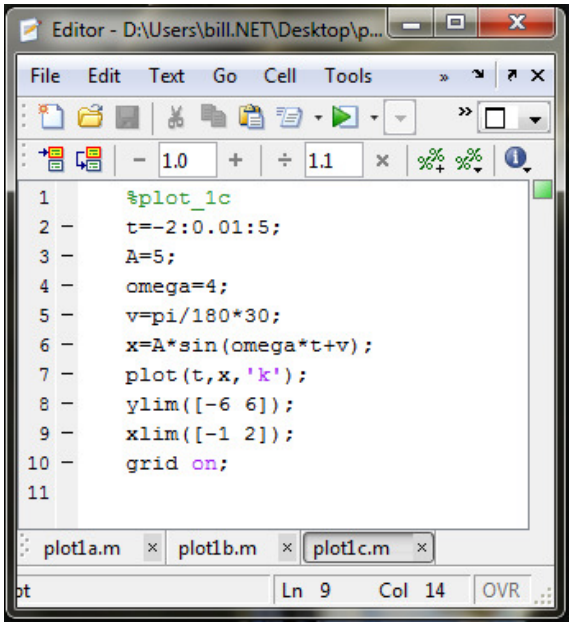
Rad 5 anger att vi vill att den vertikala axeln skall löpa mellan -6 och 6
Rad 6 anger att vi vill ha ett rutmönster som bildar ett koordinatsystem.
Detta program ger följande utskrift



Exempel

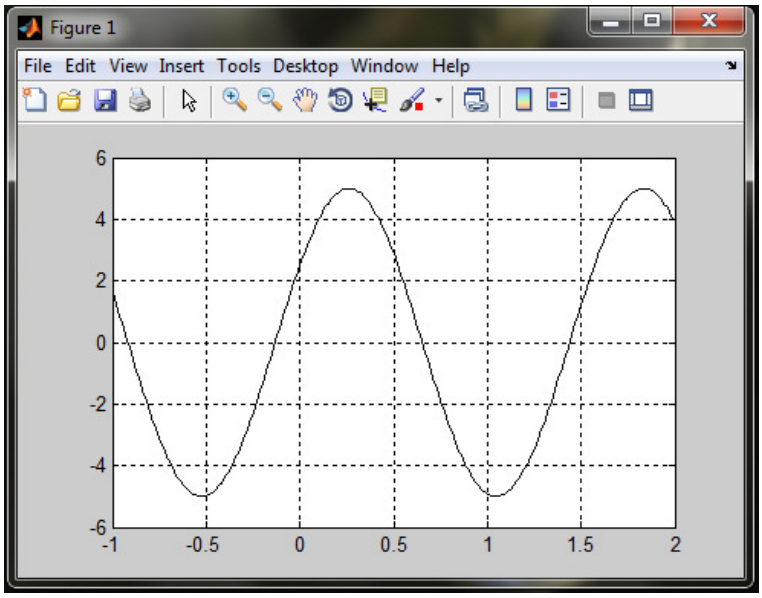
Om vi vill rita grafen för $x(t) = 4 \cdot \sin(5t + 30^\circ)$ måste vi tänka på att matlab vill ha vinklar angivna i radianer.

```
%p3 plot_1c
t=-2:0.01:5;
A=5;
omega=4;
v=pi/180*30;
x=A*sin(omega*t+v);
plot(t,x,'k');
ylim([-6 6]);
xlim([-1 2]);
grid on;
```



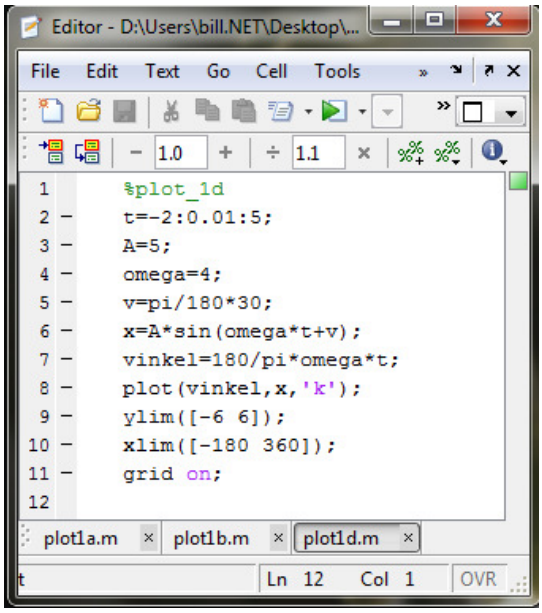
Här har vi angivit amplitud, vinkelfrekvens och fasvinkel separat. Dessutom vill vi studera kurvan i ett mindre tidsintervall vilket anges med instruktionen **`xlim`**.

Utskriften blir enligt följande



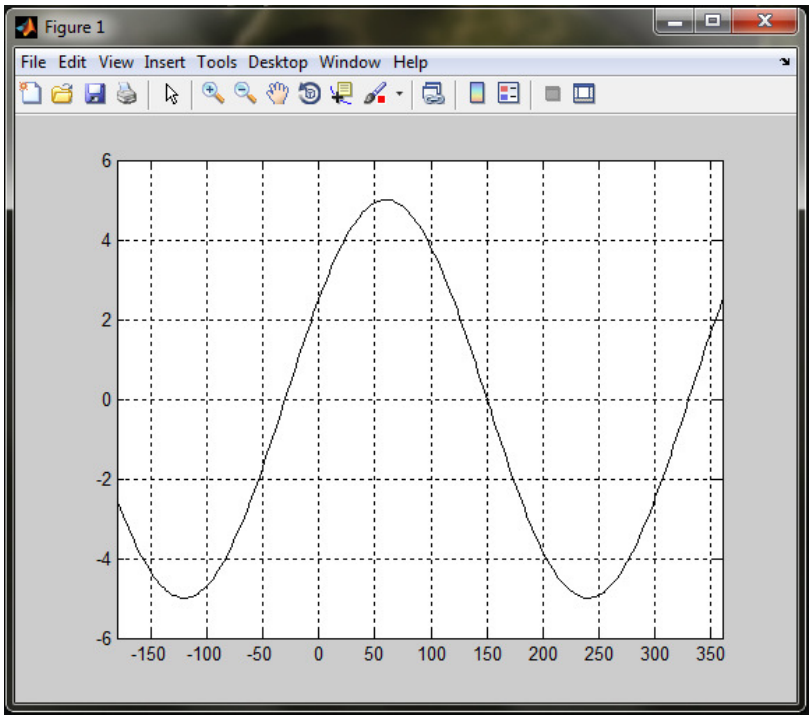
Om vi nu vill ha den oberoende variabeln i grader måste vi göra följande ändring

```
%p4 plot_1d
t=-2:0.01:5;
A=5;
omega=4;
v=pi/180*30;
x=A*sin(omega*t+v);
vinkel=180/pi*omega*t;
plot(vinkel,x,'k');
ylim([-6 6]);
xlim([-180 360]);
grid on;
```

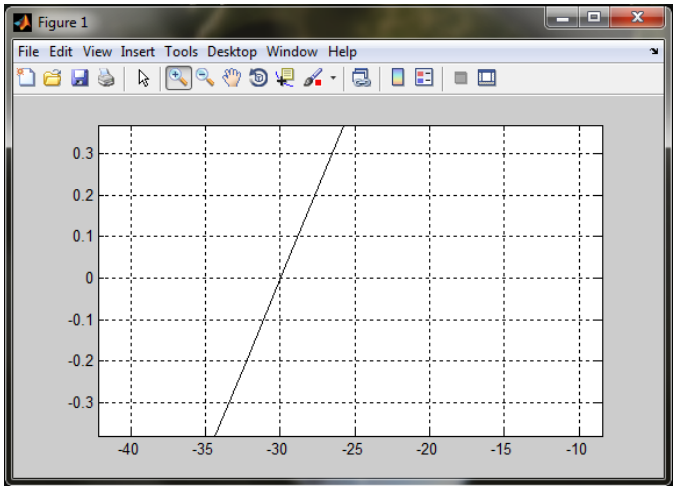


På rad 7 gör vi en ny oberoende variabel genom att räkna om tiden till grader med hjälp av vinkelfrekvensen och omvandla med faktorn $\frac{180^\circ}{\pi}$.

Vi har också ändrat begränsningen av den nya oberoende variabeln på rad 10.
Detta ger



Om vi förstorar med symbolen i menyraden får vi

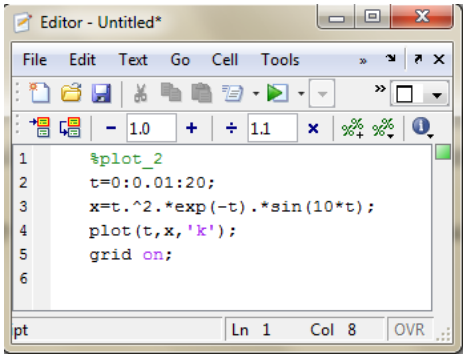


Vi ser att kurvan är förskjuten 30° åt vänster.

Exempel

Rita grafen för $x(t) = t^2 e^{-t} \sin 10t$
Vi gör det med följande program

```
%p5 plot_2a
t=0:0.01:20;
x=t.^2.*exp(-t).*sin(10*t);
plot(t,x,'k');
grid on;
```



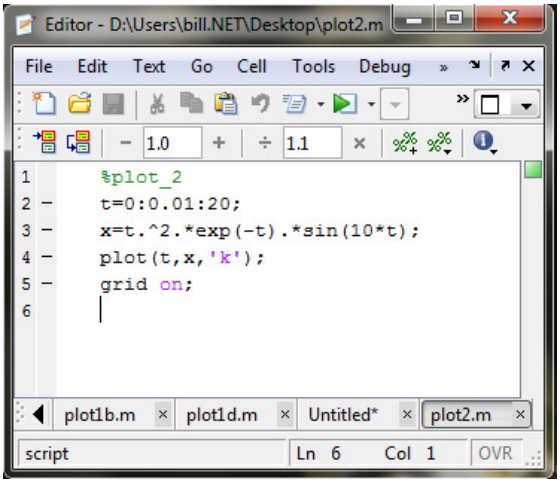
På rad 2 ges den oberoende variabeln i form av vektorn
[0 0.01 0.02 ……20].
Antalet element i denna vektor är alltså 2001.
På rad 3 beräknas funktionen (vektorn) $x(t)$ i de 2001 punkter som anges i vektorn t ,där några skrivsätt kräver sin förklaring.

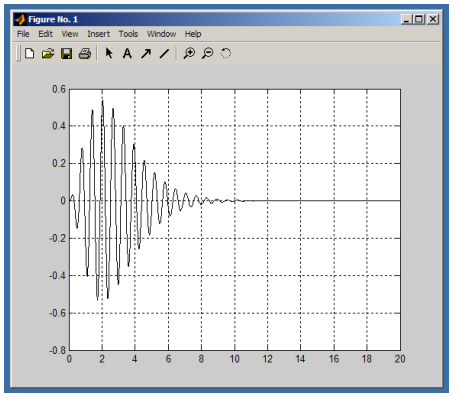
Beteckningen **.*** betyder alltså att vektorerna ***t*.^2**, **exp(-*t*)**, och **sin(10*t*)** multipliceras ***elementvis*** vilket kräver att de har lika många element!

Observera då att ***t*.^2** betyder att datorn beräknar t^2 för varje element i tidsvektorn.

På rad 4 ges ett skrivkommando `plot(t,x,'k')`.
Detta är alltså ett kommando att rita en kurva med punkterna i vektorn t på den horisontella axeln och punkterna i vektorn $x(t)$ på den vertikala axeln.
Sist i parentesen står en instruktion att rita kurvan i svart färg. Här kan man även ge andra instruktioner om hur kurva skall ritas.
På femte raden ges instruktionen `grid on` vilket betyder att vi vill ha stödlinjer i diagrammet.

Programmet sparas och körs genom att under menyn **Debug** ge kommandot **Run**
Om programmet körs direkt efter det att det skrivits heter kommandot **Save and run**!
Det kan också köras från kommandofönstret. Detta görs genom att anropa programmet med dess namn och sedan trycka på RETURN.
Utskriften visas t.h. nedan.

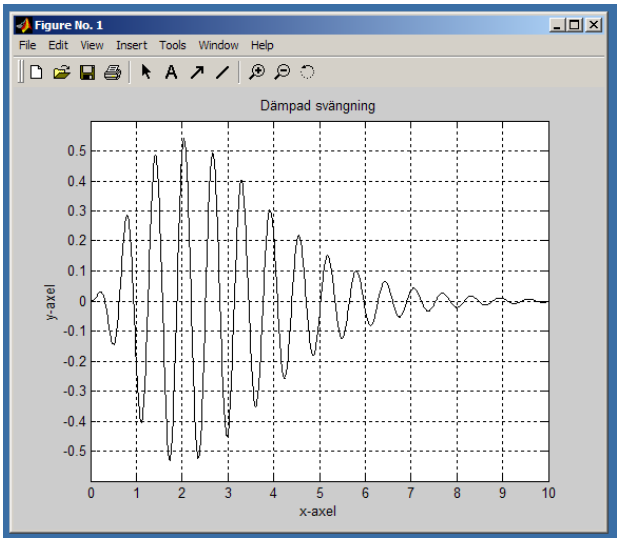




Som synes utspelar sig det intressantaste i intervallet [0,10]. Vi kan, som ovan, i plot-kommandot begränsa utskriften till detta intervall.

```
%p6 plot_2b
t=0:0.01:20;
x=t.^2.*exp(-t).*sin(10*t);
plot(t,x,'k');xlim([0 10]);ylim([-0.6
0.6]);
grid on;
title('Dämpad svängning');
xlabel('x-axel');
ylabel('y-axel');
```

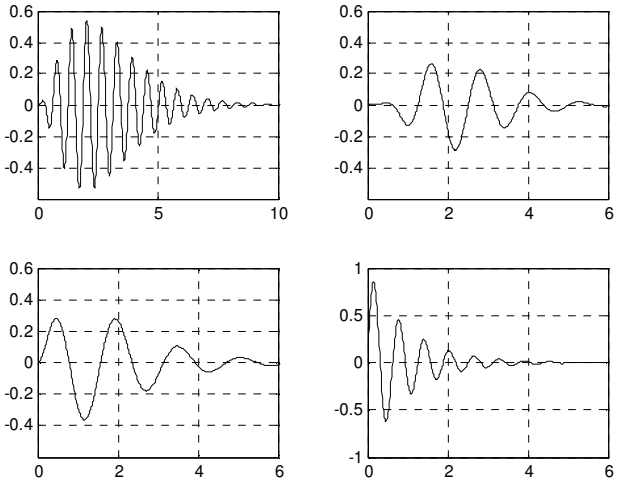
I detta program finns även tre rader som anger kurvans namn samt beteckningen på axlarna.
I bilderna på utskrifterna ovan har hela bildskärmsutsendet tagits med. I fortsättningen visas endast själva kurvan!



I nästa exempel visas utskriftskommandon för det fall där flera kurvor skall skrivas ut i olika diagram samtidigt.
Den centrala instruktionen här är `subplot(m,n,p)` som delar upp grafikfömstret i $m \times n$ delar och skriver ut i den del som anges av p.

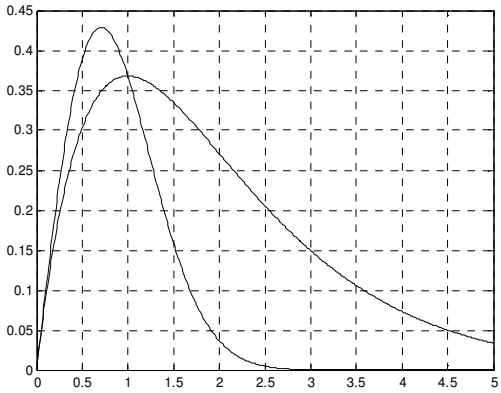
```
%p7 plot3
t=0:0.01:20;
x=t.^2.*exp(-t).*sin(10*t);
subplot(2,2,1)
plot(t,x,'k');xlim([0 10]);ylim([-0.6 0.6]);
grid on;
x=t.^4.*exp(-2*t).*sin(5*t);
subplot(2,2,2)
plot(t,x,'k');xlim([0 6]);ylim([-0.6 0.6]);
grid on;
x=t.*exp(-t).*sin(4*t);
subplot(2,2,3)
plot(t,x,'k');xlim([0 6]);ylim([-0.6 0.6]);
grid on
grid on;x=exp(-t).*sin(10*t);
subplot(2,2,4)
plot(t,x,'k');xlim([0 6]);ylim([-1 1]);
grid on;
```

Se figur nedan.



Ibland vill man ha olika kurvor utritade i samma diagram. Nedanstående program visar ett sådant fall.

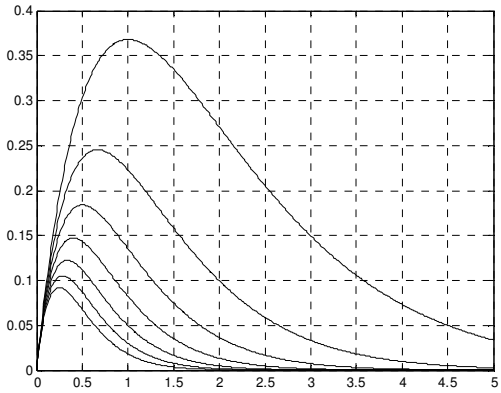
```
%p8 plot_4
t=0:0.01:5;hold off
x1=t.*exp(-t);
plot(t,x1,'k');
hold on
x2=t.*exp(-t.^2);
plot(t,x2,'k');
grid on;
```



Här ritas först kurvan $x1 = t \cdot e^{-t}$. Därefter ges instruktionen `hold on` som betyder att grafikfönstret skall hållas öppet så att även kurvan $x2 = t \cdot e^{-t^2}$ kan ritas i diagrammet.
På första raden måste instruktionen `hold off` ges så att grafikfönstret inte är öppet när programmet körs ytterligare en gång.
Detta ger diagrammet t.h.

Om man vill rita många kurvor kan man lägga in ritinstruktionen i en for-loop som styrs av t.ex. en parameter i funktionsuttrycket.

```
%p9 plot5
t=0:0.01:5;hold off
for k=1:0.5:4
    x=t.*exp(-k*t);
    plot(t,x,'k');
    hold on
    grid on;
end
```



De instruktioner som står mellan `for` och `end` utförs för
k = 1 1,5 2 2,5 3 3,5 4.

Instruktionen `hold on` gör att programmet ritar en kurva för varje k-värde i samma diagram.
Detta ger diagrammet t.h.
Instruktionen `hold off` gör att grafikfönstret stängs så att tidigare figurer inte kommer med.

Funktionsytor

Här presenteras kommandon med vilka vi kan rita funktionsytor $z = f(x,y)$

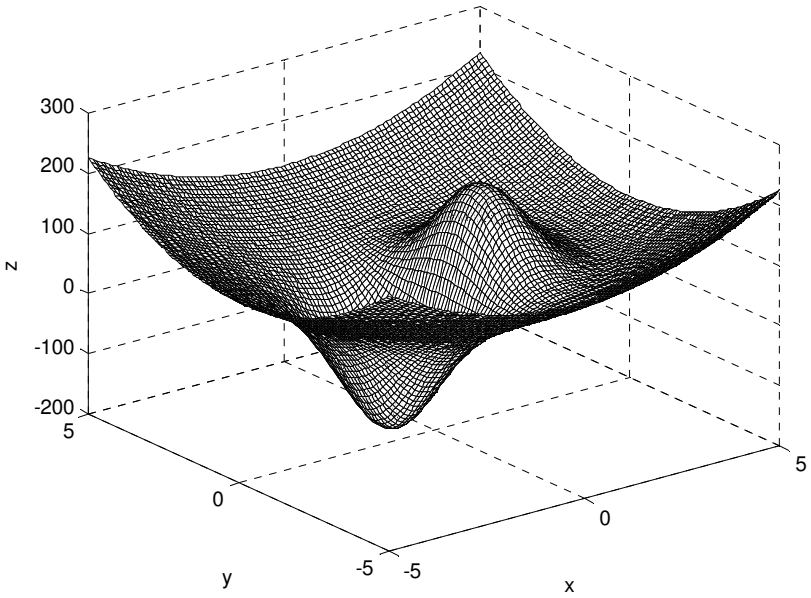
Exempel

Rita ytan $z = 4x^2 + 5y^2 + 250e^{-\left(x^2+y^2\right)}$ för $-5 \le x \le 5 \quad -5 \le y \le 5$

I programmet genereras först en matris som innehåller de x- och y-värden för vilka funktionsvärdena skall beräknas.
Detta sker med kommandot `[X,Y]=meshgrid(x,y)`
Sedan beräknas funktionsvärdena i dessa punkter med kommandot
`Z=4*X.^2+5*Y.^2+250*X.*exp(-0.35*(X.^2+Y.^2))`
Därefter ritas ytan med `mesh(X,Y,Z)`.

```
%p10 plot6 Program som ritar funktionsyta
x=-5:0.1:5  %Anger x-värden
y=-5:0.1:5  %Anger y-värden
[X,Y]=meshgrid(x,y) %Genererar gridmatris
Z=4*X.^2+5*Y.^2+250*X.*exp(-0.35*(X.^2+Y.^2))  %Beräknar funktionsvärden
mesh(X,Y,Z) %Ritar ytan
xlabel('x'), ylabel('y'), zlabel('z')
```

Detta ger följande utskrift.



En variant på `mesh(X,Y,Z)` är `contour(X,Y,Z,N)` som ger N nivåkurvor i xy-planet.
T.h. visar en utskrift med 10 nivåkurvor.