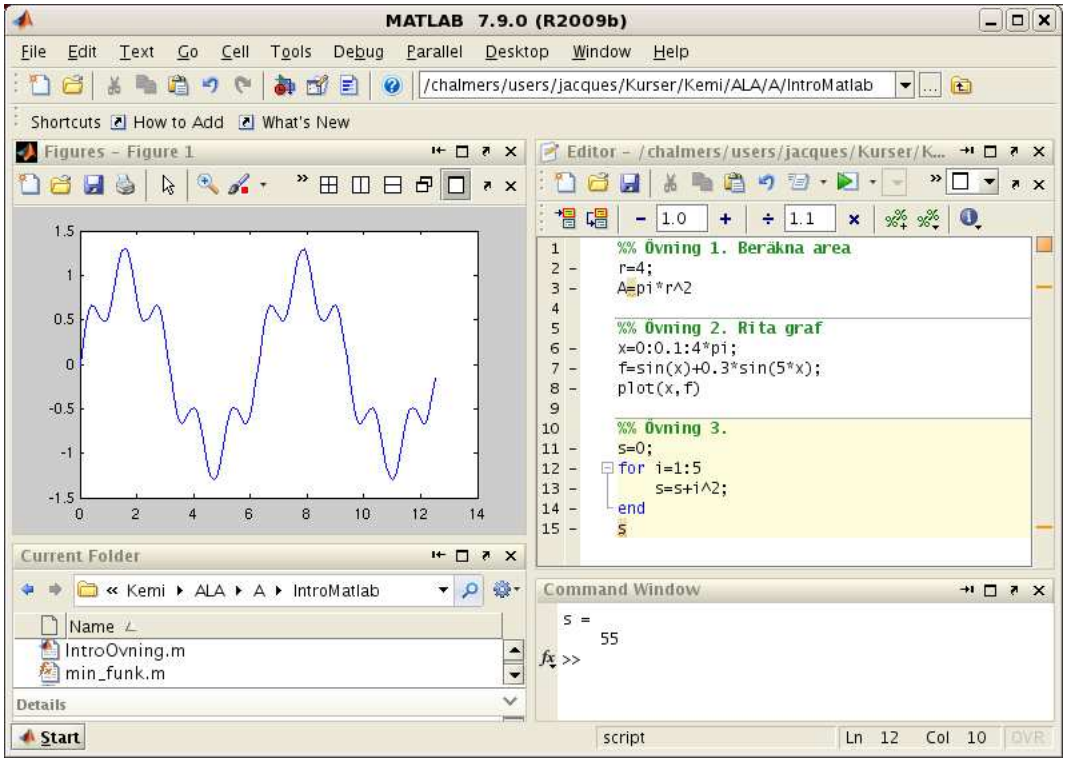
Matriser och Inbyggda funktioner i MATLAB

1 Inledning

Vi skall denna vecka se på matriser och funktioner som är inbyggda i MATLAB, dels (elementära) matematiska funktioner som sinus och cosinus, dels funktioner för att bilda och operera på vektorer och matriser. Vi skall också ta fram den desktop layout som vi skall använda i de kommande studio-övningarna.

Men allra först: För att arbeta effektivt och dokumentera arbetet kommer ni använda er av M-File (skriftfil) som vi såg på förra studio-övningen.

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 en hel studio-övning) i flera delar (varje deluppgift).

Bilden ovan visar uppgifterna från förra studio-övningen med editorn i cell-läge. 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 under Cell i verktygsfältet.

En cell kan evalueras utan att textfilen är sparad, så gör Save då och då.

2 Matriser

Grundstenen i linjär algebra, som ni läser i läsperiod 3, är matrisbegreppet. Detta är även den grundläggande datatypen i MATLAB. Namnet MATLAB står i själva verket för ”Matrix Laboratory”.

En matris är ett rektangulärt talschema

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

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)

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix}$$

Element nr i ges i MATLAB av `a(i)` och antalet element ges av `length(a)`. På motsvarande sätt för radvektorn **b**.

Som exempel tar vi

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

Vi skriver in detta i MATLAB enligt

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

Element på samma rad separeras med ett blanktecken eller ett komma, medan element på olika rader separeras med semikolon.

Uppgift 1. Skriv in matriserna i MATLAB och skriv sedan ut matriselementen a_{23}, b_{42}, a_2. Prova `size` och `length`. Ändra b_{23} genom att skriva `B(2,3)=5`.

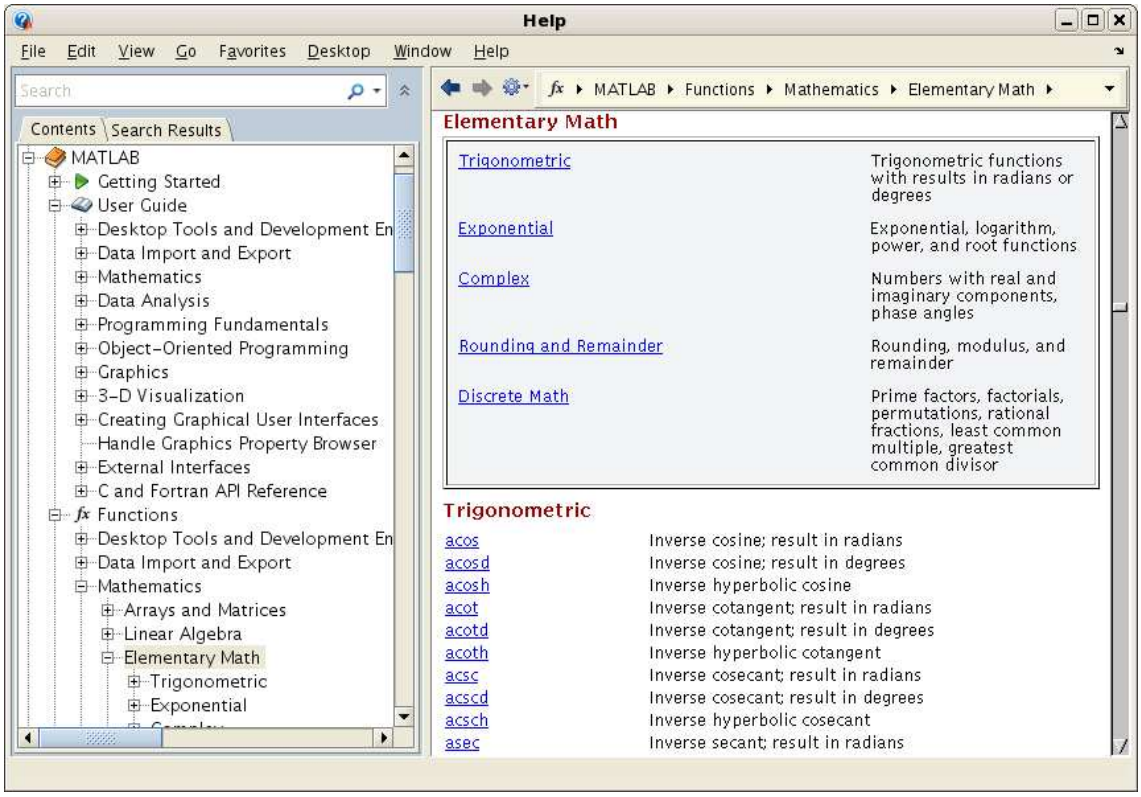
Ibland vill vi se en tabell som en matris. Som exempel tar vi: Värmeförlusten hos den som vistas i kyla beror inte enbart på temperaturen, utan även på hur mycket det blåser. Tabellen visar vilken **effektiv temperatur** det blir vid olika temperaturer T (°C) och vindhastigheter v (m/s).

v \ T	10	6	0	-6	-10	-16	-26	-30	-36
2	9	5	-2	-9	-14	-21	-33	-37	-44
6	7	2	-5	-13	-18	-26	-38	-44	-51
10	6	1	-7	-15	-20	-28	-41	-47	-55
14	6	0	-8	-16	-22	-30	-44	-49	-57
18	5	-1	-9	-17	-23	-31	-45	-51	-59

Om vi ville göra något med dessa data i MATLAB så skulle vi lagra temperaturer och vindhastigheter i rad- eller kolonnvektorer och effektiva temperaturerna i en matris. (Vill du läsa mer om värmeförlust gå till SMHI:s hemsida och sök på vindavkylning.)

3 Inbyggda funktioner

Vi letar upp hjäptexterna för elementära eller matematiska funktioner i Help Navigator genom att succesivt öppna MATLAB, User Guide, Functions, Mathematics och slutligen Elementary Math.



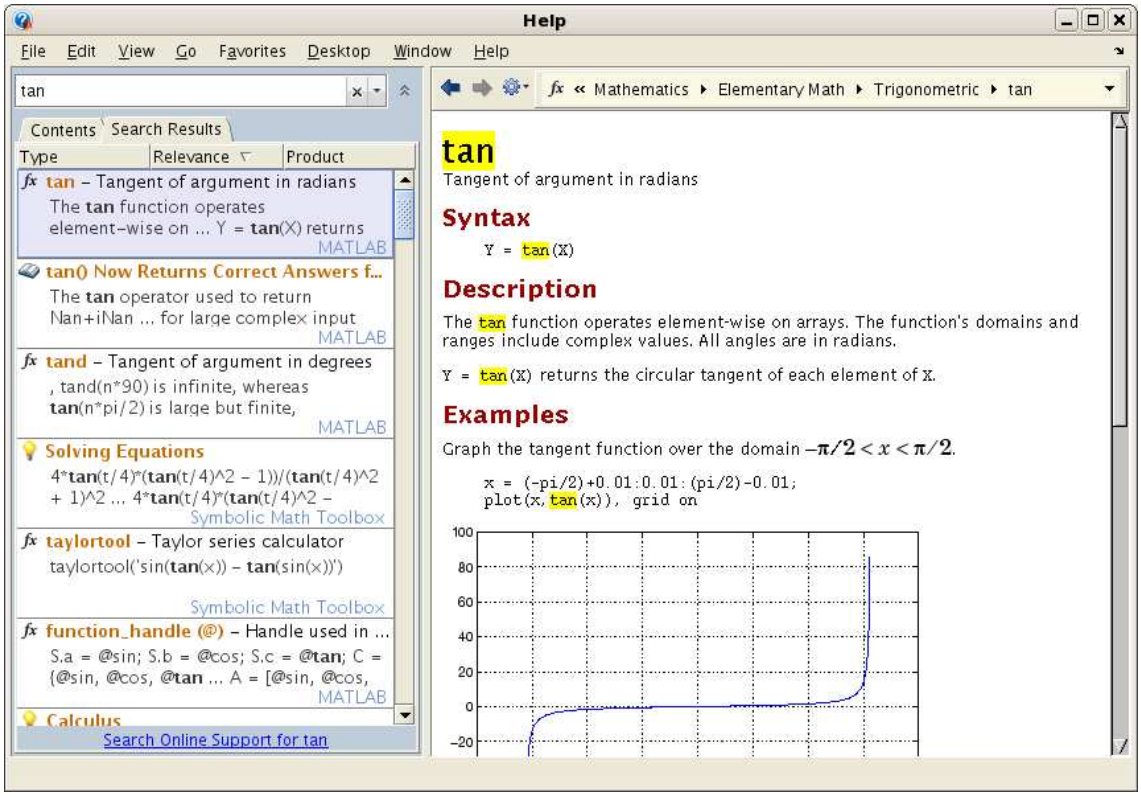
Vi har en gruppering i trigonometriska, exponential, komplexa, avrundning och rester samt diskreta funktioner.

Funktioner som sinus och cosinus, kan operera på matriser. Man får som resultat en matris av samma typ vars element är funktionsvärdet av respektive element i argumentet.

```
>> v=sin(b)
v =
         0    0.9093   -0.7568   -0.2794    0.9894

>> V=sin(A)
V =
    0.8415   -0.7568    0.6570   -0.5440
    0.9093   -0.9589    0.9894   -1.0000
    0.1411   -0.2794    0.4121   -0.5366
```

Vi söker på `tan`, dvs. tangensfunktionen, och ser på hjälptexten.



Uppgift 2. Leta upp hjälptexten du ser i figuren och rita upp tangensfunktionen enligt exemplet. Bygg vidare på skriptfilen ni nyss gjorde med en ny cell för denna uppgift.

Vi har redan sett de inbyggda funktionerna `length` och `size`. Antal element i vektorn **b** ges av

```
>> l=length(b)
l =
    5
```

och antalet rader och kolonner **A** fås med

```
>> [m,n]=size(A)
m =
     3
n =
     4
```

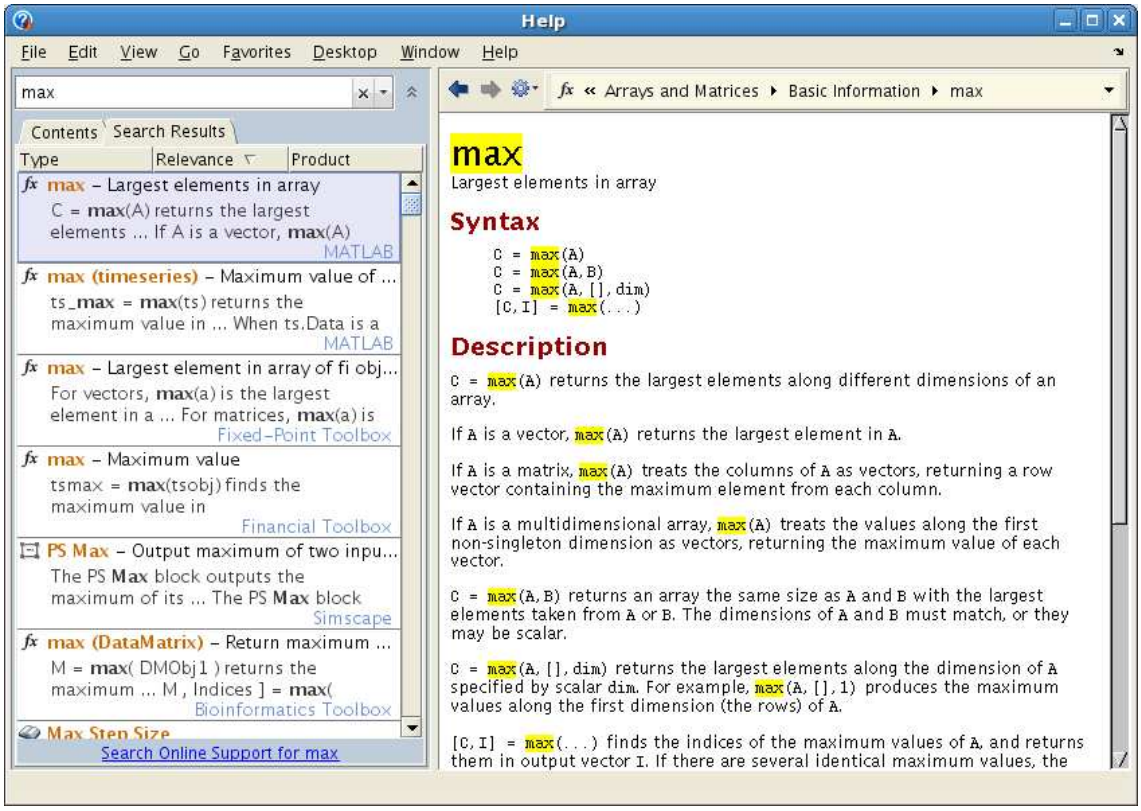
Kontrollera genom att skriva `b` respektive `A` och räkna. Pröva med **a** och **B** också!

Största och minsta elementet i en vektor fås med funktionerna `max` och `min`. För en matris blir det de största elementen i varje kolonn.

```
>> v=max(b)
v =
    8


>> v=max(A)
v =
     3     6     9    12
```

Vi ser på hjälptexten för `max`



Vi ser att vi med `[C,I]=max(A)` även kan få reda på var det maximala värdet finns någonstans.

Summan och produkten av elementen i vektorn fås med `sum` och `prod`. För en matris blir det summan eller produkten av varje kolonn.

```
>> s=sum(b)
s =
    20

>> s=sum(A)
s =
     6     15     24     33
```

Vill vi sortera en vektor i stigande ordning gör vi det med `sort`. För en matris blir det varje kolonn som sorteras om i stigande ordning.

Med funktionen `linspace` kan vi göra en radvektor med ett antal värden jämnt fördelade mellan två gränser och är speciellt användbar då man ritar grafer. T.ex. med `x=linspace(0,5,20)` får vi en radvektor med 20 värden mellan 0 och 5. Med `x=linspace(0,5)` får man istället 100 värden.

Funktionerna `zeros` och `ones` används för att bilda matriser med nollor respektive ettor. Med `zeros(2,5)` får vi en matris av typen 2×5 fylld med nollor och med `zeros(size(A))` får vi en matris fylld med nollor av samma typ som `A`. Motsvarande gäller för `ones` fast då blir det ettor istället.

4 Matrisoperationer

Vi fortsätter och ha följande matriser och vektorer som exempel

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

Redan i första studio-övningen bildade vi radvektorer eller radmatriser med en operation som kallas **kolon-notation** och som används flitigt då vi arbetar med matriser och vektorer.

Vi gjorde t.ex. en radvektor `x` av x-värden mellan 0 och 4π, med `x=0:0.1:4*pi`.

Den allmänna formen för att bilda en radvektor med kolon-notation är

```
variabel=start:steg:slut
```

och motsvarande för att bilda en kolonnvektor är

```
variabel=(start:steg:slut)'
```

Radvektorn **b** kan vi alltså bilda på två sätt

```
>> b=[0 2 4 6 8]
>> b=0:2:8
```

och kolonnvektorn **a** kan vi bilda med

```
>> a=[1;3;5]
>> a=(1:2:5)'
```

Apostrofen (') är en matrisoperation som vänder på en matris, rader blir kolonner och kolonner blir rader.

Vi låter s få tredje värdet b_3 med `s=b(3)` och bildar vektorn **v** av andra och femte värdet, dvs. $\mathbf{v} = (b_2, b_5)$, med

```
>> v=b([2,5])
v =
     2     8
```

Vi kan ändra ett element i **v**, t.ex. låta $v_2 = 0$, med

```
>>  v(2)=0
v =
     2     0
```

Vi låter s få värdet av elementet på rad 2, kolonn 3 i matrisen **A** från inledningen med `s=A(2,3)` och vi bildar en radvektor **v** av rad 3, alla kolonner med

```
>> v=A(3,:)
v =
    3     6     9    12
```

samt en kolonnvektor **u** av rad 2-3, kolonn 2 med

```
>> u=A(2:3,2)
u =
    5
    6
```

Vi bildar en matris **V** av blocket rad 1-2, kolonn 2-3

```
>> V=A(1:2,2:3)
V =
    4     7
    5     8
```

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

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

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

T.ex. vid grafritning behövs de komponentvisa motsvarigheterna till multiplikation och division

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

Här har vi lånat beteckningar från MATLAB där vi skriver `u.*v` respektive `u./v` för att utföra beräkningarna.

Vi behöver även komponentvis upphöjt till, t.ex. kvadrering

$$\mathbf{u} \texttt{ .\^{} } 2 = (2, 3, 5) \texttt{ .\^{} } 2 = (2^2, 3^2, 5^2) = (4, 9, 25)$$

Även här har vi lånat beteckningen från MATLAB.

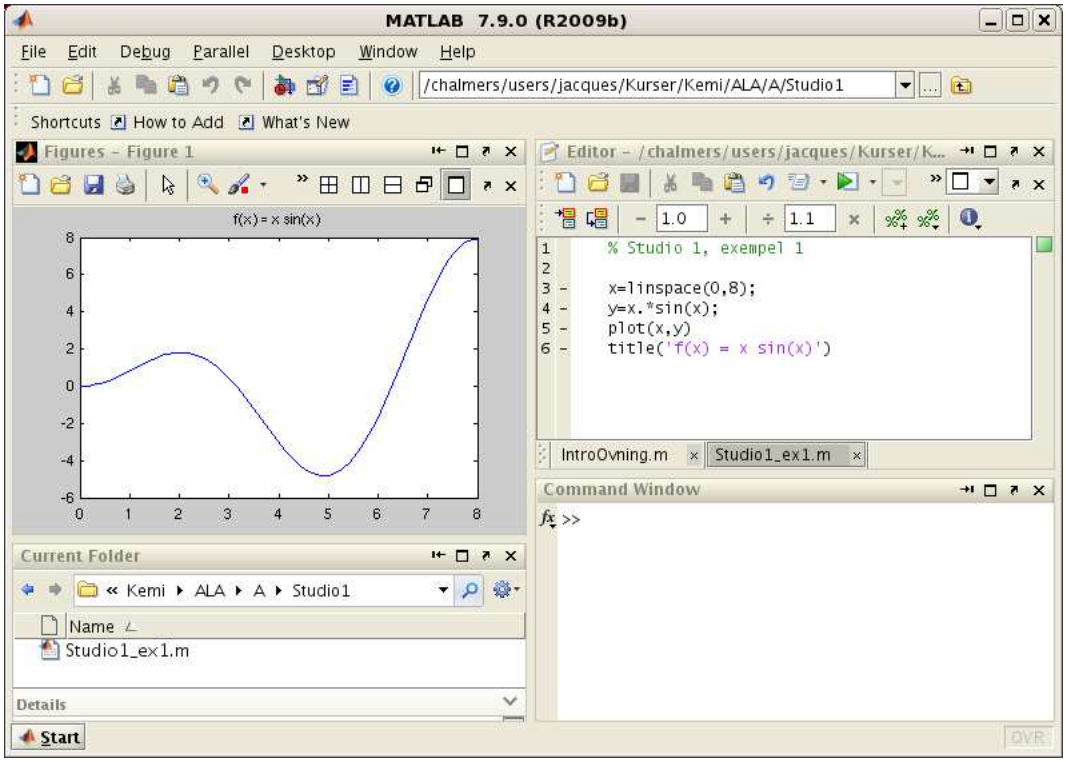
Exempel 1. Rita grafen till $f(x) = x\sin(x)$ över intervallet $0 \leq x \leq 8$.

Vi bildar en vektor $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ med värden jämnt fördelande över intervallet $0 \leq x \leq 8$. Sedan bildar vi vektorn $\mathbf{y} = (f(x_1), f(x_2), \cdots, f(x_n)) = (x_1\sin(x_1), x_2\sin(x_2), \cdots, x_n\sin(x_n))$ och ritar upp grafen. För att bilda vektorn **y** behövs den komponentvisa multiplikationen.

Vi ritar grafen med

```
>> x=linspace(0,8);
>> y=x.*sin(x);
>> plot(x,y)
>> title('f(x) = x sin(x)')
```

och så här ser resultatet ut



5 Desktop Layout

För att handledning och redovisning skall fungera effektivt kräver vi att all redovisning görs via en sammanhållande skriptfil tillsammans med nödvändiga funktionsfiler (från en skriptfil anropas t.ex. funktionsfiler som behövs för att lösa uppgiften). Skriptfilen som används för redovisning bör vara lämpligt uppdelad med hjälp av t.ex. kommandot `pause` eller alternativt vara i cell-läge (Cell Mode).

Vi käver också att ni har en MATLAB desktop layout av ett slag som visas i figuren ovan. Man åstadkommer detta genom att ”docka” in MATLAB-editorn respektive figurfönstret 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”).

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 3. Gör en desktop layout som ser ut ungefär som den i bilden ovan. Spara denna layout med ett lämpligt namn, vilket görs genom att välja Save Layout ... under Desktop i verktygsfältet. Denna layout skall ju sedan användas under studio-övningarna.