Allmänt om Matlab

- Utvecklades på 70-talet som ett lättanvänt gränssnitt till programbiblioteken LINPACK (linjär algebra) och EISPACK (egenvärdesproblem), ursprungligen skrivna i Fortran.
- En kommersiell version programmerad i C gavs ut år 1984 av företaget Mathworks.
- Bra för numeriska beräkningar.
- I nya versioner finns även stöd för symboliska beräkningar i samarbete med Maple.
- Programmet finns bland annat till Windows och Linux.
- Senaste versionen är 7.10.
- Finns även open source varianter som Octave (www.octave.org) och Scilab (www.scilab.org).

Köra Matlab

På Tuxedo från Windows dator

- Instruktioner finns att laddas ner från kursens hemsida.

Linux

- Linuxdatorn måste vara igång i X-windows läge.
- Logga sedan in på `tuxedo.abo.fi` genom att skriva `ssh -X tuxedo.abo.fi`, starta sedan Matlab med kommandot `matlab`.

Windows

- Finns installerad i t.ex. Axelias PC-klass och Matematiska institutionens datasal.

Grundläggande kommandon

- Kommandon evalueras med <enter>.

```
>> 1+1
ans =
     2
```

- Utskrift av resultatet fås om man ej avslutar med ;.

```
>> 1+1;
```

- Man kan använda vanliga uttryck: `1+2`, `4*5`, `2/4`, `2*(3-5)`.
- Variabler tilldelas värden enligt `a=2+sin(2)`.
- Inbyggda funktioner och konstanter har logiska namn och skrivs alltid med små bokstäver: `sin(pi/2)`, `exp(i*pi)`, `log(2)`, `eps`, `inf`.
- `format long` och `format short` anger hur många siffror som ska skrivas ut.
- Med hjälp av <tab> kompletteras kommandon.

Hjälpkommandon

- Med `help kommando` får man fram en hjälptext om kommandot.
- `doc kommando` visar manualen för kommandot i ett skilt fönster.
- Olika typer av demonstrationer fås via `demo`.
- Genom att skriva `help log` och trycka på `tab` så visar Matlab en lista med kommandon som börjar på `log`.

Speciella symboler

- Vissa symboler har speciella betydelser, här är de vanligaste:

Symbol	Används för att	Exempel
=	ge en variabel ett värde	`a=5*atan(1)`
==	jämförelse (jfr. <, >, <=, >=, ~=)	`1==2`
;	lämna bort outputen, ny rad i matris	`a=5*atan(1);`
:	definiera en uppräkning	`1:5, 1:0.1:5`
,	separera argument eller uttryck	`tan(2), sin(2)`
.	vektor/matrisoperation elementvis	`A.*B`
'	transponera/konjugera en matris/vektor	`A'`
.'	transponera en komplex matris	`A.'`
...	fortsätt inmatningen på nästa rad	`A=[1 2; ... 3 4]`
(och)	ge argument, gruppera uttryck	`sin(pi/2), 5*(2+3)`
[,]	definiera vektorer, matriser	`a=[1 2; 3 4]`
[och]	ange flera returvärden	`[V,D]=eig(A)`
[och]	tömma matrisen	`a=[], clear a`

Vektorer och matriser

- Radvektorer skrivs in som: `a=[1 2 3]` eller `a=[1,2,3]`.
- Kolonnvektorer skrivs in som: `a=[1;2;3]`.
- Matriser skrivs in som `c=[1 2 3; 2 3 4]` eller `c=[1,2,3;2,3,4]`.
- Variabler (även vektorer och matriser) kan även skapas och ändras "grafiskt" i *Workspace*-listan.

Vektorer och matriser, forts.

- ”Ekvidistanta” matriser och vektorer skapas genom:

```
>> a = 1:3
a =
     1     2     3
>> b = 1:0.5:3
b =
    1.0000    1.5000    2.0000    2.5000    3.0000
```

(i det senare fallet är alltså steglängden 0.5).

Vektorer och matriser, forts.

- Vektorkomponenter tas ut genom `a(j)`.

```
>> a(1)
ans =
     1
```

- I matriser är motsvarande `c(j,i)`:

```
>> m=[1:0.5:3 ; -1:0.2:-0.2]
m =
    1.0000    1.5000    2.0000    2.5000    3.0000
   -1.0000   -0.8000   -0.6000   -0.4000   -0.2000

>> m(2,3)
ans =
   -0.6000
```

Funktioner för generering av matriser

- Matlab har kommandon som genererar speciella vektorer och matriser:
  - Enhetsmatriser `eye(n)`.
  - Matriser bestående av ettor `ones(n,m)`.
  - Matriser bestående av nollor `zeros(n,m)`.
  - Matriser bestående av likformigt fördelade och normalfördelade slumptal på intervallet $[0, 1]$ `rand(n,m)` respektive `randn(n,m)`.
  - Vektorer bestående av slumpmässigt permuterade heltal mellan 1 och n `randperm(n)`.

Matrisfunktioner

- Matlab har många inbyggda matrisfunktioner:

Kommando	Beskrivning	Exempel
`'`	transponatet	`A'`
`size`	storleken av matrisen	`[n,m]=size(A)`
`norm`	matrisnorm	`norm(A)`
`rank`	rangen	`rank(A)`
`det`	determinanten	`det(A)`
`lu`	LU-faktorisering	`[L,U]=lu(A)`
`inv`	matrisinvers	`inv(A)`
`qr`	QR-faktorisering	`[Q,R]=qr(A)`
`eig`	egenvektorer och -värden	`[X,D]=eig(A)`
`poly`	karekteristiska polynomet	`poly(A)`
`svd`	SVD-uppdelning	`[S,V,D]=svd(A)`

Lösning av ekvationssystem

- Ekvationssystem av typen $AX = B$ löses med divisionsoperatorn \.
- **Exempel:**

$$\begin{cases} 2x + 3y + z & = 7 \\ 2x - 3y + 2z & = -6 \\ x + \frac{1}{2}y - z & = 3 \end{cases}$$

```
>> A=[2 3 1; 2 -3 2; 1 0.5 -1]; B=[7;-6;3]
```

```
>> X1=A\B
X1 =
    1.0000
    2.0000
   -1.0000
```

```
>> X2=inv(A)*B
X2 =
    1.0000
    2.0000
   -1.0000
```

Polynom

- Polynom i Matlab är vektorer med polynomets koefficienter.
- Värdet av polynom i en punkt ges av `polyval`.
- Polynomet deriveras med `polyder`.

Exempel: $p(x) = x^7 + 3x^6 + 2x^5 + 5x^4 + 5x^2 + 7x + 8$

```
>> p = [1 3 2 5 0 5 7 8];
>> polyval(p,2)
ans =
   506

>> pd = polyder(p)
pd =
     7    18    10    20     0    10     7
```

Polynoms rötter

- Polynomets rötter fås med kommandot `roots`.

Exempel: $p(x) = x^7 + 3x^6 + 2x^5 + 5x^4 + 5x^2 + 7x + 8$

```
>> p = [1 3 2 5 0 5 7 8];
>> pd = polyder(p);
>> roots(pd)
ans =
  -2.4860
  -0.2717 + 1.0963i
  -0.2717 - 1.0963i
   0.4721 + 0.6523i
   0.4721 - 0.6523i
  -0.4863
```

Anpassa polynom till datapunkter

- Med kommandot `polyfit` anpassas polynom till datapunkter.

```
>> x=0:5
x =
     0   1   2   3   4   5

>> y=sin(x)
y =
    0  0.8415  0.9093  0.1411 -0.7568 -0.9589

>> z=polyfit(x,y,5)
z =
   -0.0054  0.0861 -0.3919  0.2672  0.8855  0.0000
```

Funktioner

- Funktioner definieras främst i skilda filer (senare).
- Enkla funktioner kan definieras genom

```
>> f = inline('x^2+sin(x)')
f =
     Inline function:
     f(x) = x^2+sin(x)

>> f(2)
ans =
    4.9093
```

Nollställen till funktioner

- Kan använda kommandot `fzero` för att hitta nollställen till funktioner. Måste också ge ett startvärde.

```
>> f = inline('x^2-2*x-2');

>> fzero(f,0)
ans =
    -0.7321

>> fzero(f,2)
ans =
    2.7321
```

Minimum av funktioner

- Kan använda kommandot `fminbnd` för att hitta minimum av en funktion f i ett intervall.
- Maximum hittas genom att man söker minimum för $-f$.

```
>> f = inline('x^2-2*x-2');

>> [x,val] = fminbnd(f,0,2)
x =
    1.0000
val =
   -3
```

Egna funktioner

- Egna funktioner skapas i egna filer med filändelsen `.m`.
- Genom att välja `New->M-file` i Directory-fältet skapas en ny fil i den aktuella mappen.
- Vettigt att spara .m filer i en egen mapp.
- Syntaxen för filen är:

```
function [ut1,ut2] = funktionsnamn(in1,in2,in3)

    ut1 = 2*in1;
    ut2 = in2+in3;

end
```

Exempel: Arean av en triangel

- Skriv en funktion som beräknar arean av en triangel.
- Skapar en fil `area_triangel.m` med följande innehåll:

```
function [A] = area_triangel(b,h)
 Beräknar arean av en triangel med basen b
 och höjden h.
A = 0.5 * b * h;
end
```

- Körs sedan genom att skriva

```
>> area_triangel(2,3)
ans =
    3.0000
```

Loopar

- Upprepningar kan åstadkommas med följande kommandon
  - `for`

```
for i=1:10
    a(i) = i;
end
```

  - `while`

```
n=1;
while n<=10
    a(n) = n;
    n = n+1;
end
```

Villkor

- Villkorssatser görs med `if - else - elseif`:

```
a = 10*rand();
if a < 5
    disp('Talet är mindre än 5');
elseif a < 7
    disp('Talet är större än eller lika med 5,
          men mindre än 7');
else
    disp('Talet är större än eller lika med 7');
end
```

- Obs! Man kan ha hur många `elseif` satser som helst.

Exempel: Fibonaccis talföljd

```
function fib = fibonacci(n)
 FIBONACCI Funktion som ger ut de
 n första talen i Fibonaccis talföljd

    if n > 1
        F = zeros(1,n);
        F(1) = 1;   F(2) = 1;

        for i = 3:n
            F(i) = F(i-1) + F(i-2);
        end
    else
        F(1) = 1;
    end
    fib = F;
end
```

Rekursiva funktioner

- Skapar en funktion som räknar fakulteten (!) av ett tal genom att skriva följande i en fil `fakultet.m`:

```
function fak = fakultet(n)

    if n==0
        fak = 1;
    else
        fak = n*fakultet(n-1);
    end

end
```

- Observera att detta inte är det mest effektiva sättet!

Allmänt om plottning

- Plotkommandon i Matlab är bl.a. `plot`, `ezplot`, `plot3`, `contour`, `surf`, etc.
- Graferna öppnas i ett skilt fönster; i allmänhet ersätts en tidigare graf med en ny om man ger ett nytt "plot"-kommando.
- Vill man visa flera grafer i samma fönster skriver man först `hold on`, ritar sedan upp graferna, och avslutar med `hold off`.
- Oftast enklast att först rita upp grafen utan att ange hur den ska se ut, och sedan ändra utseendet med det grafiska verktyget.

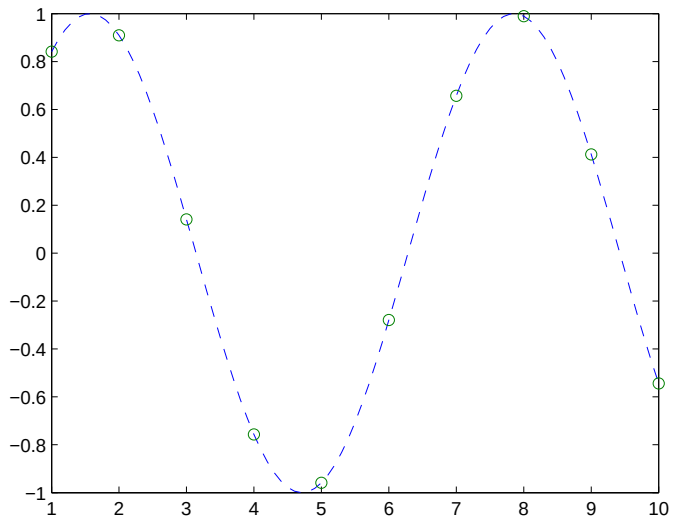
Plottning av 2D-data

- Enkla grafer skapas med `plot`-kommandot. Syntaxen är `plot(x-data, y-data)`. Man ger alltså in två listor med lika längd som motsvarar x- och y-värdena.
- Som standard ritas automatiskt linjer mellan punkterna ut. Vill man endast visa datapunkterna kan man t.ex. skriva `plot(x,y,'.')`.
- Det går även att visa flera funktioner i samma graf genom att ge in flera datalistor enligt `plot(x-data1, y-data1, x-data2, y-data2)`.

Plottning av 2D-data, exempel

- Genom att ge in en extra sträng efter x- och y-data i `plot`-kommandot kan man ge olika utseende åt kurvorna.

```
>> dx = linspace(1,10,100);      dy = sin(dx);
>> x = 1:10;     y = sin(x);
>> plot(dx,dy,'- -',x,y,'o');
```

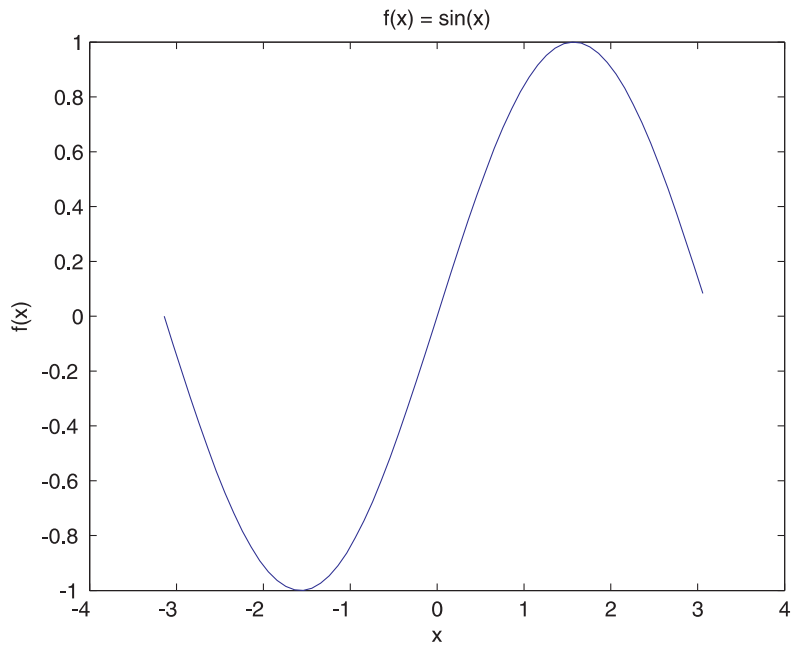


- Med `linspace(min,max,n)` skapas alltså en ekvidistant vektor mellan `min` och `max` med `n` element.

Rubriker i grafer

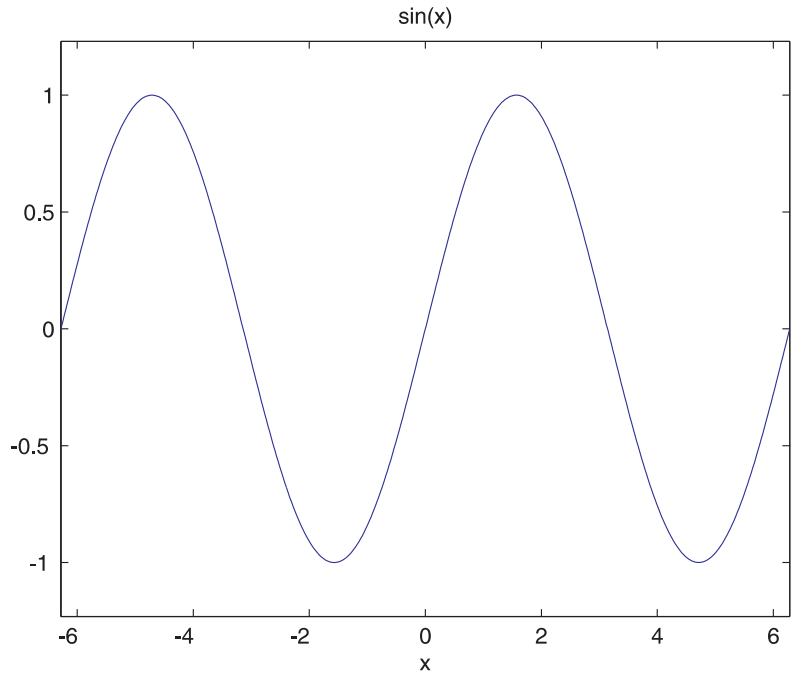
- Rubriker för grafen och axelbenämningar fås med `title('')` respektive `xlabel('')` och `ylabel('')`.

```
>> x = -pi:0.1:pi;
>> plot(x,sin(x));
>> title('f(x) = sin(x)');
>> xlabel('x');     ylabel('f(x)');
```



Plotta funktioner

- Man kan även använda `ezplot`-kommandot för att snabbt rita upp enkla grafer.
- **Exempel:** kommandot `ezplot('sin(x)')` ritar upp grafen av $\sin(x)$ på intervallet $[-2\pi, 2\pi]$.

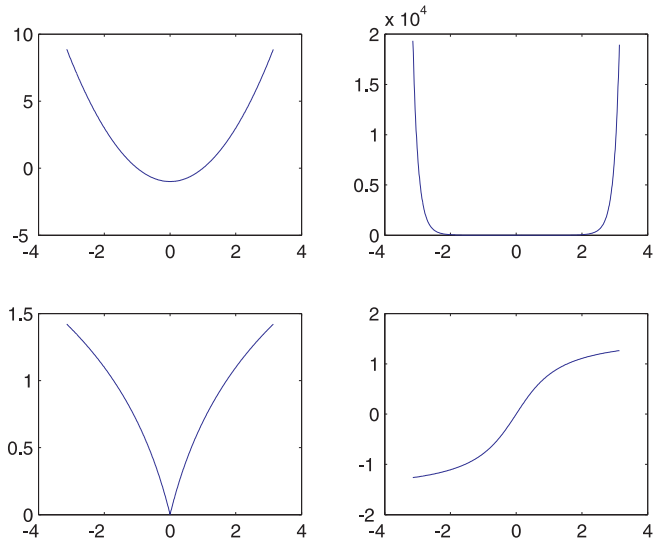


- Annat intervall fås med `ezplot('sin(x)',[xmin,xmax])`.

Flera grafer i samma figur

- Med hjälp av `subplot(m,n,p)`-kommandot placeras grafer i en ”tabell” med m rader och n kolumner.

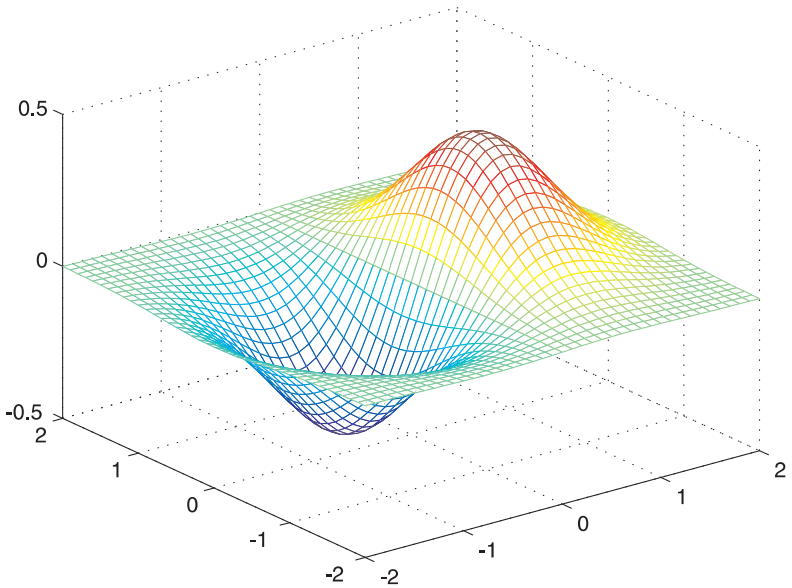
```
>> x = -pi:0.01:pi;
>> y1 = x.^2-1;
>> y2 = exp(x.^2);
>> y3 = log(abs(x)+1);
>> y4 = atan(x);
>> subplot(2,2,1), plot(x,y1);
>> subplot(2,2,2), plot(x,y2);
>> subplot(2,2,3), plot(x,y3);
>> subplot(2,2,4), plot(x,y4);
```

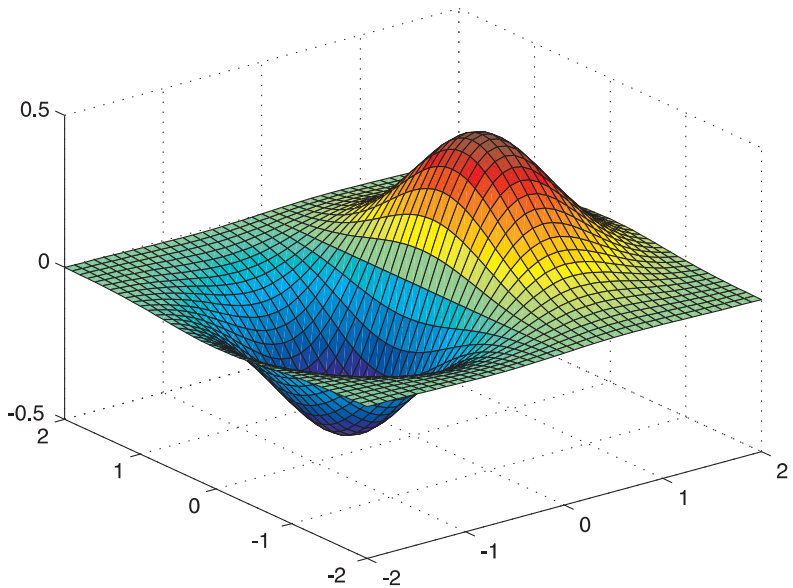


3D-plottning

- En funktion $z = f(x, y)$ plottas enklast genom att man först skapar en s.k. `meshgrid` och sedan använder något av kommandona `mesh`, `contour` eller `surf`.
- **Exempel:** $f(x, y) = x\, e^{-x^2-y^2}$, $x, y \in [-2, 2]$

```
>> [X,Y] = meshgrid(-2:0.1:2, -2:0.1:2);
>> Z = X.*exp(-X.^2 -Y.^2);
>> mesh(X,Y,Z);
>> surf(X,Y,Z);
```

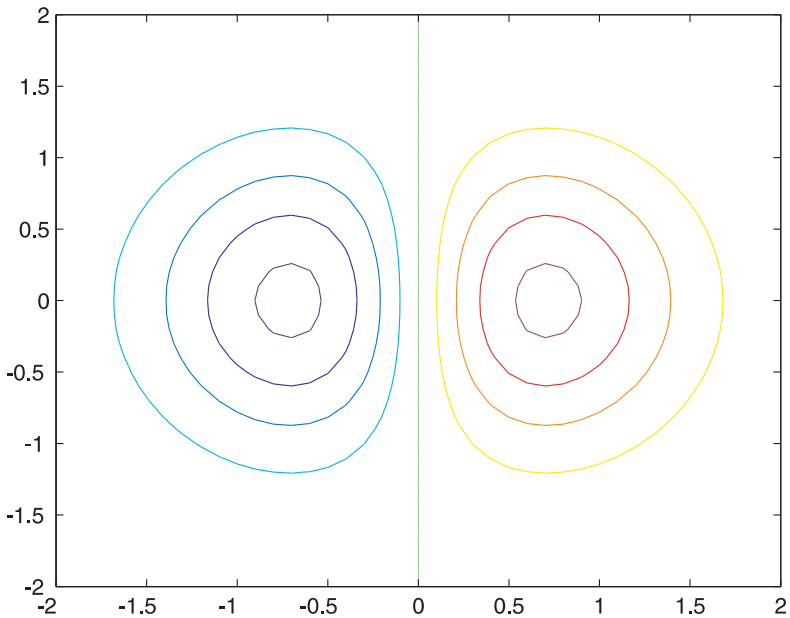


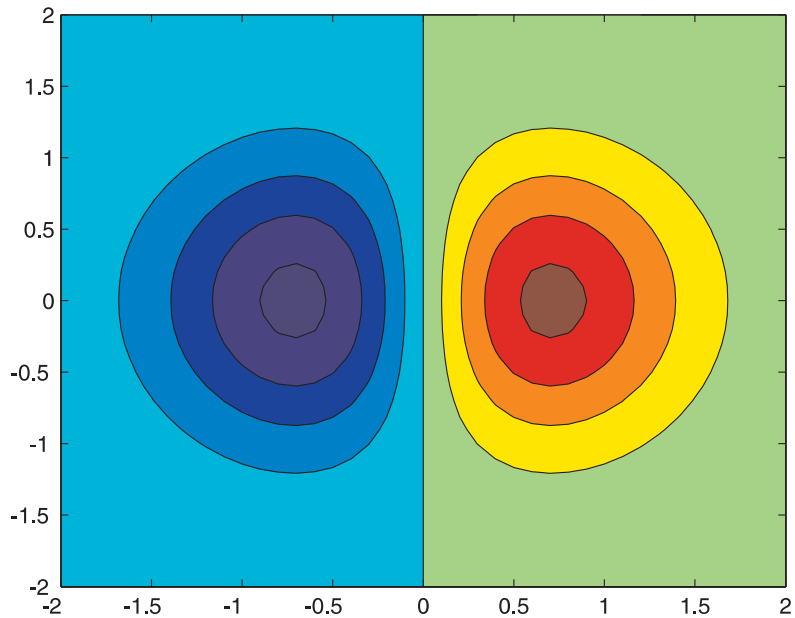


3D-plottning, forts.

- **Exempel:** $f(x, y) = x\ e^{-x^2-y^2}$, $x, y \in [-2, 2]$

```
>> [X,Y] = meshgrid(-2:0.1:2, -2:0.1:2);
>> Z = X.*exp(-X.^2 -Y.^2);
>> contour(X,Y,Z);
>> contourf(X,Y,Z);
```

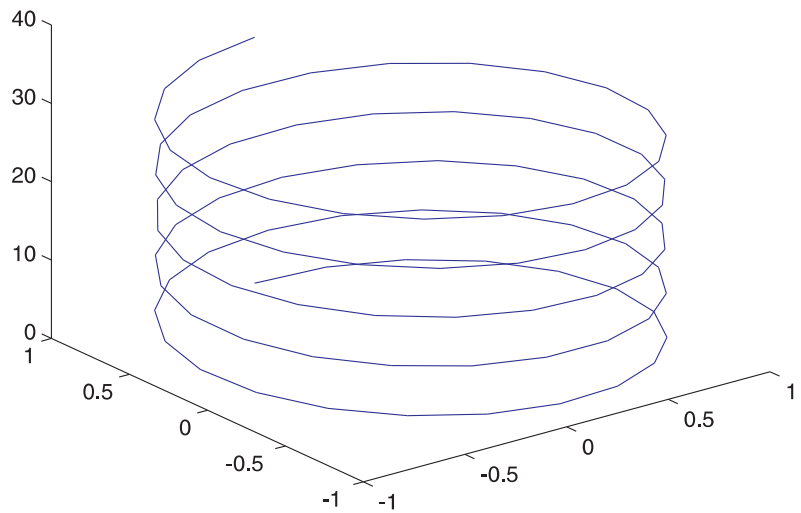




Rymdparameterkurvor

- Parameterkurvor i rymden åstadkomms med `plot3`.

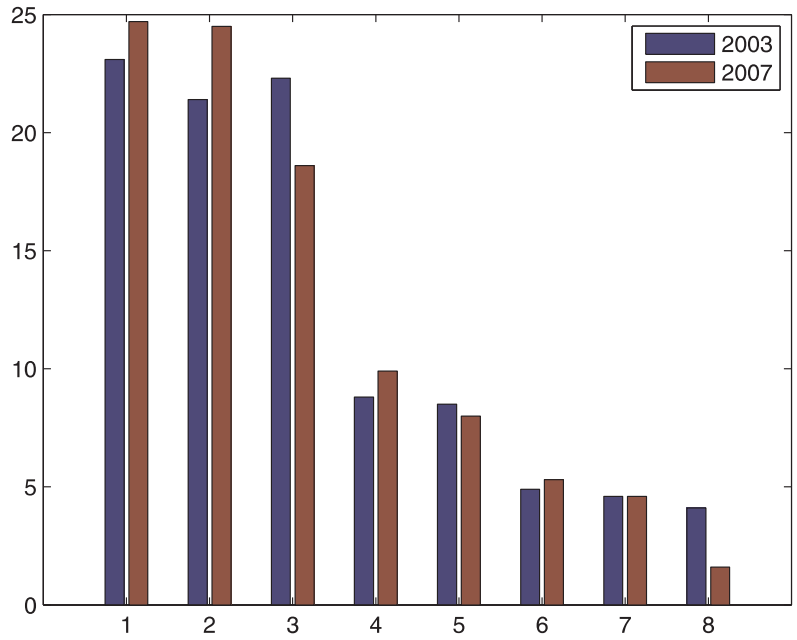
```
>> t = linspace(0,10*pi,100);
>> plot3(sin(t),cos(t),t);
```



Stapeldiagram

- Stapeldiagram åstadkoms med kommandona `bar` och `barh`.

```
>> res
res =
   23.1000   24.7000
   21.4000   24.5000
   22.3000   18.6000
    8.8000    9.9000
    8.5000    8.0000
    4.9000    5.3000
    4.6000    4.6000
    4.1000    1.6000
>> bar(res)
```



Cirkeldiagram

- Cirkeldiagram åstadkoms med kommandona `pie` och `pie3`.

```
>> subplot(1,2,1), pie3(res(:,1)), title('2003');
>> subplot(1,2,2), pie3(res(:,2)), title('2007');
>> legend('C','SDP','Saml.','VF', ...
          'Grön.','KD','SFP','Sannf.');
```