```
% Föreläsning 4 22/2

clear
hold off

 % Vi repeterar en liten del av förra föreläsningen:

% Vi kan definiera en egen funktion på följande sätt:
f = @(x) 2*exp(-x/4) + x.^2 - 7*sin(x)
f(2) % Detta ger nu funktionsvärdet i x=2
% Notera att funktionen är definierad så att den fungerar på vektorer
% (med .^). Detta är nödvändigt om vi exempelvis vill rita dess graf
x = [ 1 2 3 ]
f(x) % Här får vi en vektor med motsvarande funktionsvärden
x=linspace(-4,4);
plot(x,f(x)) % Och grafen
grid on

% Vi fortsätter nu jobba med denna funktion:

% Matlab har ett kommando för att leta nollställen av en funktion
fzero(f,0) % Letar efter ett nollställe till f genom att utgå från x=0
f(ans) % Verkar stämma
fzero(f,2) % Här hittar vi det andra nollstället
% Lämpligen så studerar vi grafen som vi ritat för att se vilka
% x-värden vi skall utgå från.
% Matlab letar nollställen genom att leta punkter där funktionens värden
% har olika tecken, och successivt "stänga in" nollstället
fzero('x.^3',1) % Skulle bli noll, men OK, vi räknar ju numeriskt
% Notera att vi måste ha fnuttar kring funktionsuttrycket då vi anger det
% direkt i fzero
fzero('x.^2',1) % Nu bli Matlab förvirrad!
% Anledningen till förvirringen är att f(x)=x^2 inte antar några negativa
% värden, och därför funkar inte algoritmen

 % Vi tittar nu lite på optimering. Matlab har kommandon för att leta
% minsta värde av en funktion.
g = @(x) 1./((6-x).^2+1)-3*(10-x)./((10-x).^2+3)
x=linspace(0,10);
plot(x,g(x))
grid on
fminbnd(g,0,4) % Letar minpunkt i intervallet [0,4]
% Returnerar alltså x-värdet
g(ans) % Här får vi motsvarande funktionsvärde
fminbnd(g,4,7.5) % Här får vi ena ändpunkten, men ger den minsta värde?
g(4)
g(7.5)
% Vi ser alltså att värdet är mindre i x=7.5. Matlab svarar med EN
% lokal minpunkt, inte nödvändigtvis den som ger minst värde.
fminbnd(g,7.5,9)
fminbnd(g,0,9) % Uppenbarligen inte minpunkten för hela intervallet.
% Vi måste alltså exempelvis också rita grafen så att vi ser vad det är
% som vi får ut.
fminsearch(g,3) % Letar efter ett lokalt min genom att utgå från x=3
fminsearch(g,8)
% Genom att studera grafen först vet vi ungefär var vi skall börja
```

```
% Matlab har inget kommande för maxpunkter, men en maxpunkt för g är en
% minpunkt för -g
gneg = @(x) -g(x)
fminsearch(gneg,6) % Maxpunkt finns nära x=6
g(ans) % Om vi nu vill motsvarande värde av g
fminbnd(gneg,0,10) % Hittar samma punkt
% Största och minsta värde globalt då?
x=linspace(-100,100);
plot(x,g(x))
% Verkar finnas mellan x=0 och x=20:
x=linspace(0,20);
plot(x,g(x))
xmin=fminsearch(g,8)
ymin=g(xmin) % Minsta värdet
xmax=fminsearch(gneg,12)
ymax=g(xmax) % Största värdet

% Väldigt kort om integrering:
quad(g,0,10) % Beräknar integralen av g(x) från x=0 till x=10
quad('x.^2',0,1) % Stämmer
quad('x.*cos(x)',0,pi/2) % Vi återkommer till denna
% Vi kan också integrera en serie mätpunkter
xv=[ 1 2 3 4 5 6 ]
yv=[ 1.7 3.4 2.3 4.6 2.1 4.0 ]
% Följande kommando ger arean under/över kurvan som fås genom att förbinda
% punkterna med räta linjer:
trapz(xv,yv)
% I vårt fall arean under följande kurva:
plot(xv,yv)
axis equal
% Följande ger "integralen fram till varje punkt":
ct=cumtrapz(xv,yv)
plot(xv,ct)

clear

% Vi tittar nu lite på symboliska beräkningar. (Men använd då hellre t.ex.
% Maple som är gjord för sådana.) Vill använda mina tidigare funktioner,
% så vi ritar upp dess grafer igen. Nu definierar vi dem som
% "symboliska funktioner".
syms f(x) g(x)
f(x) = 2*exp(-x/4) + x.^2 - 7*sin(x)
g(x) = 1./((6-x).^2+1)-3*(10-x)./((10-x).^2+3)
% f = @(x) 2*exp(-x/4) + x.^2 - 7*sin(x)
% g = @(x) 1./((6-x).^2+1)-3*(10-x)./((10-x).^2+3)
subplot(2,1,1)
x=linspace(-4,4);
plot(x,f(x))
title('y=f(x)')
grid on
subplot(2,1,2)
x=linspace(0,20);
plot(x,g(x))
title('y=g(x)')
grid on
```

```
syms x % Definierar x som symbolisk variabel
solve(x.^2==0) % Nu kan vi lösa denna
solve(x.^2-2==0) % Vi får också alla lösningar
% fzero gav som bekant bara en lösning
solve(f==0) % Här får vi dock bara en
% Matlab har (av förklarliga skäl) svårt att lösa detta symboliskt.

dg=diff(g) % Så här får man derivatan av g
pretty(dg) % Ganska lustigt kommando
solve(dg==0) % Ja, detta hade vi ju inte så mycket nytta av.
% Lite för komplicerad funktioner. Vi tar en annan:
syms h(x)
h(x) = x^5 + 7*x^2 - 1/x;
solve(h==0) % Detta ger ett vettigt resultat (även alla komplexa lösningar).
h(ans(3)) % Vad nu?
simplify(ans) % Stämmer!
dh=diff(h)
solve(dh==0) % Vi får alla stationära punkter (4 komplexa!?) till h.

ig=int(g) % Ger en privitiv funktion
g-diff(ig) % Borde bli noll. Blir det det?
simplify(ans) % Japp!

int(x.*cos(x),0,pi/2) % Beräknar integralen av x*cos(x) från 0 till pi/2
% Blir kanske "snyggare" än när vi räknade symboliskt tidigare

limit(sin(x)/x,0) % Så här kan man beräkna gränsvärden
limit(g(x),inf)
limit(g(x),-inf) % Verkar stämma med vår graf

subplot(1,1,1) % Återställer till ett fönster


% Vi tittar nu på några viktiga saker/principer när det gäller programmering.
% Detta får räknas som överkurs, men ni kommer säkert att stöta på dem
% förr eller senare.

% Först "for"-loopar.
% Vi skapar en vektor med element från 1 till tio:
V=zeros(1,10) % Först med nollor, så att Matlab vet att V finns.
for i=1:10
    V(i)=i;
end
V % Såja! Vi diskuterar vad som hände.

% Samma sak går att går att göra med en "while"-loop:
V=zeros(1,10)
i=1;
while i<=10
V(i)=i;
i=i+1;
end
V % Vi diskutera återigen vad som hände.
```

```
% Samma resultat får vi, som vi tidigare sett, med
V=1:10

% Slutligen "if-else"
for i=1:10
    if V(i)<=5
        V(i)=1;
    else
        V(i)=2;
    end
end
V % Ja, vi diskuterar vad som hände.

% En aktuell fråga:
S=0;
for i=1:10000
    A=rand(49);
    if rank(A)<49
        S=1;
    end
end
S % Vad gjorde jag?


% Nu kollar vi på er:

A=imread('studenter.jpg'); % Laddar in er
imshow(A) % Här är ni

size(A) % Här ser vi ett exempel på en "tredimensionell" matris.
% Innehåller 3 stycken 2160*3840-matriser, en för varje "grundfärg"
% rög, grön och blå.
A(:,:,1)=0; % Nollställer era röda jag,
A(:,:,3)=0; % och era blåa,
imshow(A) % så nu är ni bara gröna.

A=imread('studenter.jpg'); % Laddar in igen
B=rgb2gray(A); % Gör er i gråskala
imshow(B)
size(B) % Nu är ni bara "en vanlig" matris.

C=B; % Vill bara ha en ny matris av samma typ
S=size(B,2) % Antalet kolonner i B
for i=1:S
    C(:,i)=B(:,S+1-i); % Slänger om kolonnerna
end
imshow(C) % Då blir bilden spegelvänd

D=B;
S=size(B,1) % Antal rader
for i=1:S
    D(i,:)=B(S+1-i,:); % Slänger om raderna
end
imshow(D) % Upp och ner
```

```
% "Svärtan" i varje pixel ges av ett tal mellan 0 och 255.
E=B;
E(:,:)=255; % Fyller med 255:or
F=E-B;
imshow(F) % Vi får ett negativ!

G=B'; % Jag transponerar slutligen er.
imshow(G)
```