```
% Föreläsning 3 15/2

clear
hold off

% Vi återvänder till kommandot A\Y som löser AX=Y

% Åter till ekvationssystemen från föreläsning 1.
% Uppgift 1.3 i övningsboken:
A1=[ 1 -2 1 ; 2 -6 6 ; -3 5 1 ] % Matar in systemmatrisen
Y1=[ 1  2  3 ]' % Och högerledet
A1\Y1 % Detta ger lösningen
% Uppgift 1.4:
A2=[ 1 -2 1 ; 2 -6 6 ; -3 5 -1 ] % Systemmatrisen
Y2=[ 1  2  3 ]' % Högerledet
A2\Y2 % Detta skall sakna lösning.
% Uppgift 1.11:
A3=[ 1 -2 1 ; 2 -6 6 ; -3 5 -1 ] % Systemmatrisen
Y3=[ 1  4  -2 ]' % Högerledet
A3\Y3 % Här skulle vi ju ha parameterlösning.

% Kommandot A\Y klagar (för kvadratiska matriser) på fallen
% då vi har ingen eller oändligt många lösningar. Vi låter Matlab
% eliminera istället:
rref([A1,Y1]) % "Eliminerar systemet". Här får vi entydig lösning.
rref([A2,Y2]) % Notera "0=1" i sista ekvationen.
rref([A3,Y3]) % Nu har vi "0=0"
rref(A2) % "Eliminerar bara en matris"

% Vi kollar även inversen till systemmatriserna:
inv(A1) % Finns
inv(A2) % Finns inte ("singulär" matris samma som ej inverterbar)
inv(A3) % Finns inte heller
% Du har kanske misstänkt att det finns entydig lösning precis
% då systemmatrisen är inverterbar.

% Vi återvänder till uppgift 1.14 där vi har fler obekanta än ekvationer,
% och förväntar oss parameterlösning
A=[ 2 3 4 ; 4 -3 2 ] % Systemmatrisen
Y=[ 5 1 ]' % Högerledet
A\Y % Som vi sett ger Matlab en lösning, men bara en.
% Hur kan vi få alla? Repetition:
% Alla lösningar till AX=Y = en lösning till AX=Y + alla till AX=0.
% Vi har fått en lösning ovan, och alla till AX=0 är precis nollrummet:
null(A) % Detta ger en bas för nollrummet, med vektorer av längd 1
null(A,'r') % Ger "rationella koordinater", vilket kanske känns bättre ibland
% För att få alla lösningar kan vi alltså göra så här:
XP=A\Y
XH=null(A,'r')
syms t % Vi sätter in en parameter som vi låter vara symbolisk.
X=XP+t*XH % Så här brukar vi ju skriva vår lösning.
% Vi provar också med uppgift 1.16, där vi förväntar oss två parametrar:
A=[ 2 1 -1 3 -3 ; 3 2 1 2 2 ; -4 3 2 1 -1 ] % Systemmatrisen
Y=[ 0 0 0 ]' % Högerledet
A\Y % En lösning, som vi såklart kunde se direkt! Här är lösningen såklart
% bara nollrummet.
```

```
XH=null(A,'r') % Kolonnerna ger en bas för nollrummet
XH1=XH(:,1) % Plockar ut första kolonnen,
XH2=XH(:,2) % och andra
syms t1 t2
X=t1*XH1+t2*XH2 % Vi får nog en bättre bild av att bara titta på matrisen XH

% Exemplet vi hade igår:
A=[ 2 1 -1 ; 1 2 1 ; -1 1 2] % Eller, detta var egentligen 3*A
R=rank(A) % Ger rangen
N=null(A,'r') % Nollrummet
S=size(N) % Vi tar ut typen på matrisen där basen för nollrummet finns.
Nd=S(2) % Tar ut andra elementet i S, dvs. antal kolonner i N, som är
% antalet basvektorer till nollrummet.
size(N,2) % Ett alternativ till de två senaste raderna. Varför?
R+Nd % Skall bli antalet kolonner enligt dimensionssatsen

% Vi kollar också rangen av matriserna i de tre första exemplen:
rank(A1)
rank(A2)
rank(A3)
% Stämmer med våra beräkningar ovan av eventuella inverserna, för en
% matris är ju inverterbar precis då kolonnerna utgör en bas, dvs. då
% de är linjärt oberoende och vi har full rang.

% Vad händer när vi kör kommandot A\Y för kvadratiska system?
% Är det så att Matlab försöker invertera matrisen?
% Vi kollar genom att återgå till det första exemplet:
inv(A1)*Y1 % Ger såklart samma resultat som A1\Y1. Men gör Matlab samma
% sak? Vi provar genom att mäta tidsåtgången:
A=rand(1000); % 1000*1000-matris med slumptal
Y=rand(1000,1); % Högerled
tic; A\Y; toc % Ganska lustigt kommando
tic; inv(A); toc
% Går alltså (minst) dubbelt så snabbt med A\Y, så Matlab inverterar här
% inte matrisen. Slutsats: Använd A\Y om ni har bråttom!

% Vi påminner också om att vi förra gången såg vad som händer om vi har
% fler ekvationer än obekanta. Systemet i uppgift 1.9 är på denna form,
% och som väntat har det ingen lösning. Men Matlab ger lösning med
A=[ 1 1 ; 1 -2 ; 3 4 ]
Y=[ -4 2 1]'
A\Y
% Lösningen vi får är i minsta-kvadrat-mening

% Vi tittar nu på ett förskräckligt exempel, och löser systemet
%     1.12065 x1 + 0.98775 x2 = 2.12341
%     2.24135 x1 + 1.97553 x2 = 4.24601
A=[ 1.12065 0.98775 ; 2.24135 1.97553 ];
Y=[ 2.12341 4.24601 ]';
A\Y
% Kolla nu på
%     1.12065 x1 + 0.98775 x2 = 2.12241
%     2.24135 x1 + 1.97553 x2 = 4.24601
% där vi bara ändrat det övre talet i högerledet med en tusendel.
% Detta borde väl inte ändra lösningen så mycket, eller?
Y=[ 2.12241 4.24601 ]'; % Samma A, men Y lite ändrad
```

```
A\Y
% Resultatet blir helt annorlunda! Detta system är alltså väldigt "känsligt
% för störningar". Måttet på denna känslighet ges av det som kallas
% "konditionstalet" av systemmatrisen.
cond(A) % Detta ger konditionstalet
% Utan att gå in på detaljer så konstaterar vi att detta konditionstal
% är högt. Var blir det för en "godtycklig" 2*2-matris? Kör följande två
% rader ett antal gånger:
A=rand(2) % 2*2 med slumptal
cond(A)
% Ett system som är lite (minst) känsligt för störningar är
%       x1     = y1
%           x2 = y2
% och motsvarande systemmatris, som är enhetsmatrisen, har lägsta möjliga
% konditionstal, som är 1
A=eye(2)
cond(A)
% Vad tror du en matris skall ha för egenskaper för att ha konditionstal 1?


% Funktioner

% 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
% Vi kommer att titta vidare på denna funktion nästa gång, och exempelvis
% derivera, intergrera och optimera den.

% I matlab finns ingen funktion som direkt beräknar nolldimensionen. Vi
% skapar nu en sådan funktion:
nolld = @(A) size(null(A),2)
A = [ 1 2 ; 3 4 ]
nolld(A)
B = [ 1 2 ; 2 4 ]
nolld(B) % Variabeln som vi skickar in kan såklart heta vad som helst
% Funktioner definierade på detta sätt måste beskrivas på "en rad".
% Dessutom försvinner de när vi rensar minnet:
clear
B = [ 1 2 ; 2 4 ]
nolld(B) % Borta!

% Många gånger kan det vara bra att skapa en funktion i en egen m-fil,
% genom att göra så här:
% Skriv in följande fem rader i ett nytt script:
function y = nolldim(A) % Funktionen heter "nolldim"
N = null(A);
s = size(N);
y = s(2); % Ger resultatet som returneras
end % Behövs egentligen inte
% Spara scriptet som "nolldim.m" (föreslås automatiskt)
```

```
% Nu kan man använda funktionen:
nolldim(B)
clear
B = [ 1 2 ; 2 4 ]
nolldim(B) % Funktionen finns såklart kvar nu
% Se till så att du verkligen förstår vad som händer när vi definierar vår
% funktion "y = nolldim(A)"; att vi läser in en matris A som vi kan jobba med,
% hur vi till sist anger vad "y" skall vara, och så vidare.
```