Projekt: Ljudvågor

SF1547 Numeriska metoder

Grupp 38

Larisa Cof
19981121-7323
larisac@kth.se

Robin Dahlkvist
19970319-1651
robindah@kth.se

Problem 1

För en samling datapunkter definierade enligt den icke-linjära funktionen

$$c(z) = 4800 + p_1 + p_2 \frac{z}{1000} + p_3 \, \mathrm{e}^{-p_4 \, z/1000}.$$

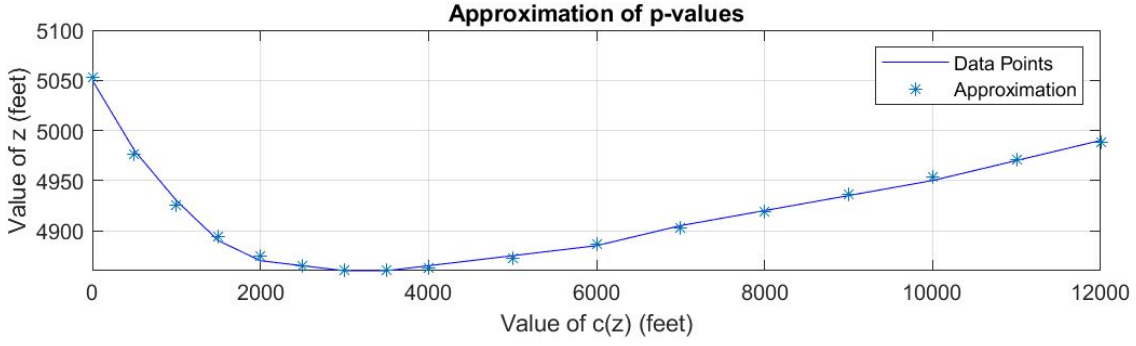
skulle värden på p1, p2, p3 och p4 bestämmas i minsta kvadratmening med hjälp av Gauss-Newtons metod. Inledningsvis formulerades uppgiften som ett linjärt problem genom att ansätta p4 = 1 (startgissning p1-p4: [0, 0, 0, 1]). Ekvationssystemet löstes med hjälp av MATLAB:s inbyggda funktion för icke-linjära MKM-problem `lsqnonlin` samt de givna punkterna för (z, c(z)). Därefter ansattes den linjära lösningen som startvärdet i en funktion g(p1, p2, p3, p4) för vilken Jacobi-matrisen togs fram:

```
g = @(p1,p2,p3,p4) 4800 + p1 + p2 .* (z/1000) + p3 .*  exp(-p4 .* (z/1000)) - cV;

[ 1,   0,             1,                      0];
[ 1, 1/2,     exp(-p4/2),       -(p3*exp(-p4/2))/2];
[ 1,   1,       exp(-p4),             -p3*exp(-p4)];
[ 1, 3/2, exp(-(3*p4)/2), -(3*p3*exp(-(3*p4)/2))/2];
[ 1,   2,     exp(-2*p4),         -2*p3*exp(-2*p4)];
[ 1, 5/2, exp(-(5*p4)/2), -(5*p3*exp(-(5*p4)/2))/2];
[ 1,   3,     exp(-3*p4),         -3*p3*exp(-3*p4)];
[ 1, 7/2, exp(-(7*p4)/2), -(7*p3*exp(-(7*p4)/2))/2];
[ 1,   4,     exp(-4*p4),         -4*p3*exp(-4*p4)];
[ 1,   5,     exp(-5*p4),         -5*p3*exp(-5*p4)];
[ 1,   6,     exp(-6*p4),         -6*p3*exp(-6*p4)];
[ 1,   7,     exp(-7*p4),         -7*p3*exp(-7*p4)];
[ 1,   8,     exp(-8*p4),         -8*p3*exp(-8*p4)];
[ 1,   9,     exp(-9*p4),         -9*p3*exp(-9*p4)];
[ 1,  10,    exp(-10*p4),       -10*p3*exp(-10*p4)];
[ 1,  11,    exp(-11*p4),       -11*p3*exp(-11*p4)];
[ 1,  12,    exp(-12*p4),       -12*p3*exp(-12*p4)]];
```

Notera att cV är de givna värdena på c(z) i uppgiften.

Genom användning av Gauss-Newtons metod erhölls de värden på p1-p4 som minimerade den euklidiska (2)-normen. Värden för p1-p4: `[-20.209  17.337  272.91  0.75278]`.



Svaren som erhållits genom Gauss-Newton beräknas med en felmarginal på 10^{-10}.

Problem 2

Givet en utgångspunkt och utgångsriktning, skulle ljudvågor som beskrivs enligt differentialekvationen z(x) av ordning 2

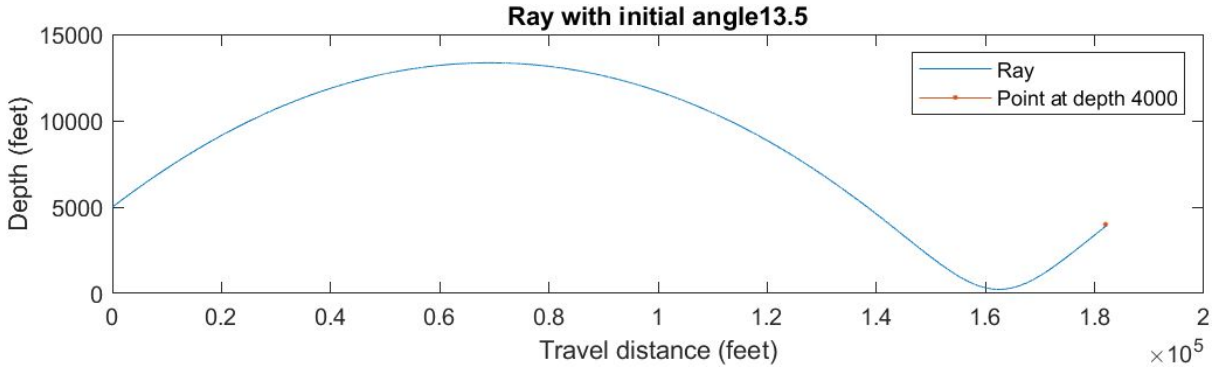
$$\frac{d^2z}{dx^2} = -q_0\frac{c'(z)}{c(z)^3}, \qquad q_0 = (c(z_0)/\cos\beta_0)^2$$

spåras. Ekvationens lösning berodde på följande begynnelsevärden: $x = 0$, $z = z_0$, $dz/dx = \tan\beta_0$, där β_0 utgjordes av startvinkeln. Givet i uppgiften var även att vid djupet x = 30 sjömil skall z vara nära 4000 fot. Systemet löstes genom att först utföra en transformation från andra ordningens ODE till ett system av första ordningen enligt följande:

$$u = dz/dx$$

$$u' = d^2z/dx^2.$$

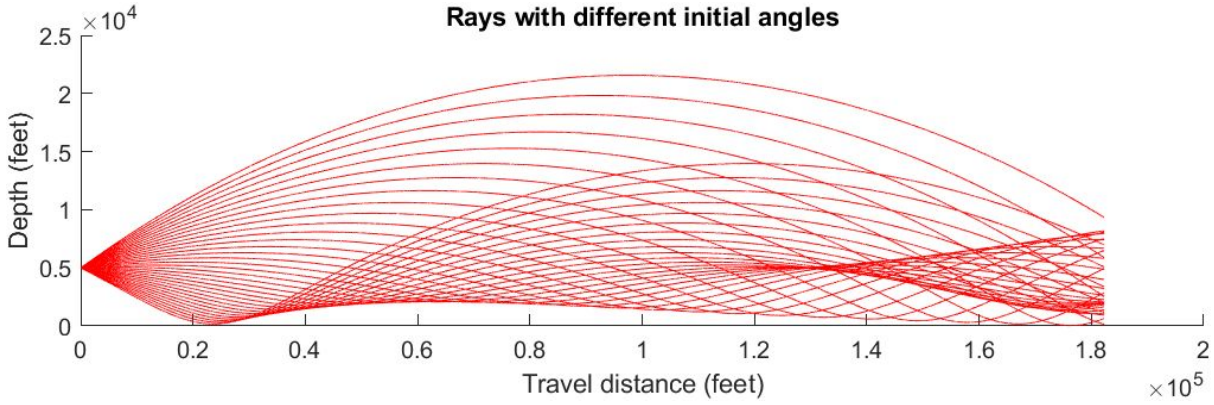
Sedan användes MATLAB:s funktion `ode45` för att lösa systemet. Lösningen till differentialekvationen, ljudvågens väg under 30 sjömil, visualiseras i plotten nedan.



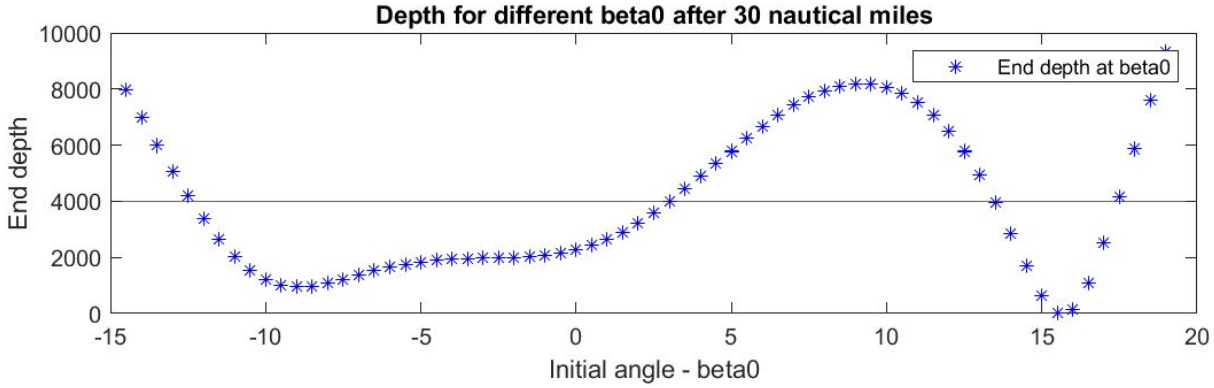
Den relativa toleransen för `ode45` kan justeras manuellt i MATLAB för att uppnå önskad noggrannhet. Felet i svaren som erhålls från `ode45` är av ordning 10^{-6}, därav görs inga vidare justeringar av toleransen då värdena anses vara tillräckligt väl approximerade för lösningen av problemet.

Problem 3

En ljudkälla på djupet 5000 fot transmitterar ljudsignaler till en mottagare 30 sjömil bort vid ett djup av 4000 fot. På grund av den icke-linjära karaktären hos ekvationen i Problem 2, medför det att strålar som lämnar mottagaren med olika utgångsvinklar kan nå mottagaren. I plotten nedan visualiseras de strålar med utgångsvinkel β_0 mellan -15 upp till 20 grader som når mottagaren på djupet 4000 fot efter 30 sjömil.



Plot över slutdjupet $z(x_f)$ som en funktion av β_0 :



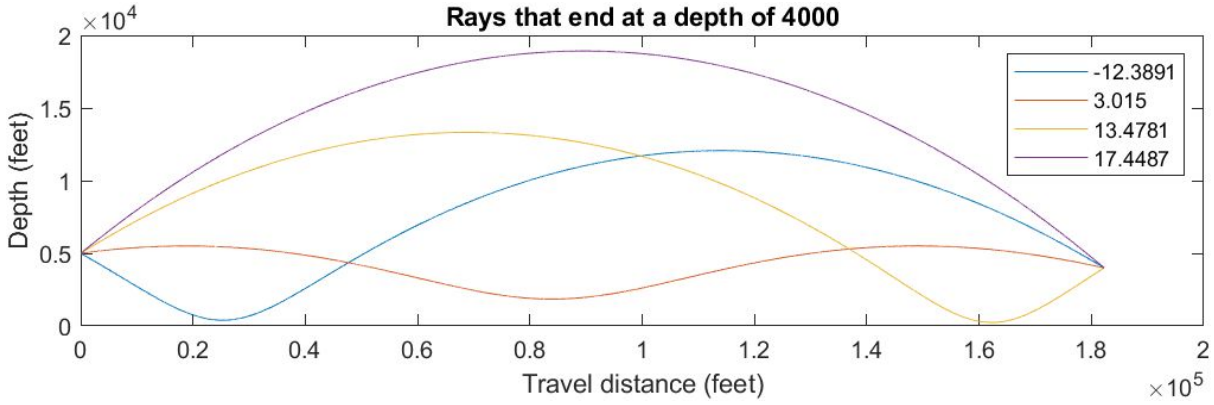
Problem 4

Genom användning av sekantmetoden, söks de värden på β_0 för vilka $z(x_f) = 4000$. För att lösa uppgiften ansattes en funktion $f(x) = z(x_f) - 4000$ i syfte om att hitta värden på β_0 som uppfyller noggrannhetskravet bäst. Sekantmetoden kräver två startgissningar. I plotten till föregående uppgift visualiseras ändpunkterna (efter 30 sjömil) i förhållande till β_0, där går

det att avläsa ungefär för vilka β_0 som z(x) närmar sig 4000 fot. Med hjälp av plotten utsågs passande startpunkter för att lösa problemet:

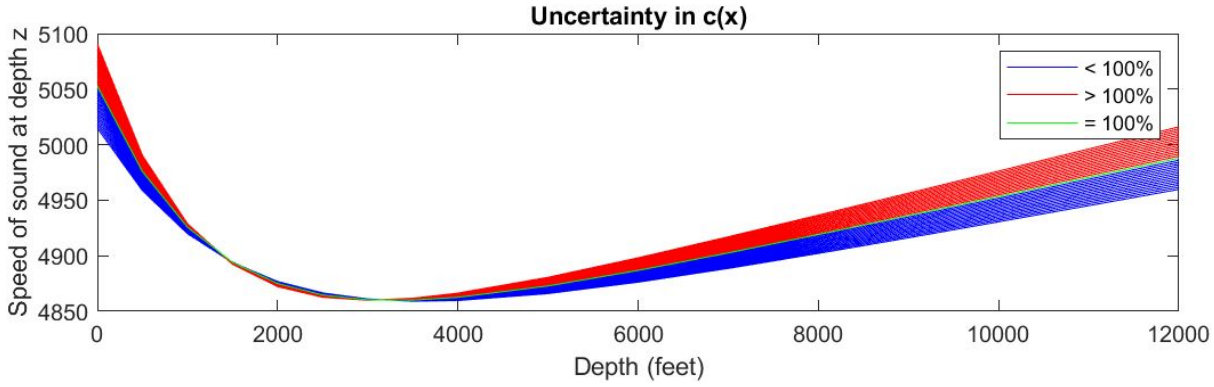
```
startg = [-13, -12; 4 ,5; 13,14; 17,18];
```

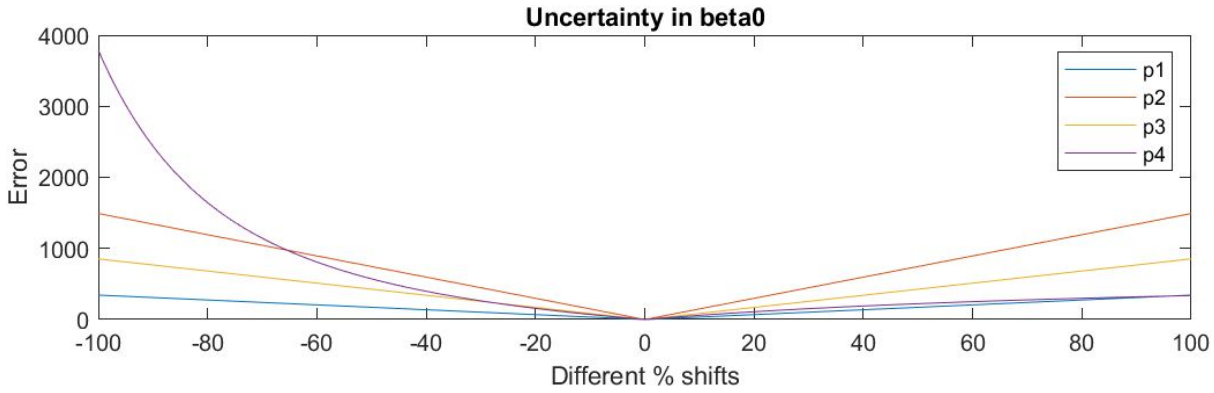
Varje par av punkter för β_0 utgör en startgissning.



Problem 5

I denna del av uppgiften skulle det antas att koefficienterna p1-p4 erhållits med 1 % osäkerhet efter MKM-lösningen i problem 1. I den första plotten visualiseras instanser av c(z), där p1-p4 störs med -15 % avvikelse upp till +15 % (grön linje motsvarar de datapunkter som erhölls i lösningen till problem 1). Vidare utfördes en störningsanalys i syfte av att uppskatta osäkerheten i resultatet då koefficienterna p1-p4 störtes en i taget. I denna plot sattes störningsfaktorn från -100 % till +100 %, sedan togs summan av absolutbeloppen för lösningarnas störningar för att skildra osäkerheten i modellen.





När p1-p4 enskilt avviker upp till 1 % har de ingen större påverkan på det totala felet. I plotten visualiseras varje koefficients enskilda påverkan på felet beroende på störningsfaktorn. Utifrån denna plott kan en slutsats dras om att p2 har störst inverkan på det totala felet. Intressant att tillägga är p4 övergår till att ha störst inverkan på det totala felet när störningsfaktorn är ca 66 %.

Egen arbetsinsats

Vi har tagit del av den hjälp som försetts i samband med laborationshandledningarna. Vi fick hjälp med att tolka hur vi skulle kunna gå tillväga för att lösa vissa deluppgifter samt respons på den kod vi skrivit dittills och vad eventuella fel i koden kunde bero på.
Utöver detta har vi använt oss av olika källor på internet som berört liknande uppgifter utformade efter P8-15 i Kahaner-Moler-Nash.
Den tillämpade arbetsmetoden som vi jobbade enligt var parprogrammering där vi samarbetat via Zoom-möten samt använt oss av git.

Kod i MATLAB

```
%All kod för uppgift Ljudvågor är i detta dokument

%Skapare: Robin Dahlkvist, Larisa Cof


%
clc; clf; clearvars; clear all;
format long g;
```

```
global grader;



%----------------- a) ----------------------%



%Värden på z
z = [    0,
500,1000,1500,2000,2500,3000,3500,4000,5000,6000,7000,8000
,9000,10000,11000,12000]';
%Värden på c(z)
cV =
[5050,4980,4930,4890,4870,4865,4860,4860,4865,4875,4885,49
05,4920,4935, 4950, 4970, 4990]';
%Funktionen
func = @(p) -cV + 4800 + p(1) + p(2) .* (z/1000) + p(3) .*
exp(-p(4) .* (z/1000))
%Stargissningarna
p = [0 0 0 1]; %p(4) = 1;
%Vi använder oss av MATLAB's metod för att lösa MKM
p = lsqnonlin(func,p)


%Vårt p innehåller nu våra stargissningar.


syms p1 p2 p3 p4
%Funktion vi använder för att få frab jacobian
g = @(p1,p2,p3,p4) 4800 + p1 + p2 .* (z/1000) + p3 .*
exp(-p4 .* (z/1000)) - cV;


%Om man skriver detta så får vi jacobian matrisen vi har
nedan
%J = jacobian(g, [p1 p2 p3 p4]);


J1 = @(p1,p2,p3,p4)[
[ 1,   0,              1,                          0];
[ 1, 1/2,     exp(-p4/2),       -(p3*exp(-p4/2))/2];
```

```
[ 1,   1,       exp(-p4),             -p3*exp(-p4)];
[ 1, 3/2, exp(-(3*p4)/2), -(3*p3*exp(-(3*p4)/2))/2];
[ 1,   2,     exp(-2*p4),          -2*p3*exp(-2*p4)];
[ 1, 5/2, exp(-(5*p4)/2), -(5*p3*exp(-(5*p4)/2))/2];
[ 1,   3,     exp(-3*p4),          -3*p3*exp(-3*p4)];
[ 1, 7/2, exp(-(7*p4)/2), -(7*p3*exp(-(7*p4)/2))/2];
[ 1,   4,     exp(-4*p4),          -4*p3*exp(-4*p4)];
[ 1,   5,     exp(-5*p4),          -5*p3*exp(-5*p4)];
[ 1,   6,     exp(-6*p4),          -6*p3*exp(-6*p4)];
[ 1,   7,     exp(-7*p4),          -7*p3*exp(-7*p4)];
[ 1,   8,     exp(-8*p4),          -8*p3*exp(-8*p4)];
[ 1,   9,     exp(-9*p4),          -9*p3*exp(-9*p4)];
[ 1,  10,    exp(-10*p4),        -10*p3*exp(-10*p4)];
[ 1,  11,    exp(-11*p4),        -11*p3*exp(-11*p4)];
[ 1,  12,    exp(-12*p4),        -12*p3*exp(-12*p4)]];



%Gauss Newton från Lab 2.


%Vi tar våra punker p från MGM.
X=p';
tau=1e-10;
dxnorm=1;
iter=0;


while (dxnorm>tau && iter<100)
   dx = -J1(X(1),X(2),X(3),X(4))\g(X(1),X(2),X(3),X(4));
   dxnorm = norm(dx);
   X = X + dx
   iter = iter+1;
   disp([iter dxnorm])
end


p=X';


%Nu har vi förbättrade variabler med en högre nogrannhet i
X.
```

```
%Plotta kurvan c(z)
g = @(z) 4800 + X(1) + X(2) .* (z/1000) + X(3) .*
exp(-X(4) .* (z/1000));
for i=1:1:size(z)
   sum2(i) = g(z(i)');
end
disp("Plot 1");
subplot(4,2,1);
plot(z, cV, 'b');
hold on;
plot(z, sum2, '*');
legend("Data Points","Approximation");
xlabel("Value of c(z) (feet)");
ylabel("Value of z (feet)")
title("Approximation of p-values")
grid on;
```

```
%----------------- b) ----------------------%
```

```
disp("Plot 2");
subplot(4,2,2);
%Vi går från 0 till 30 nautical miles
x=0:10:6076*30;
%Ode45
grader = 13.5;
[t,y]=ode45(@fDiff,x,[5000 tand(grader)])
%Plotta z(x)
plot(t,y(:,1));
xlabel("Travel distance (feet)");
ylabel("Depth (feet)")

hold on
%Plotta punkten vi 30 nautical miles, som ligger nära ett
djup på 4000 feet
```

```
plot(182000,4000,'.-');
legend("Ray", "Point at depth 4000")
title("Ray with initial angle" + grader);



%----------------- c) ----------------------%



disp("Plot 3");
subplot(4,2,3);


clear endpoints endvalues;
%Intervall av olika grader
degreesinterval = -14.5:0.5:19;
iter = 1;
for i=degreesinterval
   x=0:10:6076*30;
   %Global variable som behövs för att i ska vara den
statisk i ode.
   grader = i;
   [t,y]=ode45(@fDiff,x,[5000 tand(grader)]);
   %Sparar värdet efter 30 natuical miles för att plotta
nästa nästa graf.
   endvalues(iter) = y(end,1);
   %Så att man tydligare ska kunna se graferna.
   if mod(iter,2) == 0
       plot(t,y(:,1), 'r')
   end
   hold on
   iter = iter + 1;
end
title("Rays with different initial angles")
xlabel("Travel distance (feet)");
ylabel("Depth (feet)")



%----------------- d) ----------------------%
```

```
%Plotta punkter med avseende på deras djup efter 30
nautical miles för ett
%viss beta0.


disp("Plot 4");
subplot(4,2,4);
plot(degreesinterval, endvalues, 'b*');
yline(4000);
legend("End depth at beta0")
title("Depth for different beta0 after 30 nautical miles")
xlabel("Initial angle - beta0");
ylabel("End depth")




%Genom att analysera figure 4 så ser vi vart linjen y=4000
och grafen med
%"endvalues" korsar varandra. Vi tar då värden nära denna
korsning och
%baserar våra startgissningar på det.


startg = [-13, -12; 4 ,5; 13,14; 17,18];


%Vi har då 8 stargissningar, av 4 par.
%Vi löser detta problem med sekantmetoden.
for i=1:1:size(startg)
   counter = 2;
   M = 4000;
   clear a;
   a(1) = startg(i,1);
   a(2) = startg(i,2);
   disp('    a            e          ');
   maxIt = 100;
   tolerance = 1e-8;
   error(1) = abs(a(2) - a(1));
   disp([a(2) error(1)]);

   while(error(counter-1) >= tolerance) && (counter <=
maxIt)
       x=0:10:6076*30;
```

```
        %f(xn)
        grader = a(counter);
        [t,y]=ode45(@fDiff,x,[5000 tand(a(counter))]);
        fxn = y(end,1) - M; %f(x)

        %f(xn-1)
        grader = a(counter-1);
        [t,y]=ode45(@fDiff,x,[5000 tand(a(counter-1))]);
        fxn1 = y(end,1) - M; %(f(x-1))

        %Ekvationen
        current = a(counter) - fxn .*
((a(counter)-a(counter-1))./(fxn-fxn1));


        a(counter+1) = current;
        error(counter) = abs(a(counter) - a(counter-1));
        disp([current error(counter)]);
        counter = counter + 1;
    end
    %Sparar varje resultat för varje par av startgissning
    results(i) = a(end);

end


%Nu plottar vi de graför där för specifica vinklar på
beta0 ger att
%slutpunkten hamnar på djupet 4000 feet.


disp("Plot 5");
subplot(4,2,5);
for i=1:1:size(results')
    x=0:10:6076*30;
    grader = results(i);
    [t,y]=ode45(@fDiff,x,[5000 tand(grader)]);
    plot(t,y(:,1))
    hold on
end
legend("" + results(1),"" + results(2),"" + results(3),""
+ results(4))
xlabel("Travel distance (feet)");
```

```
ylabel("Depth (feet)")
title("Rays that end at a depth of 4000")



%----------------- e) ----------------------%


%Visualisera osäkerheten
fUncertainty = @(p) 4800 + p(1) + p(2) .* (z/1000) + p(3)
.*  exp(-p(4) .* (z/1000));


%Vi börjar med att räkna ut hur mycket våra värden på c(z)
skulle avika
%beroende på hur mycket våra värden på p skulle skifta
mellna 85% och 115%
%procent. Genom att göra detta ser man tydligen i plotten
hur och vart
%datan påverkas som mest med dessa skiftningar.


subplot(4,2,6);


%För att få legend
booleanCheck = [0 0 0];


plotCounter = 0;


for i=-15:1:15
   plotCounter = plotCounter + 1;
   clear pTemp cVTemp;
   pTemp = (1 + (i/100)) .* p;
   cVTemp = fUncertainty(pTemp);
   if i < 0
       if booleanCheck(1) == 0
           plots(plotCounter) = plot(z, cVTemp,'b');
           booleanCheck(1) = 1;
       else
           plots(plotCounter) = plot(z, cVTemp,'b');
```

```
        end

    elseif i > 0
        if booleanCheck(2) == 0
            plots(plotCounter) = plot(z, cVTemp,'r');
            booleanCheck(2) = 1;
        else
            plots(plotCounter) = plot(z, cVTemp,'r');
        end

    else


    end
    hold on
end
plots(plotCounter) = plot(z, fUncertainty(p),'g-');
title("Uncertainty in c(x)")
legend([plots(1) plots(20) plots(end)],'< 100%','>
100%','= 100%')
xlabel("Depth (feet)");
ylabel("Speed of sound at depth z")


%Estimate the resulting uncertainty in beta0
%Vi måste göra en störningsanalys


%%Vi beräknar nya värden på c(z) som är mer exakta med
våra nya värden på p
exaktCV = fUncertainty(p);


%Hur många procent åt vardera håll vi vill skifta varje p
uncertaintyLimit = 100;
%Intervallet
uncertaintyInterval =
-uncertaintyLimit:1:uncertaintyLimit;
clear cVp1_vektor cVp2_vektor cVp3_vektor cVp4_vektor


for i=uncertaintyInterval
storningsfaktor = (1 + (i/100));
clear p_p1 p_p2 p_p3 p_p4 cVp1 cVp2 cVp3 cVp4
```

```
%Här ändrar vi specifika p så att de påverkar av en viss
störningsfaktor
p_p1 = [p(1)*storningsfaktor p(2:4)];
p_p2 = [p(1) p(2)*storningsfaktor p(3:4)];
p_p3 = [p(1:2) p(3)*storningsfaktor p(4)];
p_p4 = [p(1:3) p(4)*storningsfaktor];


%Beräkna störda lösningar, där parametrarna störs en i
taget.
cVp1 = fUncertainty(p_p1);
cVp2 = fUncertainty(p_p2);
cVp3 = fUncertainty(p_p3);
cVp4 = fUncertainty(p_p4);


%Summera ihop absolutbeloppen av lösningarnas störningar
cVp1_vektor(i + (uncertaintyLimit + 1)) = sum(abs(cVp1 -
exaktCV));
cVp2_vektor(i + (uncertaintyLimit + 1)) = sum(abs(cVp2 -
exaktCV));
cVp3_vektor(i + (uncertaintyLimit + 1)) = sum(abs(cVp3 -
exaktCV));
cVp4_vektor(i + (uncertaintyLimit + 1)) = sum(abs(cVp4 -
exaktCV));
end


%Nu plottar vi dessa 4 grafer för varje p som
representerar hur stor
%påverkan de har i ekvationen.
subplot(4,2,7);
plot(uncertaintyInterval, cVp1_vektor);
hold on;
plot(uncertaintyInterval, cVp2_vektor);
hold on;
plot(uncertaintyInterval, cVp3_vektor);
hold on;
plot(uncertaintyInterval, cVp4_vektor);
hold on;
legend("p1", "p2", "p3", "p4");
xlabel("Different % shifts")
ylabel("Error")
title("Uncertainty in beta0")
```

```
%ODE funktion


function yprim=fDiff(t,y)
   %Gissningar
   x = [-20.2089965981773          17.3367686744559
272.905725401301         0.752778474783313];
   %Funktionen
   c = @(z) 4800 + x(1) + x(2) .* (z/1000) + x(3) .*
exp(-x(4) .* (z/1000));
   %z0 i feet
   z0 = y(1);
   %beta0 i grader
   global grader;
   beta0 = grader;
   zprim = y(2);
   q0 = (c(5000)/(cosd(beta0)))^2;
   cprim = @(z) x(2)/1000 -
(x(3)*x(4)*exp(-(x(4)*z)/1000))/1000;
   zbis = @(z) -q0 .* (cprim(z)/c(z)^3);

   %Det vi returnerar
   yprim=zeros(2,1);
   yprim(1) = zprim;
   yprim(2) = zbis(z0);
end
```