MECH 579
Multidisciplinary Design Optimization
Siva Nadarajah

Project 3: Brachistochrone Problem

17/20

Dritan Harizaj
260426327

October 30, 2013

Table of Contents

Introduction

The aim of this project is to solve the Brachistochrone problem, i.e. to find the quickest path of descent between two points, assuming only the gravitational force is acting on the system. In more mathematical terms,

Minimize $T = \int_{x_i}^{x_f} \frac{ds}{V}$, with respect to x$\in\Re^n$, subject to gravitational force.

The derivation of the solution to this problem (both analytical and numerical) is included in the notes. The exact solution to the problem is given by a cycloid:

$$y(\theta) = \frac{1}{2}C^2(1-\cos\theta)$$
$$x(\theta) = \frac{1}{2}C^2(\theta-\sin\theta)$$

In order to minimize the time using numerical methods, the x-dimension was discretized in several intervals $x_j = j\Delta x$, while the y-dimension was initialized to a straight line and the ends kept fixed. The gradient used to test for convergence was given by:

$$G = -\frac{1+y_j'^2+2y_jy_j''}{2\left(y_j\left(1+y_j'^2\right)\right)^{3/2}}$$

Where the derivatives of y_j are given by the following finite difference formulas:

$$y_j' = \frac{y_{j+1}-y_{j-1}}{2\Delta x}$$
$$y_j'' = \frac{y_{j+1}-y_j+y_{j-1}}{(\Delta x)^2}$$

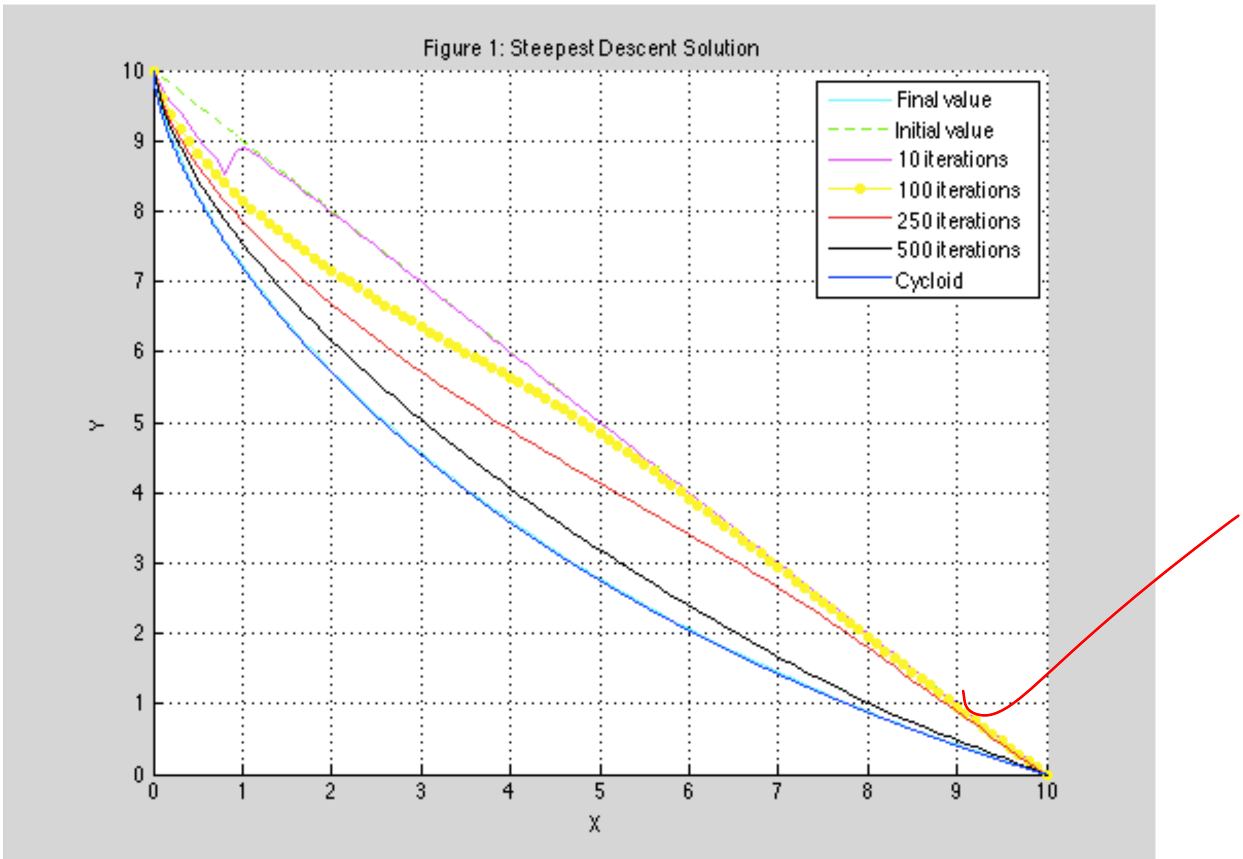
To minimize computation time, the points (0,10) and (10,0) were chosen as the fixed values of y, and the mesh was divided into 100 elements. A simple backtracking algorithm was used with $\alpha_0 = 1$, $\rho = 0.9$, $c = 10^{-4}$, while the tolerance was kept at 10^{-3}.

Several "snapshots" of the intermediate shapes were taken at various points of the program runtime. Both a steepest descent and a DFP Quasi-Newton line search method were used to find the shape of steepest descent.

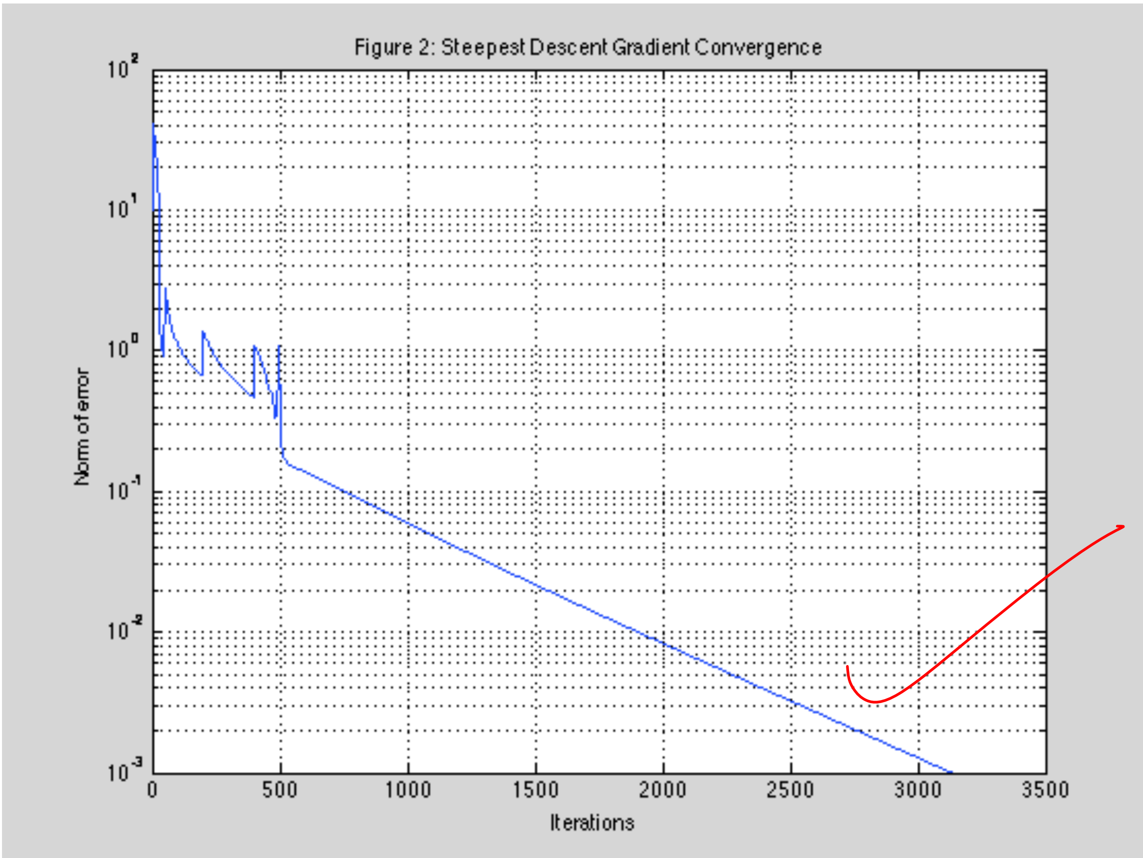
To demonstrate the effect of varying parameters, ρ was changed to 0.4, then a random initial shape was chosen, and finally the middle point of the line was kept fixed.

Part 1: Steepest Descent

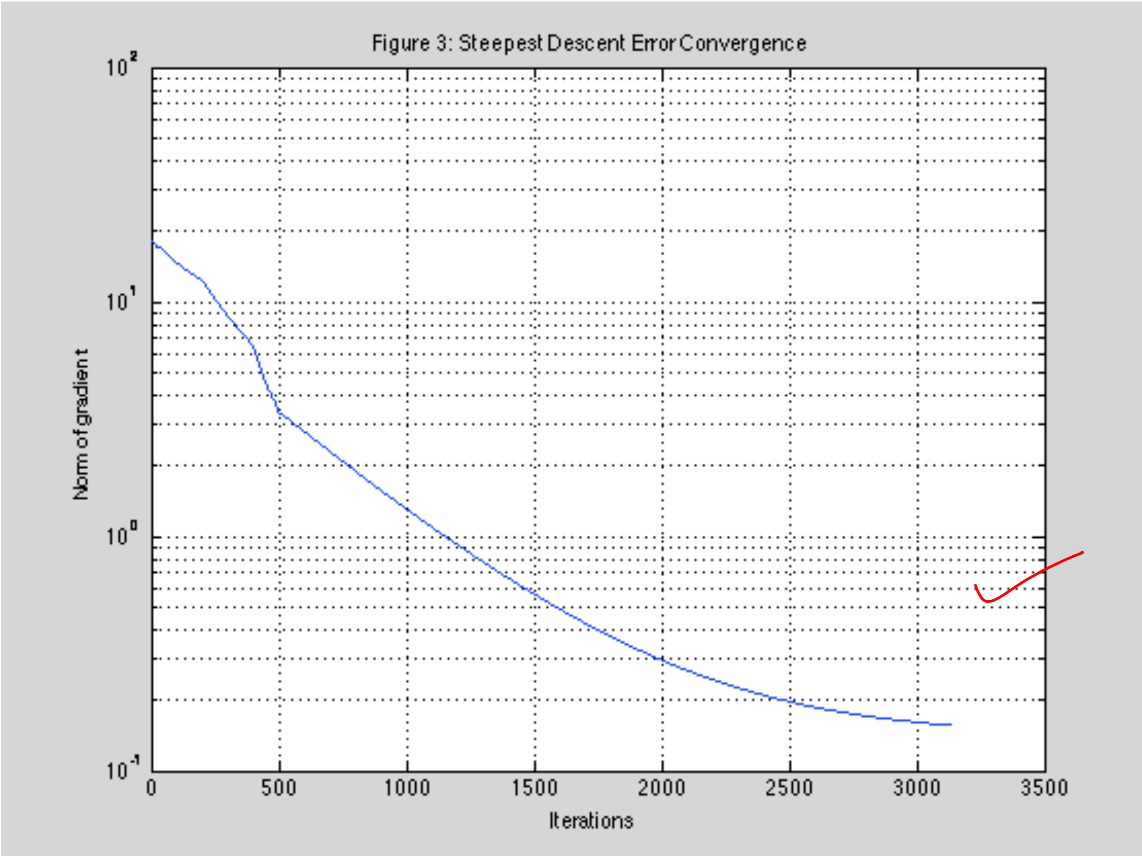
Solution



Gradient Convergence



Error Convergence

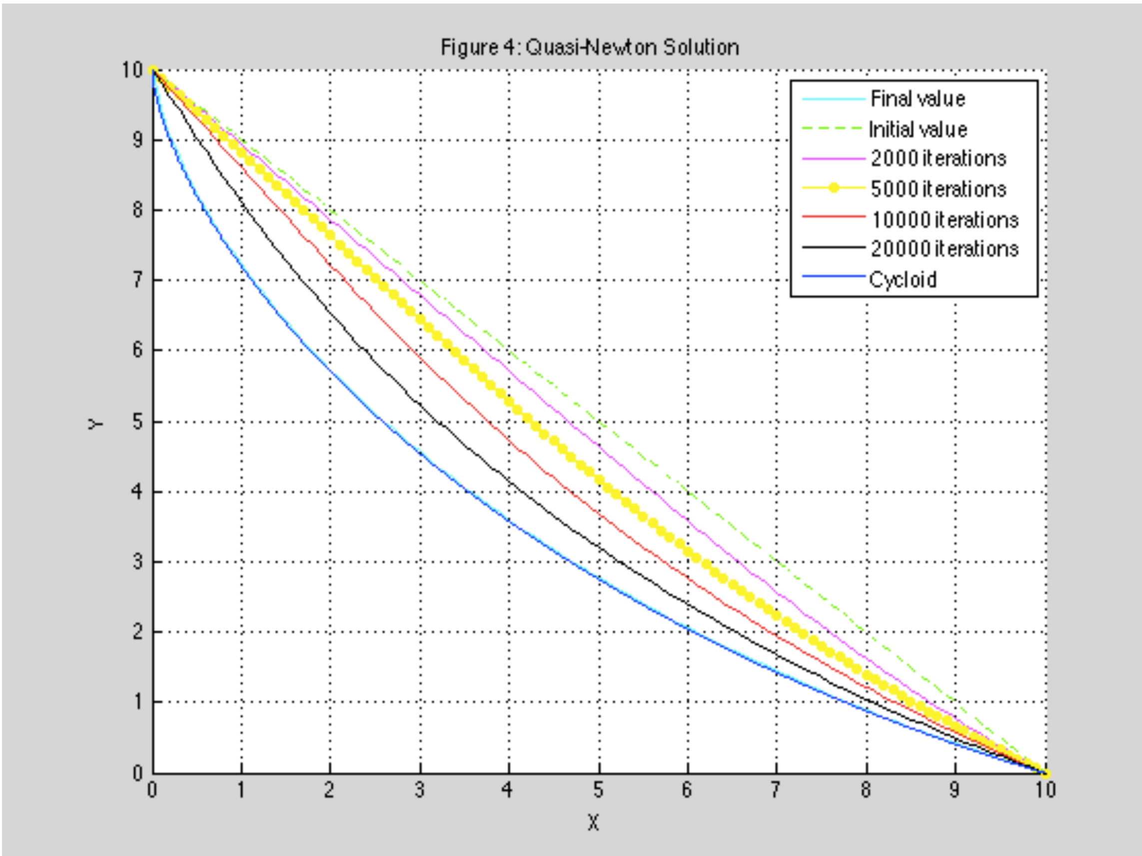


Iterations to convergence: 3137

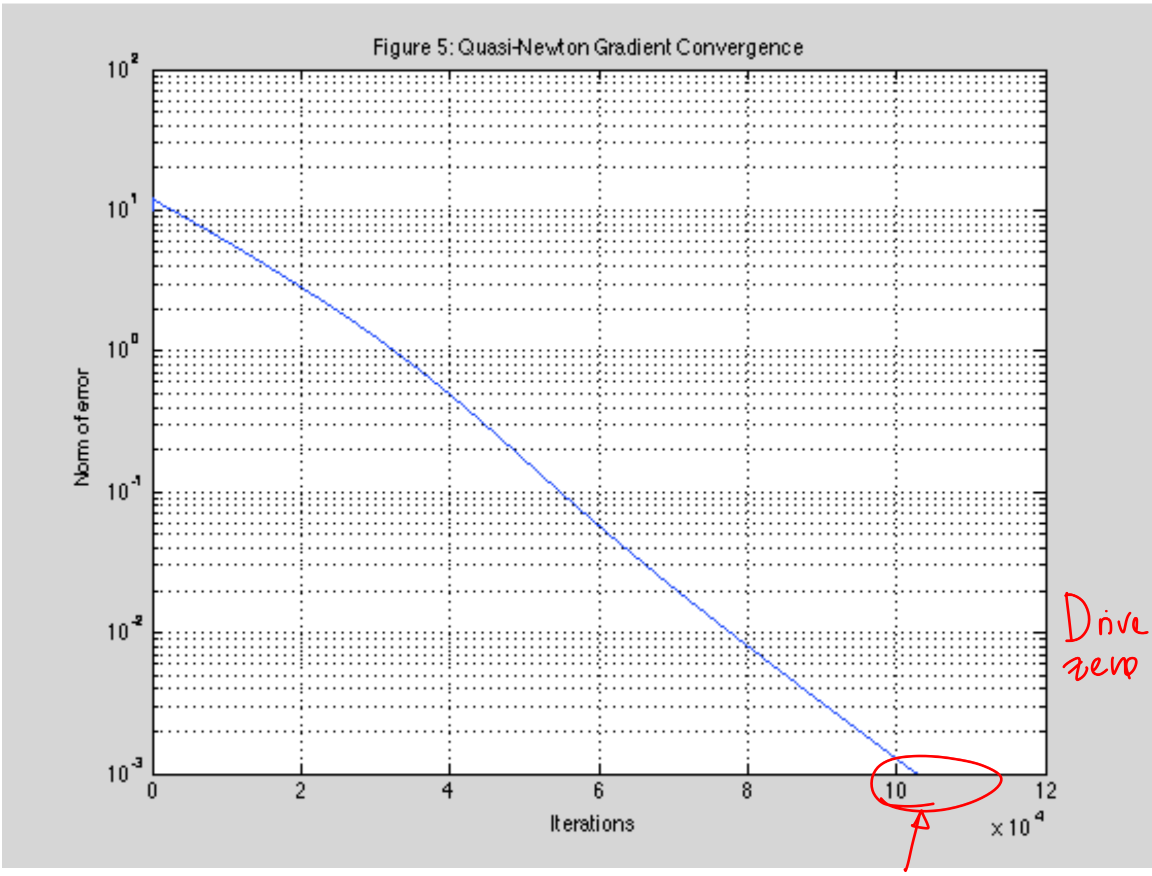
Note that the norm of the gradient was found by taking the L-2 norm of the 100x1 matrix containing the gradient at each mesh point. This was done for each iteration to obtain Figure 2. Likewise, the norm of the error was found by taking the L-2 norm of the 100x1 matrix containing the difference between the value of y and the cycloid value at each mesh point. This was done for each iteration to obtain Figure3.

Part 2: Quasi-Newton

Solution



Gradient Convergence

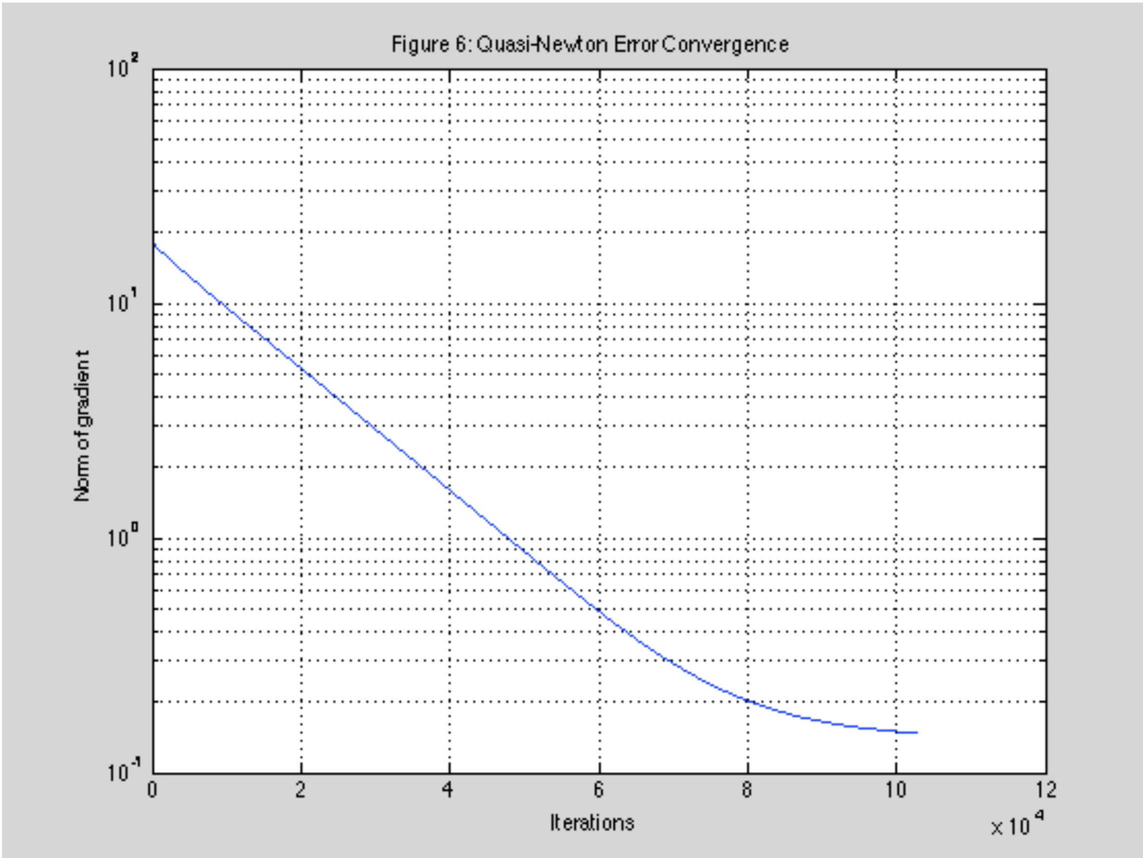


Drive it to machine zero in the future.

-3

Based on the previous page solution, it looks as though it still have not converged after 20K iterations compared to the 10K shown here.
What isn't clear is whether the implementat of the quasi-Newton is correct
Did you try using a fixed time step?

Error Convergence



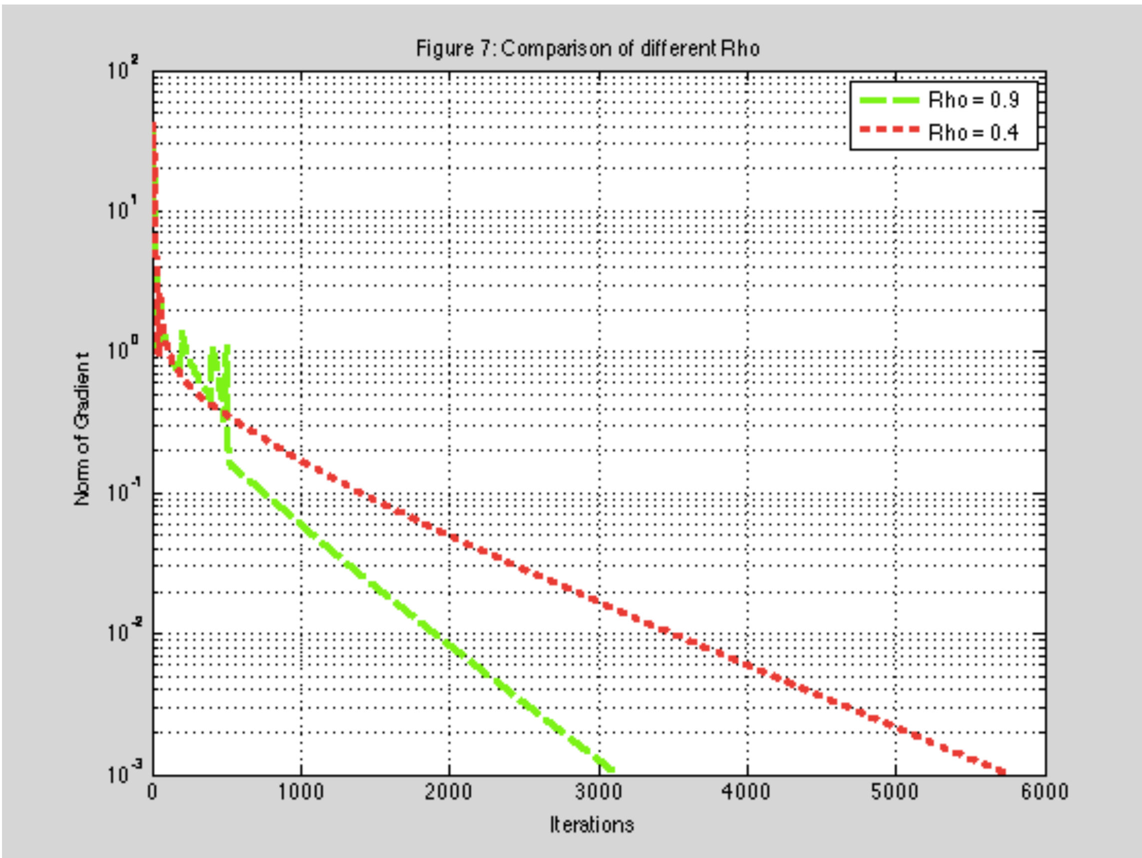
Iterations to convergence: 102801

Note that the norm of the gradient was found by taking the L-2 norm of the 100x1 matrix containing the gradient at each mesh point. This was done for each iteration to obtain Figure 2. Likewise, the norm of the error was found by taking the L-2 norm of the 100x1 matrix containing the difference between the value of y and the cycloid value at each mesh point. This was done for each iteration to obtain Figure 3.

Unfortunately, the Quasi-Newton method took an abnormally long time to converge for the Brachistochrone problem. It is worth noting that the intermediate shapes during optimization for the Quasi-Newton method were different from the ones obtained using Steepest Descent. Whereas SD began its convergence toward the top-left corner, the QN method had a more evenly distributed convergence, although it is initially focused in the lower right corner.

Part 3: Parameter Variation

Changing ρ to 0.4

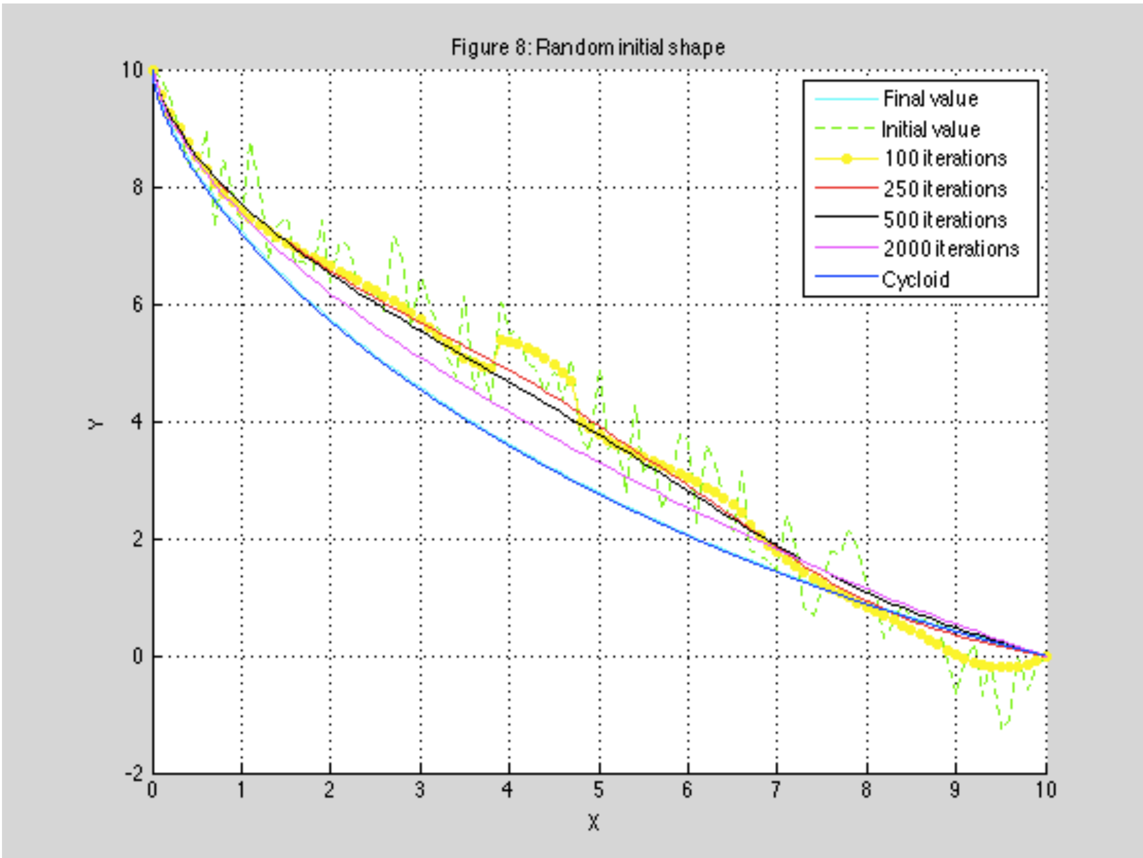


ρ = 0.9:
3136 iterations

ρ = 0.4:
5768 iterations

This is to be expected, because a ρ closer to 1.0 allows a finer approximation to the largest value of α that guarantees a sufficient decrease. Making ρ too small can cause a slower convergence by decreasing α too much.

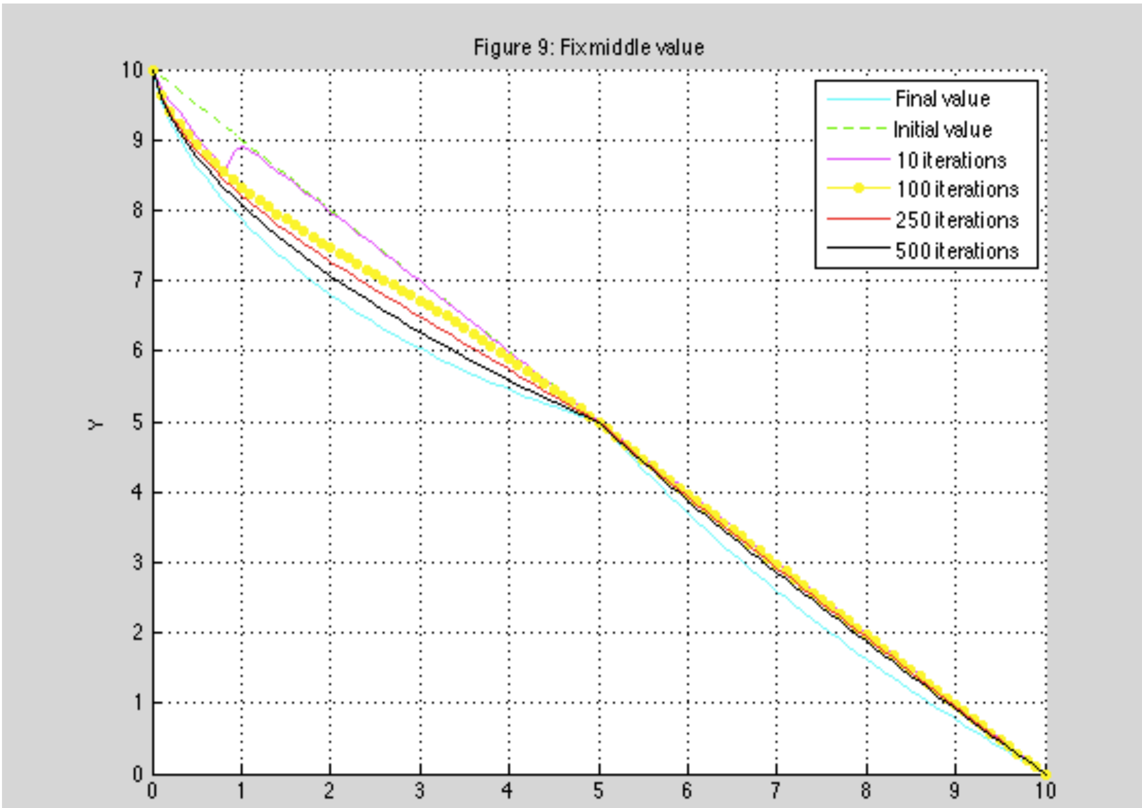
Changing initial shape



It took 15841 iterations to converge given the above initial shape. That being said, random numbers are used to obtain the initial shape of the green line, so the number of iterations can vary but is generally around 16000.

The final shape is once again a cycloid.

Fixing 1 point



How did you achieve this? Did you fix the point through a constraint or was it artificially forced to be at that point.

Keeping the middle point fixed produces predictable results. Rather than having one large cycloid between (0,10) and (10,0), we obtained two smaller cycloids, one between (0,10) and (5,5), and another between (5,5) and (10,0).

Conclusion

Both the Steepest Descent and the DFP Quasi-Newton method converge to the shape of a cycloid, which also happens to be the exact solution to the Brachistochrone problem. The SD method had a much faster convergence than the QN method, contrary to the result of minimizing the Rosenbrock function. Varying the initial shape from a straight line had no effect on the final value, although it did slow down convergence. Keeping a point fixed split the problem into multiple regions, each of whose solutions converged to a cycloid.

Appendix

Matlab Code

```
function test8()
% DFP/BFGS Quasi-Newton Final Code
clearvars
clc

fin = 10;
mesh = 100;
deltaX = fin/mesh;
[C,D,th]=centroid;

rho = 0.9;
c = 10^(-4);
alpha = 0.00005;
limit = 101;

X = 0:deltaX:fin;
Y = fin:(-deltaX):0;
Z = Y;
W = Y;
lim = 120000;
m = 1;
Gra = ones(1,limit-2);
Lra = ones(lim-1,1);
Tra = ones(lim-1,1);
DnX=ones(limit-2,3);
u=0;
while(1)
    u=u+1;

    if u==limit-1
        break
    end
        now=X(u+1);
[row,column] = find(D>now-0.001 & D<now+0.001);
    DnX(u)=column(1);

end
DnX=DnX(:,1,1);

H_old = Y;
H_new = Y;
for u=1:size(Y,2)
    H_old(u) = 1;
    H_new(u) = 1;
end

% big loop: real number of iterations
while(1)

    m = m+1;
    Lra(m-1)=norm(Gra,2);
```

```
  if m==lim
      m
      break
  end
      if norm(Gra,2)<0.001
      m
      break
  end
  k = 1;
      if m==5000
      L=Y;
  elseif m==10000
      O=Y;
  elseif m==20000
      P=Y;
  elseif m==2000
      S=Y;
  end
  H_old=H_new;
  Z=Y;
  % small loop: goes from 1 to 100, i.e. each point of the line
  while(1)
  k = k+1;
  if k==limit
      break
  end

  dy = (Y(k+1)-Y(k-1))/(2*deltaX);
  ddy = (Y(k+1)-2*Y(k)+Y(k-1))/(deltaX^2);
  Gee = G(Y(k),dy,ddy);
  p_k = -H_old(k)*Gee;
  Gra(k-1)=Gee;

  %backtracking
alpha=1;
u=0;
while(1)
    u=u+1;
    if u==100
        break
    end
   if Y(k)+alpha*p_k> Y(k)+alpha*Gee*c*p_k
       alpha=alpha*rho;
   else
       break
   end

end

  deltaD = alpha*p_k;
      Y(k) = Z(k) + deltaD;

  deltaG = G(Y(k),dy,ddy) - G(Z(k),dy,ddy);

  % DFP
  H_new(k) = H_old(k) + ((deltaD * deltaD)/(deltaD * deltaG)) - ...
     (H_old(k) * deltaG * deltaG * H_old(k) )/(deltaG * H_old(k) *
```

```
deltaG);

    %  BFGS
    %H_new(k) = H_old(k) + (1 + (deltaG * H_old(k) *
deltaG)/(deltaG*deltaD))*...
    %  ((deltaD*deltaD)/(deltaD*deltaG)) -
(H_old(k)*deltaG*deltaD+H_old(k)*...
    % deltaG*deltaD)/(deltaG*deltaD);

    % Rank 1 formula:
    % H_new(k)=H_old(k)+(deltaD-H_old(k)*deltaG)/deltaG;


    % Should p_k = -Gee?

end
Tra(m-1)=norm(Y(2:100)-fliplr(C(DnX)));
end

hold on
grid on
plot(-X+10,-Y+10,'c')
plot(-X+10,-W+10,'g--')
plot(-X+10,-S+10,'m')
plot(-X+10,-L+10,'y.-')
plot(-X+10,-O+10,'r')
plot(-X+10,-P+10,'black')

[C,D]=centroid;

plot(D,-C+10)
xlabel('X')
ylabel('Y')
title('Figure 4: Quasi-Newton Solution')
legend('Final value','Initial value','2000 iterations','5000
iterations','10000 iterations','20000 iterations','Cycloid')

figure
semilogy(1:m-2,Lra(1:m-2))
grid on
xlabel('Iterations')
ylabel('Norm of error')
title('Figure 5: Quasi-Newton Gradient Convergence')

figure
semilogy(1:m-2,Tra(1:m-2))
grid on
xlabel('Iterations')
ylabel('Norm of gradient')
title('Figure 6: Quasi-Newton Error Convergence')

end

function [A,B,theta] = centroid()
    maxthet = 2.412011143913525342526482065 7;
    theta = 0:0.0001:maxthet;
```

```
    %theta = 0:0.0239:maxthet+0.0239;
    c2=11.45834075;
    A = 0.5*c2*(1-cos(theta));
    B = 0.5*c2*(theta-sin(theta));
end

function G1 = G(y,y_1,y_2)

    G1 = (-1)*(1+y_1^2+2*y*y_2)/(2*((y*(1+y_1^2))^(3/2)));

end
```