Assignment 0

Name: Julien Neves
Name: Matthieu Ranger
Date: January 30, 2017

Problem 1

4.1 - Derivative-Free Methods

The naive idea is to "bracket around" the acceptable region in optimization. This type of method requires no derivatives of the objective function and is robust (guarantees success under wide conditions).

Golden Search is one example, where we pick values (two per dimension), split the "boxed interval" in two parts (per dimension) and repeat the procedure in the region where the function is evaluated to a greater value.

Choosing a point to evaluate **F** is generally done by selecting $x_i = a + a_i(b-a)$ where

$$a_1 = \frac{3-\sqrt{5}}{2}$$

and

$$a_2 = \frac{\sqrt{5}-1}{2}$$

These values come from the criteria where the length of the new interval is independent of the lower or upper interval being chosen, and the ability to reuse interior points of previous iterations.

The **Nelder-Mead** method iterates over a simplex formed from the evaluation of **F** (starting with an initial guess). At each iteration, the point with the lowest value in the simplex (a corner) will reflect the opposite edge of the simplex (by twice the distance from the vertex to the opposing edge), prompting a search over the formed triangle outside the simplex for function values greater than ones in the simplex. If one is found, the simplex is expanded and we iterate.

If none are found, we contract the simplex by halving the distance between the point and its opposing edge, and shrink the simplex towards the best point if this new point is no better than the second worst.

4.2 - Newton-Rhapson Method

Newton-Raphson uses quadratic approximations of **F** to compute the root of the gradient (hence finding a max/minima). The updating rule is

$$x^{(k+1)} \leftarrow x^{(k)} - [f''(x^{(k)})]^{-1} f'(x^{(k)}$$

which is a result from second order Taylor approximations of **F** at $x^{(k)}$, when solving the first order condition. Note that Newton-Raphson requires the

function being twice differentiable (or double differentiation being numerically approximable, at least). The method is well-behaved on globally concave/vex functions, but may get stuck in "flat" areas if an initial guess is too far from a local max/minima.

4.3 - Quasi-Newton Methods

The Quasi-Newton methods employ the same strategy as the Newton method, but restrict the Hessian matrix to be Negative-definite approximation. This insures that the value function will increase at every step.

Like the Newton method, we get the that the direction we should take to optimize our function is given by

$$d^{(k)} = -B^{(k)} f'(x^{(k)})$$

where $B^{(k)}$ is the approximation to the Hessian $f'(x^{(k)})$ is the gradient evaluated at $x^{(k)}$.

Then, the updating of $x^{(k)}$ is given by the following formula.

$$x^{(k+1)} = x^{(k)} - s^{(k)} d^{(k)}$$

where $s^{(k)}$ is scaling factor for the $d^{(k)}$. If $B^{(k)}$ is the actual Hessian matrix and $s^{(k)}$ is set equal to one, it is easy to see that we are back at the original Netwon method.

Now, the choice of $B^{(k)}$ is important in terms of efficiency. The easiest choice for $B^{(k)}$ is to set it equal to the negative identity matrix $-I$. This insures the that $B^{(k)}$ is negative definite, but at the same time this is not the most optimal way to go about it. This choice of $B^{(k)}$ let's $d^{(k)} = f'(x^{(k)})$ which is the equivalent to having a Newton-step in the direction of the gradient. This method is now as the method of steepest ascent.

Note that to satisfy the Quasi-Newton method, the matrix $B^{(k)} f'(x^{(k)})$ is require to satisfy two conditions:

1. Quasi-Newton Condition:

$$d^{(k)} = B^{(k+1)}[f'(x^{(k)} + d^{(k)}) - f'(x^{(k)})]$$

2. The inverse $B^{(k)}$ needs to be both symmetric and negative definite

Two methods that satisfy this are the Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfard-Shano (BFGS) algorithms.

For the DFP algorithm, $B^{(k)}$ is updated in the following way:

$$B^{(k+1)} = B^{(k)} + \frac{dd'}{d'u} - \frac{B^{(k)} dd' B^{(k)}}{u' B^{(k)} u}$$

where $d = x^{(k+1)} - x^{(k)}$ and $u = f'(x^{(k+1)}) - f'(x^{(k)})$

For the BFGS algorithm, $B^{(k)}$ is updated in the following way:

$$B^{(k+1)} = B^{(k)} + \frac{1}{d'u}\left(wd' + dw' - \frac{w'u}{d'u}dd'\right)$$

where $d = x^{(k+1)} - x^{(k)}$, $u = f'(x^{(k+1)}) - f'(x^{(k)})$ and $w = d - B^{(k)}u$

BFGS is known to be superior to the DFP algorithm which both perfom better than the steepest ascent method usually.

One issue that the Quasi-Newton methods have is that due to the possible explosive nature of $\frac{1}{d'u}$. One way to prevent for this is to test that the following

$$|d'u| < \epsilon||d||||u||$$

if it doesn't hold, we can either stop updating $B(k)$ or reset it to a negative identity matrix.

4.4 - Line Search Methods

Like previously stated we might want to change our s in our Quasi-Newton methods to improve our search. For example the golden search method is an example of linear search method.

Other example of linear search methods are for example the Armijo search and the Goldstein search.

For the Arminjo search, our goal is to find the minimum power of j such that

$$\frac{f(x + sd) - f(x)}{s} \geq \mu f'(x)^T d$$

where $s = \rho^j$, $0 < \rho < 1$, and $0 < \mu < 0.5$. The basic goal is of this method is to start from a step-size of one, an backtrack until the slope on the lefthand side is a fraction μ of the slope on the righthand side.

For the Goldstein search, we find any values of s such that

$$\mu_0 f'(x)^T d \leq \frac{f(x + sd) - f(x)}{s} \leq \mu_1 f'(x)^T d$$

for some values of $0 < \mu_0 \leq 0.5 \leq \mu_1 < 0$. Note that in the case of Goldstein method, we only have a stopping rule and not a method of selecting candidates.

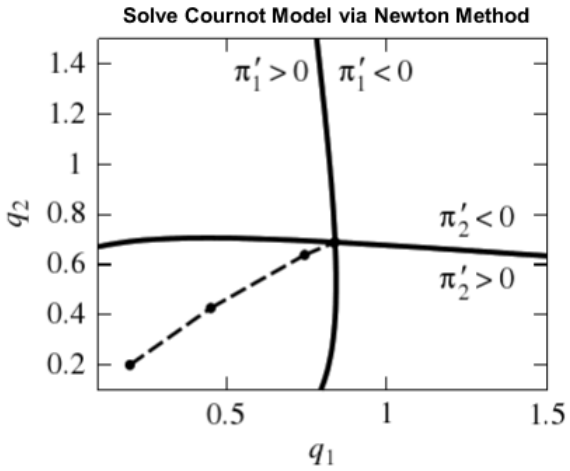
One way to select a candidate s in this case, is to double s until we find a point that is such that it respect the Goldstein criterion or that is such that $\mu_0 f'(x)^T d > \frac{f(x+sd)-f(x)}{s}$ which we can then backtrack to a point that respect the criterion.

Another method is to start with $s = 1$ and s is updated using the cubic approximation of the objective function until an appropriate point it found. Generaly this method will be fast, and better than other line search methods in most cases (especially if the function is smooth).

Finally the golden search, use the same procedure as described in 4.1 to find an step size by setting a bracket of possible values of s and reducing it until we are satisfied.

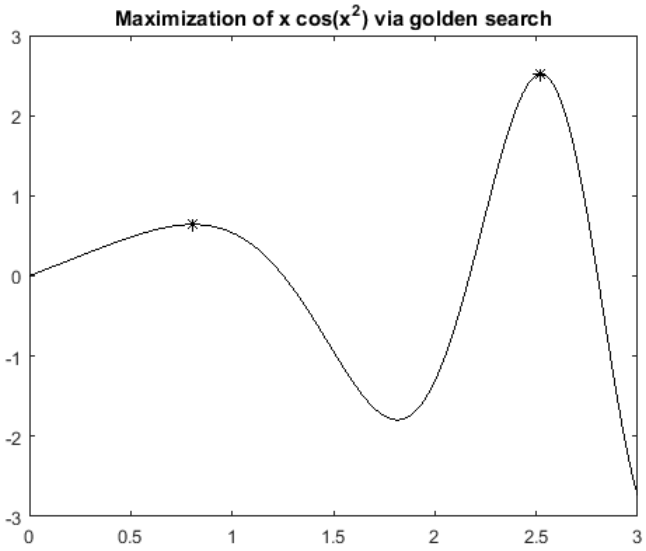
Problem 2

3.3



On this figure, we see an example of the Cournot problem where $\eta = 1.6$, $c_1 = 0.6$, and $c_2 = 0.8$. As we can see, the best response is that at the intersection of the reactions curves of both players. Using the Newton method and starting at $q_1 = q_2 = 0.2$, it takes roughly three steps to reach the equilibrium.

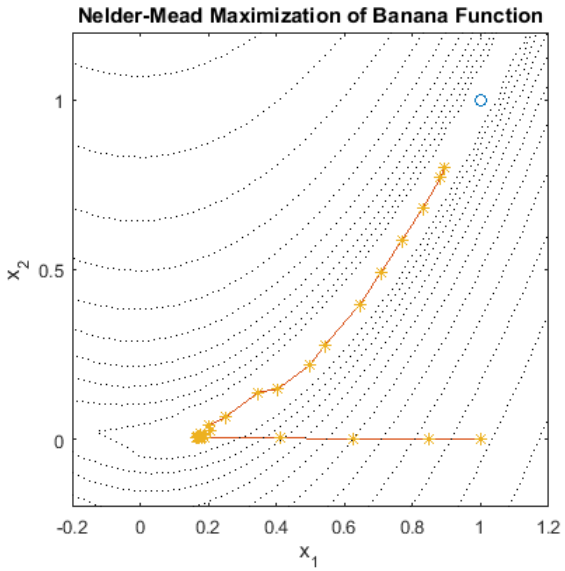
4.1



Using golden search, we maximizes the function $x\cos(x^2)$. With this algorithm, we reach $x = 0.8083$. It is straightforward to see that it is only a local

maximum and not a global one. In fact, at $x = 2.5$ we have a global maximum on $x = [0, 3]$. This is problem arises from the fact that the optimand is not concave

4.3



On this figure, we maximizes the banana function $f(x) = -100(x_2 - x_1^2)^2 - (1 - x_1)^2$ using the Nelder-Mead algorithm. It reaches its stopping point at $x = (1, 1)$ which is the global maximum. It takes the Nelder-Mead method 55 steps to reach this global maximum. The Nelder-Mead algorithm is based on evaluating the vertices of a simplex and then updating the simplex through reflection, expansion, contraction and shrinkage depending on the value of the vertices in order to converge to the maximum.

Appendix - Code

```
% DEMSLV12 Cournot demonstration
disp('DEMSLV12 Cournot demonstration')
close all

n = 50;
q1 = nodeunif(n,0.1,1.5);
q2 = nodeunif(n,0.1,1.5);

for i=1:n
for j=1:n
   z = cournot([q1(j);q2(i)]);
   z1(i,j) = z(1);
   z2(i,j) = z(2);
end
end

warning off

close all
figure(1)
axis([0.1 1.5 0.1 1.5])
[C,h] = contour(q1,q2,z1,[0 0]);
set(h,'LineStyle','-','LineWidth',2,'Tag','F1','edgecolor',[0 0 0],'CDataMapping','direct');
hold on
[C,h] = contour(q1,q2,z2,[0 0]);
set(h,'LineStyle','-','LineWidth',2,'Tag','F2','edgecolor',[0 0 0],'CDataMapping','direct');
text(.55,1.4,'\pi_1''>0');
text(.85,1.4,'\pi_1''<0');
text(1.2,.75,'\pi_2''<0');
text(1.2,.55,'\pi_2''>0');
title('Solve Cournot Model via Newton Method')
h=xlabel('q_1');
set(h,'VerticalAlignment','cap')
h=ylabel('q_2');
set(h,'VerticalAlignment','bottom')

q = [0.2;0.2];
optset('newton','maxit',1);
optset('newton','maxsteps',0);
for i=1:10
  qnew = newton('cournot',q);
  plot([q(1) qnew(1)],[q(2) qnew(2)],'-');
  plot(q(1),q(2),'*');
  q = qnew;
end
hold off

figure(2)
axis([0.1 1.5 0.1 1.5])
[C,h] = contour(q1,q2,z1,[0 0]);
set(h,'LineStyle','-','LineWidth',2,'Tag','F1','edgecolor',[0 0 0],'CDataMapping','direct');
hold on
[C,h] = contour(q1,q2,z2,[0 0]);
```

```
set(h,'LineStyle','-','LineWidth',2,'Tag','F2','edgecolor',[0 0 0],'CDataMapping','direct');
text(.55,1.4,'\pi_1''>0');
text(.85,1.4,'\pi_1''<0');
text(1.2,.75,'\pi_2''<0');
text(1.2,.55,'\pi_2''>0');
title('Solve Cournot Model via Broyden''s Method')
h=xlabel('q_1');
set(h,'VerticalAlignment','cap')
h=ylabel('q_2');
set(h,'VerticalAlignment','bottom')

q = [0.2;0.2];
optset('broydeni','maxit',1);
optset('broydeni','maxsteps',0);
fjac = fdjac('cournot',q);
fjacinv = inv(fjac);
for i=1:10
  [qnew,fval,fjacinv] = broydeni('cournot',q,fjacinv);
  plot([q(1) qnew(1)],[q(2) qnew(2)],'-');
  plot(q(1),q(2),'*');
  q = qnew;
end
hold off

warning on


prtfigs(mfilename,'Cournot Model Solved Using Newton''s Method',1)
prtfigs(mfilename,'Cournot Model Solved Using Broyden''s Method',2)


optset('newton','defaults')
optset('broydeni','defaults')
```

```
% DEMOPT01 Illustrates maximization via golden search
function demopt01
close all

disp(' ')
disp('DEMOPT01 Illustrates maximization via golden search')

x = nodeunif(500,0,3);

f = inline('x.*cos(x.^2)');

x1 = golden(f,0,1);
x2 = golden(f,2,3);
xx = golden(f,0,3);

close all
figure(1)
plot(x,f(x),'k',x1,f(x1),'k*',x2,f(x2),'k*')
title('Maximization of x cos(x^2) via golden search')

prtfigs(mfilename,'Maximization of x cos(x^2) via golden search',1)
```

```
% DEMOPT03 Demonstrates Nelder-Mead simplex method
% Creates and plays a movie of Nelder-Meade simplex iterations when
% maximizing banana function f(x,y)=-100*(y-x*x)^2-(1-x)^2, starting at [0;1].
% To view the movie again use
%   M=demopt03; movie(M);
function M=demopt03
close all

disp(' ')
disp('DEMOPT03 Demonstrates Nelder-Mead simplex method')

n = [20 20];
xmin = [-0.2 -0.2];
xmax = [ 1.2  1.2];
[x,xcoord] = nodeunif(n,xmin,xmax);
[x1,x2] = meshgrid(xcoord{1},xcoord{2}) ;

y = banana(x');
y = reshape(y,n(1),n(2))';
conts = -exp(0.25:0.5:20);

figure(1)
contour(xcoord{1},xcoord{2},y,conts,'k:')
xlabel('x_1'),ylabel('x_2')
title('Nelder-Mead Maximizes the Banana Function')

optset('neldmead','maxit',1);
k = 50;
x = [1;0];
warning off
[xx,S] = neldmead('banana',x);
warning on
hold on
hp = patch(S(1,:),S(2,:),[0.5 0.5 0.5]);
M  = moviein(k);
for i=1:k
  xvec(:,i) = x;
  warning off
  [x,S] = neldmead('banana',x,S);
  warning on
  set(hp,'xdata',S(1,:)','ydata',S(2,:)');
  M(:,i) = getframe;
end
hold off
optset('neldmead','defaults');

%for i=1:size(M,2),movie(M(:,i),0);end

figure(2)
plot(1,1,'o')
hold on
plot(xvec(1,:),xvec(2,:))
plot(xvec(1,:),xvec(2,:),'*')
contour(xcoord{1},xcoord{2},y,conts,'k:')
hold off
axis square
title('Solve Cournot Model via Newton Method')
```

```
h=xlabel('q_1');
set(h,'VerticalAlignment','cap')
h=ylabel('q_2');
set(h,'VerticalAlignment','bottom')
axis([-.2 1.2 -.2 1.2])
set(gca,'ytick',[0 0.5 1])

prtfigs(mfilename,'Nelder-Mead Maximization of Banana Function',2)
```