M368K Homework 8

§ 10.3 #2b[1],10b § 10.4 #1b[2] § 10.5 #1b[3],2c[3]

Hershal Bhave (hb6279)

March 27, 2013

1 § 10.3

1.1 2b[1]

Use Broyden's method with $\mathbf{x}^{(\mathbf{0})} = \mathbf{0}$ to compute $\mathbf{x}^{(\mathbf{2})}$ for the following nonlinear system, given $\mathbf{x}^{(\mathbf{0})} = \begin{pmatrix} 5 \\ 2 \\ 0 \end{pmatrix}$

$$\begin{aligned} x_1^2 + x_2 - 37 &= 0, \\ x_1 - x_2^2 - 5 &= 0, \\ x_1 + x_2 + x_3 - 3 &= 0. \end{aligned}$$

Using the algorithm specified in `broyden.m` (listing 1), I achieved the following:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\Vert\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\Vert_2$
0	5	2	0	
1	6.0732	1.2683	-4.3415	4.5316
2	5.9790	1.1178	-4.0967	0.30241
3	6.0040	1.0413	-4.0453	0.095495
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
∞	6	1	-4	$\lim_{k\to\infty} = 0$

```
#!/usr/bin/octave
# Created by Hershal Bhave on 3/25/13
# For M368K HW8,  10.3 Number 2b
# Written in GNU Octave
#
# Description: Uses the Broyden Algorithm to approximate the solution
# to the nonlinear system given an initial approximation vector x and
# an array of functions specified in a function handler.
#

function x = broyden(x, f, n)

  pkg load optim;

  tol = 10^-6;

  A = jacob(x,f);
  v = f(x);
  A = inv(A);
  s = -A*v;
  x = x+s;
  k=1;

  while(k<n && norm(s)>tol)
    w=v;
    v=f(x);
    y=v-w;
    z=-A*y;
    p=-s'*z;
    u=(s'*A)';
    A=A+(1/p)*(s+z)*u';
    s=-A*v;
    x=x+s;
    k++;
  endwhile

endfunction
```

Listing 1: broyden.m

1.2 10b

By multiplying on the right by $A + \mathbf{x}\mathbf{y^t}$, show that when $\mathbf{y^t}A^{-1}\mathbf{x} \neq -1$ we have

$$(A + \mathbf{x}\mathbf{y^t})^{-1} = A^{-1} - \frac{A^{-1}\mathbf{x}\mathbf{y^t}A^{-1}}{1 + \mathbf{y^t}A^{-1}\mathbf{x}} \tag{1}$$

By right-multiplying the LHS with $A + \mathbf{x}\mathbf{y^t}$, we obtain

$$\begin{aligned}
I &= (A + \mathbf{x}\mathbf{y^t})A^{-1} - \frac{A^{-1}\mathbf{x}\mathbf{y}^tA^{-1}}{1 + \mathbf{y}^tA^{-1}\mathbf{x}} \\
&= AA^{-1} + \mathbf{x}\mathbf{y}^tA^{-1} - \frac{A^{-1}A\mathbf{x}\mathbf{y}^tA^{-1} + \mathbf{x}\mathbf{y}^tA^{-1}\mathbf{x}\mathbf{y}^tA^{-1}}{1 + \mathbf{y}^tA^{-1}\mathbf{x}} \\
&= I + \mathbf{x}\mathbf{y^t}A^{-1} - \frac{\mathbf{x}(1 + \mathbf{y^t}A^{-1}\mathbf{x})\mathbf{y^t}}{1 + \mathbf{y^t}A^{-1}\mathbf{x}} \\
&= I + \mathbf{x}\mathbf{y}^tA^{-1} - \mathbf{x}\mathbf{y}^tA^{-1} \\
I &= I.
\end{aligned}$$

The equality seems to hold, so we have shown that when $\mathbf{y^t}A^{-1}\mathbf{x} \neq -1$, Equation 1 holds true.

2 § 10.4

2.1 1b[2]

Use the method of Steepest Descent to approximate $\mathbf{x}^{(1)}$ given $\mathbf{x}^{(0)} = (1, 1.5)$, $g(x) = \|F(x)\|_2^2$ and $\alpha_0 = 1$ for the following nonlinear system:

$$F(\mathbf{x}) = \begin{cases} 3x_1^2 - x_2^2 = 0, \\ 3x_1x_2^2 - x_1^2 - 1 = 0. \end{cases}$$

$g(\mathbf{x})$ is given as follows:

$$g(\mathbf{x}) = f_1(x_1, x_2)^2 + f_2(x_1, x_2)^2 \tag{2}$$

Which implies its gradient is:

$$\nabla g(x_1, x_2) \equiv \nabla g(\mathbf{x}) = \begin{pmatrix} 2f_1(\mathbf{x})\frac{\partial f_1}{\partial x_1}(\mathbf{x}) + 2f_2(\mathbf{x})\frac{\partial f_2}{\partial x_1}(\mathbf{x}) \\ 2f_1(\mathbf{x})\frac{\partial f_1}{\partial x_2}(\mathbf{x}) + 2f_2(\mathbf{x})\frac{\partial f_2}{\partial x_2}(\mathbf{x}) \end{pmatrix} \tag{3}$$

For $\mathbf{x}^{(0)} = (1, 1.5)$, we have

$$g(\mathbf{x}^{(0)}) = 14.625, \quad \nabla g(\mathbf{x^{(0)}}) = \begin{pmatrix} 44.625 \\ 63.000 \end{pmatrix}, \text{ and } \quad z_0 = \|\nabla g(\mathbf{x^{(0)}})\|_2 = 77.204.$$

Now we will find a normalized $\mathbf{z}$. Let

$$\mathbf{z} = \frac{1}{z_0}\nabla g(\mathbf{x}^{(0)}) = \begin{pmatrix} 0.57802 \\ 0.81602 \end{pmatrix}$$

We are given that $\alpha_0 = 0$ so we will skip the generation of an interpolating polynomial. All we have to do now is simply

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha_0 \mathbf{z} \tag{4}$$

Using Equation 4 we obtain

$$\mathbf{x}^{(1)} = \begin{pmatrix} 1 \\ 1.5 \end{pmatrix} - \begin{pmatrix} 0.57802 \\ 0.81602 \end{pmatrix}$$

Which gives us our final answer

$$\boxed{\mathbf{x}^{(1)} = \begin{pmatrix} 0.42198 \\ 0.68398 \end{pmatrix}}$$

```
#!/usr/bin/octave
# Created by Hershal Bhave on 3/25/13
# For M368K HW8,  10.4 Number 1b
# Written in GNU Octave
#
# Description: Uses the Steepest Descent Method to minimize the
# function f given an initial approximation x and the maximum number
# of steps n
#

function x = steepest(x, f, n)

  tol = 10^-6; k=1;
  gg=@(x)(sum(f(x).^2));

  while k<n
    g(2)=gg(x);
    z=2*jacob(x,f)'*f(x);
    z0=norm(z);
    z=z/z0;

    alpha(2)=0;
    alpha(4)=1;

    g(4)=gg(x-alpha(4).*z);

    while g(4)>g(2)
      alpha(4)=alpha(4)/2;
      g(4)=gg(x-alpha(4).*z);
      if alpha(4)<tol/2
  fprintf("No likely improvement\n");
  k=n;
      endif
    endwhile

    alpha(3)=alpha(4)/2
    g(3)=gg(x-alpha(3).*z)

    h1=(g(3)-g(2))/alpha(3)
    h2=(g(4)-g(3))/(alpha(4)-alpha(3))
    h3=(h2-h1)/(alpha(4))

    alpha(1)=0.5*(alpha(3)-h1/h3);

    g(1)=gg(x-alpha(1).*z);
    [gmin,minidx] = min(g);

    x=x-alpha(minidx).*z;

    if abs(gmin-g(2)) < tol
      k=n;
    endif
    k++;
  endwhile

endfunction
```

Listing 2: steepest.m

3 § 10.5

3.1 1b[3]

The nonlinear system

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 - x_2^2 + 2x_2 = 0 \\ f_2(x_1, x_2) &= 2x_1 + x_2^2 - 6 = 0 \end{aligned}$$

has two solutions,

$$\mathbf{x}^{(1)} = \begin{bmatrix} 0.625204094 \\ 2.179355825 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^{(2)} = \begin{bmatrix} 2.109511920 \\ -1.334532188 \end{bmatrix}$$

Use the continuation method and Euler's method with $N = 2$ to approximate the solutions where $\mathbf{x}(0) = (1, 1)^t$ and identify which of the two known solutions the continuation curve is approaching.

Using the algorithm for Euler's Method of order four and $N = 2$ described in in `eulers.m` (listing 3), I was able to obtain the following:

$$\boxed{\begin{array}{c} \mathbf{x}^* = \begin{cases} 0.42105 \\ 2.61842 \end{cases} \\ \text{The solution appears to be approaching } \mathbf{x}^{(1)}. \end{array}}$$

```
#!/usr/bin/octave
# Created by Hershal Bhave on 3/25/13
# For M368K HW8,  10.4 Number 1b
# Written in GNU Octave
#
# Description: Uses Euler's method to approximate the solution to the
# function f given an initial approximation x and number of steps n
#

function x = eulers(x, f, N)

  b = -(1/n)*f(x);
  for i=1:N
    A = jacob(x,f);
    x = x+A\b;
  endfor
endfunction
```

Listing 3: `eulers.m`

3.2 $2c^3$

The nonlinear system

$$f_1(x_1, x_2) = x_1^2 - x_2^2 + 2x_2 = 0$$
$$f_2(x_1, x_2) = 2x_1 + x_2^2 - 6 = 0$$

has two solutions,

$$\mathbf{x}^{(1)} = \begin{bmatrix} 0.625204094 \\ 2.179355825 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^{(2)} = \begin{bmatrix} 2.109511920 \\ -1.334532188 \end{bmatrix}$$

Use the Runge-Kutta method of order four with order four with $N = 1$ to approximate the solutions where $\mathbf{x}(0) = (3, -2)^t$ and identify which of the two known solutions the continuation curve is approaching.

Using the algorithm for Runge-Kutta of order four and $N = 1$ described in in `rungekutta.m` (listing 4), I was able to obtain the following:

$$\boxed{\begin{array}{c} \mathbf{x}^* = \begin{cases} 2.1094 \\ -1.3346 \end{cases} \\ \text{The solution appears to be approaching } \mathbf{x}^{(2)}. \end{array}}$$

```
#!/usr/bin/octave
# Created by Hershal Bhave on 3/25/13
# For M368K HW8,  10.5 Number 1b
# Written in GNU Octave
#
# Description: Uses the Runge-Kutta Method of order 4 to approximate
# the solution to the function f given an initial approximation x
#

function x = rungekutta(x, f, N)

  h = 1/N;
  b = -h*f(x);

  if size(x,1) < size(x,2)
    x=x';
  endif

  for i=1:N
      A = jacob(x,f);
      k(:,1) = A\b;

      A = jacob(x+1/2.*k(:,1),f);
      k(:,2) = A\b;

      A = jacob(x+1/2.*k(:,2),f);
      k(:,3) = A\b;

      A = jacob(x+k(:,3),f);
      k(:,4) = A\b;

      x = x + (k(:,1)+2*k(:,2)+2*k(:,3)+k(:,4))/6;
  endfor

endfunction
```

Listing 4: `rungekutta.m`

4 Programming Minilab

The solutions generally converge to the same point.