MATH 151B - Applied Numerical Methods - Homework 6

Darren Tsang, 405433124, Discussion 1A

Produced on Sunday, September 13, 2020 @ 02:34:03 AM

Question 1

We begin by finding:

$$F(1,1,1) = \begin{bmatrix} 1^2+1-37 \\ 1-1^2-5 \\ 1+1+1-3 \end{bmatrix} = \begin{bmatrix} -35 \\ -5 \\ 0 \end{bmatrix}$$

The Jacobian is:

$$J(x_1,x_2,x_3) = \begin{bmatrix} 2x_1 & 1 & 0 \\ 1 & -2x_2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

At the initial point:

$$J(1,1,1) = \begin{bmatrix} 2 & 1 & 0 \\ 1 & -2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

The inverse:

$$J^{-1}(1,1,1) = \begin{bmatrix} .4 & .2 & 0 \\ .2 & -4 & 0 \\ -.6 & .2 & 1 \end{bmatrix}$$

We do matrix multiplication:

$$J^{-1}(1,1,1)F(1,1,1) = \begin{bmatrix} -15 \\ -5 \\ 20 \end{bmatrix}$$

Thus, $x^{(1)}$ is:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -15 \\ -5 \\ 20 \end{bmatrix} = \begin{bmatrix} 16 \\ 6 \\ -19 \end{bmatrix}$$

From earlier, $F(1,1,1) = [-35,-5,0]$, we find $F(16,6,-19) = [225,-25,0]$. Now, we see that:

$$\sqrt{35^2+5^2+0^2} = \sqrt{1250} < \sqrt{225^2+25^2+0^2} = \sqrt{51250}$$

This means that our new approximation is not better than the starting guess because the result of $F(x^{(1)})$ is further than $F(x^{(0)})$ is to $[0,0,0]$.

```python
def newton(F_all, J, start, k):
  x = [start]
  e = [np.linalg.norm(F_all(start))]
  for i in range(0, k):
    temp = x[i] - np.dot(inv(J(x[i])),F_all(x[i]))
    x.append(np.around(temp, 10))
    e.append(np.linalg.norm(F_all(temp)))
  return(x, e)
```

```python
def F_all(x):
  return([[x[0][0]*x[0][0] + x[1][0] - 37],
          [x[0][0] - x[1][0]*x[1][0] - 5],
          [x[0][0] + x[1][0] + x[2][0] - 3]])

def J(x):
  return([[2*x[0][0], 1, 0],
          [1, -2*x[1][0], 0],
          [1, 1, 1]])
```

```python
x, e = newton(F_all, J, [[1], [1], [1]], 4)
pd.DataFrame(data={'[x_1, x_2, x_3]':x, 'error':e})
```

```
##                                      [x_1, x_2, x_3]       error
## 0                                    [[1], [1], [1]]   35.355339
## 1                          [[16.0], [6.0], [-19.0]]  226.384628
## 2  [[9.0519480519], [3.3376623377], [-9.3896103896]]   48.793002
## 3  [[6.4654261642], [1.8883599911], [-5.3537861553]]    7.012088
## 4  [[6.0005806085], [1.2091137511], [-4.2096943596]]    0.509469
```

```python
x, e = newton(F_all, J, [[1], [1], [1]], 8)
pd.DataFrame(data={'[x_1, x_2, x_3]':x, 'error':e})
```

```
##                                      [x_1, x_2, x_3]         error
## 0                                    [[1], [1], [1]]  3.535534e+01
## 1                          [[16.0], [6.0], [-19.0]]  2.263846e+02
## 2  [[9.0519480519], [3.3376623377], [-9.3896103896]]  4.879300e+01
## 3  [[6.4654261642], [1.8883599911], [-5.3537861553]]  7.012088e+00
## 4  [[6.0005806085], [1.2091137511], [-4.2096943596]]  5.094688e-01
## 5  [[5.9985434542], [1.0174805783], [-4.0160240324]]  3.672327e-02
## 6   [[5.999988146], [1.0001443352], [-4.0001324812]]  3.005526e-04
## 7    [[5.9999999992], [1.00000001], [-4.0000000092]]  2.083024e-08
## 8                            [[6.0], [1.0], [-4.0]]  0.000000e+00
```

Question 3

```python
def steepest(g, grad_g, max_iter, initial, tol):
  x = [initial]
  i = 0
  while i <= max_iter:
    g1 = g(x[i])
    z = grad_g(x[i])
    z0 = np.linalg.norm(z)
    if z0 == 0:
      print("here1")
      return(x)
    z = z/z0
    alpha1 = 0
    alpha3 = 1

    g3 = g(x[i] - alpha3*z)

    while(g3 >= g1):
      alpha3 = alpha3/2
      g3 = g(x[i] - alpha3*z)
      if alpha3 < tol/2:
        print("here2")
        return(x)
    alpha2 = alpha3/2
    g2 = g(x[i] - alpha2*z)

    h1 = (g2 - g1)/alpha2
    h2 = (g3 - g2)/(alpha3 - alpha2)
    h3 = (h2 - h1)/alpha3

    alpha0 = .5*(alpha2 - h1)/h3
    g0 = g(x[i] - alpha0*z)
    if g0 <= g3:
      g_real = g0
      alpha = alpha0
    else:
      g_real = g3
      alpha = alpha3

    if abs(g_real - g1) < tol:
      print("here3")
      x.append(x[i] - alpha*z)
      return(x)
    else:
      x.append(x[i] - alpha*z)
    i = i + 1
  return(x)
```

```python
def F_all(x):
  x1 = x[0][0]
  x2 = x[1][0]
  x3 = x[2][0]
```

```
  return(np.array([[x1*x1*x1 + x1*x1*x2 - x1*x3 + 6],
          [math.exp(x1) + math.exp(x2) - x3],
          [x2*x2 - 2*x1*x3 - 4]]))

def J(x):
  x1 = x[0][0]
  x2 = x[1][0]
  x3 = x[2][0]
  return(np.array([[3*x1*x1 - 2*x1*x2 - x3, x1*x1, -x1],
          [math.exp(x1), math.exp(x2), -1],
          [-2*x3, 2*x2, -2*x1]]).tolist())


def g(x):
  x1 = x[0][0]
  x2 = x[1][0]
  x3 = x[2][0]
  f1 = (x1*x1*x1 + x1*x1*x2 - x1*x3 + 6)*(x1*x1*x1 + x1*x1*x2 - x1*x3 + 6)
  f2 = (math.exp(x1) + math.exp(x2) - x3)*(math.exp(x1) + math.exp(x2) - x3)
  f3 = (x2*x2 - 2*x1*x3 - 4)*(x2*x2 - 2*x1*x3 - 4)
  return(f1 + f2 + f3)

def grad_g(x):
  return(2 * inv(J(x)).dot(F_all(x)))
```

```
x_result = steepest(g, grad_g, math.inf, [[1], [1], [1]], .01)
```

```
## here3
```

```
F_x_result = []
error = []
for i in x_result:
  temp = F_all([ [i[0][0]], [i[1][0]], [i[2][0]] ])
  F_x_result.append(temp)
  error.append(np.linalg.norm(temp))

x_result = [np.around(x, 3) for x in x_result]
F_x_result = [np.around(x, 3) for x in F_x_result]
error = np.around(error, 5)

pd.DataFrame(data={'x':x_result, 'F(x_result)':F_x_result, 'error':error})
```

```
##                                 x                  F(x_result)    error
## 0                 [[1], [1], [1]]        [[7.0], [4.437], [-5.0]]  9.67900
## 1    [[0.686], [1.546], [1.777]]    [[5.832], [4.901], [-4.049]]  8.62669
## 2      [[0.517], [1.8], [2.729]]    [[5.208], [4.997], [-3.584]]  8.05845
## 3    [[0.427], [1.968], [3.711]]    [[4.852], [4.978], [-3.295]]  7.69337
## 4      [[0.371], [2.1], [4.701]]     [[4.596], [4.914], [-3.08]]  7.39960
## 5    [[0.333], [2.211], [5.694]]    [[4.386], [4.827], [-2.905]]  7.13963
## 6    [[0.305], [2.309], [6.688]]     [[4.202], [4.73], [-2.754]]  6.89978
## 7   [[0.188], [2.785], [12.102]]    [[3.825], [5.301], [-0.807]]  6.58650
## 8   [[0.161], [3.075], [17.742]]    [[3.221], [5.087], [-0.269]]  6.02696
```

```
## 9    [[0.143], [3.336], [24.54]]     [[2.563], [4.711], [0.112]]  5.36486
## 10  [[0.129], [3.575], [32.715]]     [[1.855], [4.118], [0.369]]  4.53163
## 11  [[0.117], [3.794], [42.424]]     [[1.092], [3.144], [0.472]]  3.36157
## 12  [[0.109], [3.971], [52.719]]     [[0.316], [1.449], [0.307]]  1.51471
## 13  [[0.107], [4.024], [57.371]]  [[-0.082], [-0.326], [-0.065]]  0.34218
## 14  [[0.107], [4.012], [56.371]]    [[-0.001], [-0.003], [-0.0]]  0.00337
## 15  [[0.107], [4.012], [56.355]]         [[0.0], [0.002], [0.0]]  0.00235
```

```
x_result = steepest(g, grad_g, math.inf, [[1], [1], [1]], 10**(-5))
```

```
## here3
```

```
F_x_result = []
error = []
for i in x_result:
  temp = F_all([ [i[0][0]], [i[1][0]], [i[2][0]] ])
  F_x_result.append(temp)
  error.append(np.linalg.norm(temp))

x_result = [np.around(x, 3) for x in x_result]
F_x_result = [np.around(x, 3) for x in F_x_result]
error = np.around(error, 5)

pd.DataFrame(data={'x':x_result, 'F(x_result)':F_x_result, 'error':error})
```

```
##                                x                     F(x_result)    error
## 0                 [[1], [1], [1]]        [[7.0], [4.437], [-5.0]]  9.67900
## 1    [[0.686], [1.546], [1.777]]    [[5.832], [4.901], [-4.049]]  8.62669
## 2      [[0.517], [1.8], [2.729]]    [[5.208], [4.997], [-3.584]]  8.05845
## 3    [[0.427], [1.968], [3.711]]    [[4.852], [4.978], [-3.295]]  7.69337
## 4      [[0.371], [2.1], [4.701]]     [[4.596], [4.914], [-3.08]]  7.39960
## 5    [[0.333], [2.211], [5.694]]    [[4.386], [4.827], [-2.905]]  7.13963
## 6    [[0.305], [2.309], [6.688]]     [[4.202], [4.73], [-2.754]]  6.89978
## 7   [[0.188], [2.785], [12.102]]    [[3.825], [5.301], [-0.807]]  6.58650
## 8   [[0.161], [3.075], [17.742]]    [[3.221], [5.087], [-0.269]]  6.02696
## 9    [[0.143], [3.336], [24.54]]     [[2.563], [4.711], [0.112]]  5.36486
## 10  [[0.129], [3.575], [32.715]]     [[1.855], [4.118], [0.369]]  4.53163
## 11  [[0.117], [3.794], [42.424]]     [[1.092], [3.144], [0.472]]  3.36157
## 12  [[0.109], [3.971], [52.719]]     [[0.316], [1.449], [0.307]]  1.51471
## 13  [[0.107], [4.024], [57.371]]  [[-0.082], [-0.326], [-0.065]]  0.34218
## 14  [[0.107], [4.012], [56.371]]    [[-0.001], [-0.003], [-0.0]]  0.00337
## 15  [[0.107], [4.012], [56.355]]         [[0.0], [0.002], [0.0]]  0.00235
```

As you can see above, there does not seem to be an increase as the tolerance gets smaller. The answer converges the same regardless of if the tolerance is .01 or .0001.