Assignment 9 TMA4135 Håvard Hjelmeseth

1

We have the following system of equations:

(i) $x^2 + y^2 = 4 \Rightarrow x^2 + y^2 - 4 = 0$

(ii) $xy = 1 \Rightarrow xy - 1 = 0$

Newtons method on systems is on the form

- Solve the system $J(x_k)\Delta_k = -f(x_k)$
- Let $x_{k+1} = x_k + \Delta_k$

The jacobian matrix of our system:

$$J(x) = \begin{pmatrix} \frac{\partial}{\partial x}(x^2+y^2) & \frac{\partial}{\partial y}(x^2+y^2) \\ \frac{\partial}{\partial x}(xy) & \frac{\partial}{\partial y}(xy) \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ y & x \end{pmatrix}$$

and we have

$$f(x) = \begin{pmatrix} x^2+y^2-4 \\ xy-1 \end{pmatrix} \quad \text{and} \quad x_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Giving us

$$f(x_0) = \begin{pmatrix} 2^2+0^2-4 \\ 2 \cdot 0 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

and the Jacobian matrix

$$J(x_0) = \begin{pmatrix} 2 \cdot 2 & 2 \cdot 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}$$

If we plug these into the expression

$$J(x_k)\Delta_k = -f(x_k)$$

We get

$$\begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -\begin{pmatrix} 0 \\ -1 \end{pmatrix} \Rightarrow \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$$

Giving us

$$x_1 = x_0 + \Delta x = 2 + 0 = 2$$
$$y_1 = y_0 + \Delta y = 0 + \tfrac{1}{2} = \tfrac{1}{2}$$
$$\Rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 2 \\ \frac{1}{2} \end{pmatrix}$$

Now using the same principles again, we get for $k=2$:

$$f(x_1) = \begin{pmatrix} 2^2 + (\frac{1}{2})^2 - 4 \\ 2 \cdot \frac{1}{2} - 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{4} \\ 0 \end{pmatrix}$$

$$J(x_1) = \begin{pmatrix} 2 \cdot 2 & 2 \cdot \frac{1}{2} \\ \frac{1}{2} & 2 \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ \frac{1}{2} & 2 \end{pmatrix}$$

$$\begin{pmatrix} 4 & 1 \\ \frac{1}{2} & 2 \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -\begin{pmatrix} \frac{1}{4} \\ 0 \end{pmatrix}$$

$$\Rightarrow \quad (\text{I}) \quad 4\Delta x + \Delta y = -\tfrac{1}{4}$$
$$(\text{II}) \quad \tfrac{1}{2}\Delta x + 2\Delta y = 0$$

Solving (II) for Δx:

$$(\text{II}) \quad \tfrac{1}{2}\Delta x + 2\Delta y = 0$$
$$\tfrac{1}{2}\Delta x = -2\Delta y$$
$$\Delta x = -4\Delta y$$

Putting this back into (I) gives us:

$$(\text{I}) \quad 4\Delta x + \Delta y = -\tfrac{1}{4}$$
$$4(-4\Delta y) + \Delta y = -\tfrac{1}{4}$$
$$-16\Delta y + \Delta y = -\tfrac{1}{4}$$
$$\Delta y = \frac{1}{4 \cdot 15}$$
$$\Delta y = 1/60$$

And plugging $\Delta y = 1/60$ into (II):

$$\Delta x = -4\Delta y$$
$$\Delta x = -4(1/60)$$
$$\Delta x = -1/15$$

This gives us

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}-1/15\\ 1/60\end{pmatrix} \Rightarrow \begin{pmatrix}x_2\\ y_2\end{pmatrix} = \begin{pmatrix}x_1+\Delta x\\ y_1+\Delta y\end{pmatrix} = \begin{pmatrix}2-1/15\\ 1/2+1/60\end{pmatrix} = \underline{\underline{\begin{pmatrix}1.93\\ 0.52\end{pmatrix}}}$$

Now using the same principles again, we get for $k=2$:

$$f(x_2) = \begin{pmatrix}1.93^2 + 0.52^2 - 4\\ 1.93\cdot 0.52 - 1\end{pmatrix} \approx \begin{pmatrix}-0.005\\ 0.004\end{pmatrix}$$

$$J(x_1) = \begin{pmatrix}2\cdot 1.93 & 2\cdot 0.52\\ 0.52 & 1.93\end{pmatrix} \approx \begin{pmatrix}3.86 & 1.04\\ 0.52 & 1.93\end{pmatrix}$$

$$\begin{pmatrix}3.86 & 1.04\\ 0.52 & 1.93\end{pmatrix}\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} \approx -\begin{pmatrix}-0.005\\ 0.004\end{pmatrix}$$

$$\Rightarrow \quad \begin{aligned}&(I) \quad 3.86\Delta x + 1.04\Delta y \approx -0.005\\ &(II) \quad 0.52\Delta x + 1.93\Delta y \approx 0.004\end{aligned}$$

We solve this set of equations, and get:

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} \approx \begin{pmatrix}-0.002\\ 0.003\end{pmatrix}$$

This gives us

$$\begin{pmatrix}x_3\\ y_3\end{pmatrix} = \begin{pmatrix}x_2+\Delta x\\ y_2+\Delta y\end{pmatrix} \approx \begin{pmatrix}1.93-0.002\\ 0.52+0.003\end{pmatrix} = \underline{\underline{\begin{pmatrix}1.928\\ 0.517\end{pmatrix}}}$$

This means that since

$$\begin{pmatrix}x_2\\ y_2\end{pmatrix} \approx \begin{pmatrix}x_3\\ y_3\end{pmatrix}$$

We have started converging towards the actual solution, and have reached a decent approximation of the solution.

b)

Using the function "newton_sys" in "NonLinearEquations.ipynb"

We find an approximation to the solution:

```
x0 = [2.0, 0.0]

def f(x):
    return array([x[0]**2 + x[1]**2 - 4,
                  x[0]*x[1] - 1])
def jac(x):
    J = array([[x[0]*2, x[1]*2],
               [x[1],   x[0]]])
    return J

newton_system(f, jac, x0, max_iter=10)
```

```
k =  0, x =  [2.0, 0.0]
k =  1, x =  [2.  0.5]
k =  2, x =  [1.933333333333333 0.516666666666667]
k =  3, x =  [1.931852741096439 0.517637054821929]
k =  4, x =  [1.931851652578934 0.517638090204244]
```

This also confirms that our approximation in a) was correct.

c)

Using "newton system" to solve the slightly perturbed system:

$$x^2 + y^2 = 2$$
$$xy = 1$$

With the same initial values:

```
x0 = [2.0, 0.0]

def f(x):
    return array([x[0]**2 + x[1]**2 - 2,
                  x[0]*x[1] - 1])
def jac(x):
    J = array([[x[0]*2, x[1]*2],
               [x[1],   x[0]]])
    return J

newton_system(f, jac, x0, max_iter=10)
```

```
k =  0, x =  [2.0, 0.0]
k =  1, x =  [1.5 0.5]
k =  2, x =  [1.25 0.75]
k =  3, x =  [1.125 0.875]
k =  4, x =  [1.0625 0.9375]
k =  5, x =  [1.03125 0.96875]
k =  6, x =  [1.015625 0.984375]
k =  7, x =  [1.0078125 0.9921875]
k =  8, x =  [1.00390625 0.99609375]
k =  9, x =  [1.001953125 0.998046875]
k = 10, x =  [1.0009765625 0.9990234375]
```

We see that the approximation now converges a lot more slowly, and the value it converges towards is also of course different.

Changing the method call a bit after the screenshot was taken, we see that it now takes 18 iterations to reach the chosen convergence tolerance, whilst in b) this only takes 4 iterations.

This is a result of the problem in a having a higher order of convergence.

2 a)

We recall that Simpson's rule is defined as

$$S[f](x_{i-1}, x_i) = \frac{h}{6}\left(f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)\right)$$

where $h = x_i - x_{i-1}$ and $x_{i-1/2} = \frac{x_{i-1} + x_i}{2}$

We can use that

$$\int_a^b S(x)\,dx = \sum_{i=1}^{M} \int_{x_{i-1}}^{x_i} S(x)\,dx$$

Now if we use equally spaced x_i's and low degree interpolation we partition the integration interval and obtain

$$\int_{x_{i-1}}^{x_i} f(x)\,dx = \frac{h}{6}\left[f(x_{i-1}) + 4(x_{i-1/2}) + f(x_i)\right]$$

Now summing over all sub-intervals we obtain,

$$S_M(f) = \frac{h}{6}\sum_{j=1}^{M}\left(f(x_{j-1}) + 4(x_{x_j - 1/2}) + f(x_j)\right)$$

This gives us the composite Simpson's rule:

$$\int_a^b f\,dx \approx CSR[f]([x_{i-1}, x_j]_{j=1}^{m}) = \frac{h}{6}\left[f(x_0) + 4f(x_{x_{1/2}}) + 2f(x_1)\right.$$
$$+ 4f(x_{3/2}) + 2f(x_2)$$
$$+ \ldots + 2f(x_{m-1})$$
$$\left. + 4f(x_{x_{m-1/2}}) + f(x_m)\right]$$

where:

$$x_j = a + j(b-a)/m \quad , \quad h = (b-a)/m$$

■

b)

Using the "simpson" function from "Quadrature.ipynb" (Numerical Integration

[1]:
```
%matplotlib inline

from numpy import *
from matplotlib.pyplot import *
from math import factorial
newparams = {'figure.figsize': (8.0, 4.0), 'axes.grid': True,
             'lines.markersize': 8, 'lines.linewidth': 2,
             'font.size': 14}
rcParams.update(newparams)
```

[2]:
```
def simpson(f, a, b, m=10):
# Find an approximation to an integral by the composite Simpson's method:
# Input:
#   f:    integrand
#   a, b: integration interval
#   m:    number of subintervals
# Output: The approximation to the integral
    n = 2*m
    x_noder = linspace(a, b, n+1)       # equidistributed nodes from a to b
    h = (b-a)/n                         # stepsize
    S1 = f(x_noder[0]) + f(x_noder[n])  # S1 = f(x_0)+f(x_n)
    S2 = sum(f(x_noder[1:n:2]))         # S2 = f(x_1)+f(x_3)+...+f(x_m)
    S3 = sum(f(x_noder[2:n-1:2]))       # S3 = f(x_2)+f(x_4)+...+f(x_{m-1})
    S = h*(S1 + 4*S2 + 2*S3)/3
    return S
```

Now using this function with our specific integral:

$$I(0,1) = \int_0^1 \cos\left(\frac{\pi}{2}x\right) = \frac{2}{\pi} = 0.636619\ldots$$

for $m = 4, 8, 16, 32, 64$.

[3]:
```
# Assignemnt 9
def f(x):                   # Integrand
    return cos((pi*x)/2)
a, b = 0, 1                 # Integration interval
I_exact = 2/pi              # Exact value of the integral (for comparision)
for m in [1,4,8,16,32,64]:
    S = simpson(f, a, b, m=m)   # Numerical solution, using m subintervals
    err = I_exact-S             # Error
    if m == 1:
        print('m = {:3d}, I = {:.8f},  S = {:.8f},  error = {:.3e}'.format(m, I_exact, S, err))
    else:
        print('m = {:3d}, I = {:.8f},  S = {:.8f},  error = {:.3e},  reduction factor = {:.3e}'.format(m, I_exact, S, err, err/err_prev))
    err_prev=err
```

```
m =   1, I = 0.63661977,  S = 0.63807119,  error = -1.451e-03
m =   4, I = 0.63661977,  S = 0.63662505,  error = -5.281e-06,  reduction factor = 3.639e-03
m =   8, I = 0.63661977,  S = 0.63662010,  error = -3.289e-07,  reduction factor = 6.228e-02
m =  16, I = 0.63661977,  S = 0.63661979,  error = -2.054e-08,  reduction factor = 6.245e-02
m =  32, I = 0.63661977,  S = 0.63661977,  error = -1.284e-09,  reduction factor = 6.249e-02
m =  64, I = 0.63661977,  S = 0.63661977,  error = -8.021e-11,  reduction factor = 6.250e-02
```

We observe that simpsons comp. rule gives us a more accurate approximation than the trapezoid rule. The trapezoid rule is nothing more than the average of the left and right hand Riemann sums. Simpsons rule gives us a weighted average that results in a more accurate approximation.

c)

We recall that the error of Simpson's rule on a single interval is given by

$$|I[f](a,b) - S[f](a,b)| = -\frac{(b-a)^5}{2880} f^{(4)}(\xi)$$

for some $\xi \in [a,b]$

We also know that

$$I[f](a,b) = \int_a^b f(x)\,dx$$

$$= \sum_{k=1}^{m} \int_{x_{k-1}}^{x_k} f(x)\,dx$$

$$= \sum_{k=1}^{m} I[f](x_{k-1}, x_k)$$

Now using the same method as in 2a), by using equally spaced x_i's and low degree interpolation we partition the integration interval we get that the error of Simpson's composite rule can be bounded by:

$$|I[f] - CSR[f]| \leq \frac{M_4}{2880} \frac{(b-a)^5}{m^4} = \frac{M_4}{2880} h^4 (b-a)$$

where $M_4 = \max_{\xi \in [a,b]} |f^{(4)}(\xi)|$