5th Oct 2023

HW 5, APPM 4600 Numerical Analysis, Lena Kellner

① a) It converges after 33 iteration to point (0.5, 0.8660254). It takes 0.00030 seconds.

So, it requires many steps for the closer method to converge.

b) For Newton we would use the inverse of the Jacobian. To avoid calculating J for every iteration one can just use the Jacobian of the initial guess.

So, $\begin{bmatrix} \frac{1}{6} & \frac{1}{18} \\ 0 & \frac{1}{6} \end{bmatrix}$ is an approximation for that:

$$J_{|(1,1)} = \begin{bmatrix} f_x = 6x & f_y = -2y \\ g_x = 3y^2 - 3x^2 & g_y = 6xy \end{bmatrix}\Big|_{(1,1)} = \begin{bmatrix} 6 & -2 \\ 0 & 6 \end{bmatrix}$$

$$J^{-1}_{|(1,1)} = \frac{1}{(6 \cdot 6 - 0 \cdot (-2))} \cdot \begin{bmatrix} 6 & -(-2) \\ 0 & 6 \end{bmatrix} = \frac{1}{36} \cdot \begin{bmatrix} 6 & 2 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & \frac{1}{18} \\ 0 & \frac{1}{6} \end{bmatrix}$$

c) Newton converges after 5 iterations to the same point. It takes 0.00032 seconds.

So, it takes approximately the same time but way less steps.

d) $f(0.5, 0.8660254) \approx 3 \cdot 0.5^2 - 0.8660254^2 = 0.75 - 0.75 = 0$

$g(0.5, 0.8660254) \approx 3 \cdot 0.5 \cdot 0.8660254^2 - 0.5^3 - 1 = 1.5 \cdot 0.75 - 0.125 - 1$

$= 1.125 - 1.125 = 0$

$\Rightarrow$ The numerical approximations are true ✓.

Codes for @ a,c

```python
import numpy as np
import math
import time
from numpy.linalg import inv
from numpy.linalg import norm

def driver():

    x0 = np.array([1,1])

    Nmax = 100
    tol = 1e-10

    t = time.time()
    for j in range(50):
      [xstar,ier,its] =  IterationScheme(x0,tol,Nmax)
    elapsed = time.time()-t
    print(xstar)
    print('IterationScheme: the error message reads:',ier)
    print('IterationScheme: took this many seconds:',elapsed/50)
    print('IterationScheme: number of iterations is:',its)

    t = time.time()
    for j in range(50):
      [xstar,ier,its] =  Newton(x0,tol,Nmax)
    elapsed = time.time()-t
    print(xstar)
    print('Newton: the error message reads:',ier)
    print('Newton: took this many seconds:',elapsed/50)
    print('Newtwon: number of iterations is:',its)


def evalF(x):

    F = np.zeros(2)

    F[0] = 3*x[0]**2 - x[1]**2
```

```
    F[0] = 3*x[0]**2 - x[1]**2
    F[1] = 3*x[0]*x[1]**2 - x[0]**3 - 1
    return F

def evalJ(x):

    J = np.array([[6*x[0], -2*x[1]], [3*x[1]**2 - 3*x[0]**2, 6*x[0]*x[1]]])
    return J

def IterationScheme(x0,tol,Nmax):

    for its in range(Nmax):
        M = np.array([[1/6, 1/18], [0, 1/6]])
        F = evalF(x0)

        x1 = x0 - M.dot(F)

        if (norm(x1-x0) < tol):
            xstar = x1
            ier =0
            return[xstar, ier, its]

        x0 = x1

    xstar = x1
    ier = 1
    return[xstar,ier,its]


def Newton(x0,tol,Nmax):

    ''' inputs: x0 = initial guess, tol = tolerance, Nmax = max its'''
    ''' Outputs: xstar= approx root, ier = error message, its = num its'''

    for its in range(Nmax):
       J = evalJ(x0)
       Jinv = inv(J)
```

```
        Jinv = inv(J)
        F = evalF(x0)

        x1 = x0 - Jinv.dot(F)

        if (norm(x1-x0) < tol):
            xstar = x1
            ier =0
            return[xstar, ier, its]

        x0 = x1

    xstar = x1
    ier = 1
    return[xstar,ier,its]


if __name__ == '__main__':
    # run the drivers only if this is called from the command line
    driver()

```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\kelln> & C:/Users/kelln/AppData/Local/Programs/Python/Python311/python.exe c:/Users/kelln/OneDrive/Desktop/HW5-1.1
[0.5       0.8660254]
IterationScheme: the error message reads: 0
IterationScheme: took this many seconds: 0.0003123331069946289
IterationScheme: number of iterations is: 33
[0.5       0.8660254]
Newton: the error message reads: 0
Newton: took this many seconds: 0.0003810739517211914
Newtwon: number of iterations is: 5
PS C:\Users\kelln>
```

(2) [1] Check that the fixed pt iteration is continuous:

$$\rightsquigarrow G(x,y) = \begin{cases} g_1(x,y) = \frac{1}{\sqrt{2}} \cdot \sqrt{1+(x+y)^2} - \frac{2}{3} \\ g_2(x,y) = \frac{1}{\sqrt{2}} \cdot \sqrt{1+(x-y)^2} - \frac{2}{3} \end{cases}$$ is a function in the area $\mathbb{R}^2 \to \mathbb{R}^2$.

It is continuous since g_1, g_2 are continuous ($\sqrt{}$ is no problem because the number in the $\sqrt{}$ is always ≥ 1).

[2] Find a $D \subset \mathbb{R}^2$ s.t. $G(x,y) \in D \quad \forall (x,y) \in D$.

Denote $x,y \in [0,1]$.

Then: $0{,}0404 \approx \frac{1}{\sqrt{2}} - \frac{2}{3} \leq f(x,y) \leq \sqrt{\frac{5}{2}} - \frac{2}{3} \approx 0{,}9144$

$0{,}0404 \approx \frac{1}{\sqrt{2}} - \frac{2}{3} \leq g(x,y) \leq \sqrt{\frac{2}{2}} - \frac{2}{3} = \frac{1}{3}$

$\Rightarrow f, g \in [0,1] \quad \forall x,y \in [0,1]$.

$\Rightarrow$ Define $D := [0,1] \times [0,1]$

[3] Find a $K \leq 1$ s.t. $\frac{\partial g_1(x,y)}{\partial x} \leq \frac{K}{n} \qquad \forall (x,y) \in D$

$$\frac{\partial g_1(x,y)}{\partial y} \leq \frac{K}{n}$$

$$\frac{\partial g_2(x,y)}{\partial x} \leq \frac{K}{n}$$

$$\frac{\partial g_2(x,y)}{\partial y} \leq \frac{K}{n}$$

$$\rightsquigarrow |g_{1x}(x,y)| = |g_{1y}| = \left| \frac{1}{2\sqrt{2}} \cdot \frac{1}{\sqrt{1+(x+y)^2}} \cdot 2(x+y) \right| = \frac{x+y}{\sqrt{2} \cdot \sqrt{1+(x+y)^2}} \overset{x,y=1}{\leq} \frac{2}{\sqrt{10}}$$

$$|g_{2x}| = |g_{2y}| = \left| \frac{x-y}{\sqrt{2} \cdot \sqrt{1+(x-y)^2}} \right| \overset{\text{or } \substack{x=1,y=0 \\ x=0,y=1}}{\leq} \frac{1}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2}$$

$\Rightarrow \frac{K}{n} < \frac{1}{2} \Rightarrow K := 1$

$\Rightarrow$ All conditions for the theorem are met. So, the iteration will converge to an unique fixed pt. $\forall (x,y) \in D$.

③ Given: $f(x,y)$ smooth function

s.t. $f(x,y)=0$ smooth curve in x-y-plane

Wanted: Some point (x^*,y^*) : • on curve $f(x,y)=0$

• in neighborhood of an initial guess (x_0,y_0) (is of the curve)

a) Given: iteration scheme $\begin{cases} x_{n+1} = x_n - \frac{f}{f_x^2+f_y^2}\cdot f_x \\ y_{n+1} = y_n - \frac{f}{f_x^2+f_y^2}\cdot f_y \end{cases}$ with f, f_x, f_y evaluated at the location (x_n,y_n)

Hint: Look for a new iterate that ① lies on gradient line through (x_n,y_n)

② satisfies $f(x,y)=0$

⇨ Apply Newton to this 2×2 system.

Soln.:

• Gradient $\nabla f\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix} f_x(x,y) \\ f_y(x,y)\end{pmatrix}$

Gradient line through (x_n,y_n): $\frac{x-x_n}{f_x(x_n,y_n)} = \frac{y-y_n}{f_y(x_n,y_n)} \quad \Leftrightarrow \quad \frac{x-x_n}{f_x(x_n,y_n)} - \frac{y-y_n}{f_y(x_n,y_n)} = 0$

Substitute x_{n+1}, y_{n+1} into $f(x,y)=0$: $f\left(x_n - \frac{f}{f_x^2+f_y^2}\cdot f_x \;,\; y_n - \frac{f}{f_x^2+f_y^2}\cdot f_y\right) = 0$

apply Newton

Now?

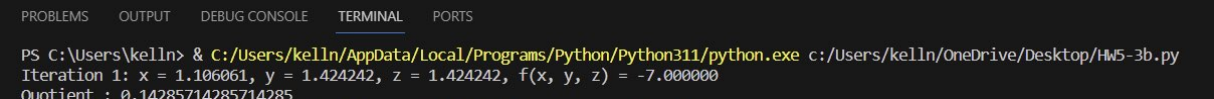
b) The point is $(1,093642 \;,\; 1,360328 \;,\; 1,360328)$.

$\frac{\text{"error of iteration } i\text{"}}{(\text{"error of iteration } (i-1)\text{"})^2} \xrightarrow{i\to\infty} 0,016.$

⇒ order of convergence is 2.

code 36

```python
import numpy as np

# Initial guess
x, y, z = 1, 1, 1

# Number of iterations
Nmax = 5

# Empty Array for the errors
errors=[]

for i in range(Nmax):
    # Calculate f, f_x, f_y, f_z
    f = x**2 + 4*y**2 + 4*z**2 - 16
    fx = 2*x
    fy = 8*y
    fz = 8*z

    # Calculate the correction using the iteration scheme
    denominator = fx**2 + fy**2 + fz**2
    correction_x = f * fx / denominator
    correction_y = f * fy / denominator
    correction_z = f * fz / denominator

    # Update x, y, z
    x -= correction_x
    y -= correction_y
    z -= correction_z

    # Compute the error, append it and create the quotient
    error = np.linalg.norm((f))
    errors.append(error)
    quotient = np.array(errors[i]) / np.array(errors[i-1])**2

    # Print results
    print(f"Iteration {i+1}: x = {x:.6f}, y = {y:.6f}, z = {z:.6f}, f(x, y, z) = {f:.6f}")
    print("Quotient :", quotient)
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\kelln> & C:/Users/kelln/AppData/Local/Programs/Python/Python311/python.exe c:/Users/kelln/OneDrive/Desktop/HW5-3b.py
Iteration 1: x = 1.106061, y = 1.424242, z = 1.424242, f(x, y, z) = -7.000000
Quotient : 0.14285714285714285
Iteration 2: x = 1.093926, y = 1.361742, z = 1.361742, f(x, y, z) = 1.451102
Quotient : 0.02961432506887058
Iteration 3: x = 1.093642, y = 1.360329, z = 1.360329, f(x, y, z) = 0.031398
Quotient : 0.014910966545067272
Iteration 4: x = 1.093642, y = 1.360328, z = 1.360328, f(x, y, z) = 0.000016
Quotient : 0.016274298436189476
Iteration 5: x = 1.093642, y = 1.360328, z = 1.360328, f(x, y, z) = 0.000000
Quotient : 0.01631423057092832
PS C:\Users\kelln>