3940 AS08

Main

Question 1

From the question, we have

$$
\begin{aligned}
x_1 = g_1(x_1, x_2) &= \frac{1}{\sqrt{5}} x_2 \\
x_2 = g_2(x_1, x_2) &= \frac{1}{4}(\sin x_1 + \cos x_2)
\end{aligned}
$$

Since

$$
\begin{aligned}
\frac{\partial g_1}{\partial x_1} &= 0 \\
\frac{\partial g_2}{\partial x_1} &= \frac{1}{4}\cos x_1 \\
\frac{\partial g_1}{\partial x_1} &= \frac{1}{\sqrt{5}} \\
\frac{\partial g_2}{\partial x_2} &= -\frac{1}{4}\sin x_2
\end{aligned}
$$

Therefore, in domain $D = \{(x_1, x_2)' | 0 \leq x_1, x_2 \leq 1\}$

$$
\begin{aligned}
|\frac{\partial g_1}{\partial x_1}| &= 0 \\
|\frac{\partial g_2}{\partial x_1}| &\leq \frac{1}{4} \\
|\frac{\partial g_1}{\partial x_1}| &= \frac{1}{\sqrt{5}} \\
|\frac{\partial g_2}{\partial x_2}| &\leq -\frac{1}{4}
\end{aligned}
$$

So, $|\frac{\partial g_i}{\partial x_j}| \quad \leq \frac{K}{2}$

Therefore, the mapping has a unique fixed point. (a)

According to the fixed-point iteration

$$x_1^k = \frac{1}{\sqrt{5}}\{x_2^{k-1}\}$$

$$x_2^k = \frac{1}{4}\{\sin x_1^{k-1} + \cos x_2^{k-1}\}$$

Consider the initial guessed solutions as

$$x_1^0 = 0.25, x_2^0 = 0.25$$

Consider the iteration steps with tolerance 10^{-5} that is $||x^k - x^{k-1}|| \leq 10^{-5}$

k	x_1^k	x_2^k	$\lVert \mathbf{x}^k - \mathbf{x}^{k-1} \rVert$
0	0.2500000	0.2500000	
1	0.1118034	0.3040791	0.1381966
2	0.1359883	0.2664234	0.0376557
3	0.1191482	0.2750721	0.0168401
4	0.1230160	0.2703180	0.0047540
5	0.1208899	0.2715980	0.0021261
6	0.1214623	0.2709848	0.0006132
7	0.1211881	0.2711679	0.0002742
8	0.1212700	0.2710876	0.0000819
9	0.1212341	0.2711133	0.0000359
10	0.1212456	0.2711027	0.0000115
11	0.1212408	0.2711062	0.0000048

Hence, the solution of the non-linear equation is $x_1 = 0.12124, x_2 = 0.27111$ (b)

According to the Gauss Seidel acceleration method,

$$x_1^k = \frac{1}{\sqrt{5}}\{x_2^{k-1}\}$$
$$x_2^k = \frac{1}{4}\{\sin x_1^k + \cos x_2^{k-1}\}$$

Consider the initial guessed solution as

$$x_1^0 = 0.25, x_2^0 = 0.25$$

So,

k	x_1^k	x_2^k	$\lVert \mathbf{x}^k - \mathbf{x}^{k-1} \rVert$
0	0.2500000	0.2500000	
1	0.1118034	0.2701208	0.1381966
2	0.1208017	0.2710617	0.0089983
3	0.1212225	0.2711032	0.0004208
4	0.1212411	0.2711051	0.0000186
5	0.1212419	0.2711052	0.0000008

Therefore, the solution of the non-linear equation is $x_1 = 0.12124, x_2 = 0.271111$ (c)

Question 2

The nonlinear system of the equations are

$$3x_1^2 - x_2^2 = 0$$
$$3x_1x_2^2 - x_1^3 - 1 = 0$$

The Jacobean matrix is

$$J(x) = \begin{pmatrix} 6x_1 & -2x_2 \\ 3x_2^2 - 3x_1^2 & 6x_1x_2 \end{pmatrix}$$

and $x^{(0)=(1,1)^t}$

So,

$$\begin{aligned}F(x^{(0)}) &= (2,1)^t \\ A_0 &= J(x^0) \\ &= \begin{pmatrix} 6 & -2 \\ 0 & 6 \end{pmatrix}\end{aligned}$$

Solving the linear system,

$$\begin{aligned}J(x^{(0)})y^{(0)} &= -F(x^{(0)}) \\ y^{(0)} &= -J(x^{(0)})^{-1}F(x^{(0)}) \\ &= -\begin{bmatrix} 6 & -2 \\ 0 & 6 \end{bmatrix}^{-1}\begin{bmatrix} 2 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -0.39 \\ -0.17 \end{bmatrix}\end{aligned}$$

Now,

$$y_1 = F[x^{(0)}] - F[x^{(0)}] = \begin{bmatrix} -1.5726 \\ -0.9663 \end{bmatrix}$$

$$s_1 = x^{(1)} - x^{(0)} = \begin{bmatrix} -0.39 \\ -0.17 \end{bmatrix}$$

So that

$$[A_1]^{-1} = [A_0]^{-1} + [(s_1' - A_0^{-1}y_1)s_1'A_0^{-1}] = \begin{bmatrix} 0.238271 & .130749 \\ -0.074858 & .288279 \end{bmatrix}$$

and

$$x^{(2)} = x^{(1)} - [A_1]^{-1}F(x^{(1)}) = \begin{bmatrix} .50333 \\ .8523 \end{bmatrix}$$

Question 3

$$\begin{aligned}g(x_1,x_2) &= \cos(x_1+x_2) + \sin x_1 \cos x_2 \\ \frac{\partial g}{\partial x_1} &= -\sin(x_1+x_2) + \cos x_1 = 0 \\ \frac{\partial g}{\partial x_2} &= -\sin(x_1+x_2) - \sin x_2 = 0 \\ \Delta g &= (-\sin(x_1+x_2) + \cos x_i)i + (-\sin(x_1+x_2) - \sin x_2)j\end{aligned}$$

Question 4

Given $y'' = y' + 2y + \cos x, 0 \leq x \leq \frac{\pi}{2}, y(0) = -0.3$ and $y(\frac{\pi}{2}) = -0.1$

The actual solution is

$$y(x) - \frac{1}{10}(\sin x + 3\cos x)$$

Choose $h = \frac{\pi}{4}$, then $N = 2$

The Linear Shooting Method is stated as follows:

If $y'' = p(x) \cdot y' + q(x) \cdot y + r(x)$ for $a \leq x \leq b$ with $y(a) = \alpha$ and $y(b) = \beta$ is a linear boundary value problem and if $y_1(x)$ represents the solution of $y'' = p(x) \cdot y' + q(x) \cdot y + r(x)$ with $a \leq x \leq b, y(a) = \alpha, y(b) = 0$ and $y_2(x)$ represents the solution of $y'' = p(x) \cdot y' + q(x) \cdot y$ with $a \leq x \leq b, y(a) = 0, y(b) = 1$ respectively, then the solution is $y(x) = y_1(x) + \left[\frac{\beta - y_1(b)}{y_2(b)}\right] y_2(x)$ where $y_2(b) \neq 0$

$$\cdots (\text{Calculated by Code})$$

And the final result of approximate value

$$\overline{y}(0) = -0.3,$$
$$\overline{y}(\frac{\pi}{4}) = -0.2828427130,$$
$$\overline{y}(\frac{\pi}{2}) = -0.1$$

and the actual is

$$y(0) = -0.3,$$
$$y(\frac{\pi}{4}) = -0.2828427125,$$
$$y(\frac{\pi}{2}) = -0.1$$

And we could see that the difference is very small (less than 10^{-5})

Question 5

```
from math import cos, sin, pi

# Settings
g1 = 2
g2 = 1

m = 4
h = 1 / m
net = [i * h for i in range(m + 1)]

# Initial
r = lambda x: 2 * x + 3  # r(x)
p = lambda x: -3  # p(x)
```

```
q = lambda x: 2  # q(x)

P = [p(i * h) for i in range(m + 1)]
Q = [q(i * h) for i in range(m + 1)]
F = [-r(i * h) for i in range(1, m)]

# Solving traditional matrix by matrix factorization
A = [-(P[i + 1] / h ** 2) for i in range(m - 1)]  # upper diagonal
C = [-(P[i + 1] + P[i + 2]) / h ** 2 - Q[i + 1] for i in range(m - 1)]  # diagonal
B = [-(P[i + 2] / h ** 2) for i in range(m - 1)]  # lower diagonal

alphas = [0]
betas = [g1]
for i in range(m - 1):
    alphas.append(B[i] / (C[i] - alphas[i] * A[i]))
    betas.append((betas[i] * A[i] + F[i]) / (C[i] - alphas[i] * A[i]))

u = [g2]
for i in range(m):
    u.insert(0, alphas[m - i - 1] * u[0] + betas[m - i - 1])

for xova, yova in zip(net, u):
    outs = "{0} {1}\n".format(xova, yova)
    print(outs)
```

```
x   y
0.0 2.0

0.25 1.7270526781940054

0.5 1.4550614947965939

0.75 1.2057760824493244

1.0 1
```

Project

> *I just write it for helping calculate the problems above, so I did not show graphical results.*

```
from math import sin, log

# Settings
```

```
x1, x2 = 1, 2
y1, y2 = 1, 2

N = 10
h = (x2 - x1) / N


def r(x):
    return sin(log(x)) / x ** 2


def p(x):
    return -2 / x


def q(x):
    return 2 / x ** 2


# initial
net = [x1 + i * h for i in range(N + 1)]
P = [p(x1 + i * h) for i in range(N + 1)]
Q = [q(x1 + i * h) for i in range(N + 1)]
F = [-r(x1 + i * h) for i in range(1, N)]

# Solving traditional matrix by matrix factorization
A = [-(P[i + 1] / h ** 2) for i in range(N - 1)]  # upper diagonal
C = [-(P[i + 1] + P[i + 2]) / h ** 2 - Q[i + 1] for i in range(N - 1)]  # diagonal
B = [-(P[i + 2] / h ** 2) for i in range(N - 1)]  # lower diagonal

alphas = [0]
betas = [y1]
for i in range(N - 1):
    alphas.append(B[i] / (C[i] - alphas[i] * A[i]))
    betas.append((betas[i] * A[i] + F[i]) / (C[i] - alphas[i] * A[i]))

u = [y2]
for i in range(N):
    u.insert(0, alphas[N - i - 1] * u[0] + betas[N - i - 1])

for xi, wi in zip(net, u):
    print("{0} {1}\n".format(xi, wi))
```