

# Computational Homework11 Report

**Student:** 纪浩正, [jihz2023@mail.sustech.edu.cn](mailto:jihz2023@mail.sustech.edu.cn)

## 1 Fixed-point method

### 1.1 Introduction

The fixed-point iteration method is a powerful numerical technique for finding the roots of a system of non-linear equations. Consider a system of two non-linear equations in two variables,  $x$  and  $y$ :

$$f_0(x, y) = 0$$

$$g_0(x, y) = 0$$

In this problem, we aim to find the roots of the specific system:

$$f_0(x, y) = 4x^2 + y^2 - 4 = 0 \quad (1)$$

$$g_0(x, y) = x + y - \sin(x - y) = 0 \quad (2)$$

The core functions for these equations are defined in Python as:

```
def f0(x,y):
    return 4*x**2+y**2-4
def g0(x,y):
    return x+y-np.sin(x-y)
```

To apply the fixed-point method, we must transform this system into an equivalent fixed-point problem of the form:

$$x = F(x, y)$$

$$y = G(x, y)$$

The choice of iteration functions  $F$  and  $G$ , as well as the iterative update strategy, are crucial for convergence. We will explore two different ways of defining  $F(x, y)$  and  $G(x, y)$  (Scheme 1 and Scheme 2), and two corresponding iterative update methods (Original/Jacobi-style and Improved/Gauss-Seidel style).

## Defining Iteration Functions

We consider two schemes for defining the iteration functions:

**Scheme 1: Simple Rearrangement (Used in Original Iteration)** This scheme uses iteration functions derived from a simple algebraic rearrangement of the original equations.

$$F_1(x, y) = x + 1 - x^2 - \frac{y^2}{4} \quad (3)$$

$$G_1(x, y) = \sin(x - y) - x \quad (4)$$

In the Python implementation, these are typically defined as:

```
def f(x,y):
    return x+1-x**2-y**2/4
def g(x,y):
    return np.sin(x-y)-x
```

**Scheme 2: Alternative Rearrangement (Used in Improved Iteration)** In this scheme, we reformulate the first equation (1) to directly solve for  $x$ , assuming  $x > 0$ . This typically leads to better convergence properties.

$$F_2(x, y) = \sqrt{1 - \frac{y^2}{4}} \quad (5)$$

The second iteration function  $G_2(x, y)$  remains the same as  $G_1(x, y)$  (from the second original equation):

$$G_2(x, y) = \sin(x - y) - x \quad (6)$$

The Python functions for this scheme would be:

```
def new_f(x,y):
    return np.sqrt(1-y**2/4)
def g(x,y):
    return np.sin(x-y)-x
```

## Iterative Update Strategies

We apply two distinct fixed-point iteration strategies:

**Method A: Original Fixed Point Method (Jacobi-style)** This method employs a Jacobi-style update, where both  $x_{k+1}$  and  $y_{k+1}$  are computed exclusively from values at step  $k$ :

$$x_{k+1} = F(x_k, y_k)$$

$$y_{k+1} = G(x_k, y_k)$$

This method can be used with either function Scheme 1 or Scheme 2. The Python function ‘fp’ (when ‘mode=0’) implements this:

```
def fp(f,g,x,y,mode=0,count=0,tol=1e-6):
    if mode==0:
        x_new=f(x,y)
        y_new=g(x,y)
        d=(x-x_new)**2+(y-y_new)**2
        if count>=300:
            print(f'Reached maximum iterations ({count}). Returning current approximation.')
            return x_new, y_new
        elif d<tol**2:
            print(f'Converged in {count + 1} iterations.')
            return x_new, y_new
        else:
            return fp(f,g,x_new,y_new,mode,count+1)

def fixed_point(f,g,x0,y0,mode=0):
    return fp(f,g,x0,y0,mode)
```

**Method B: Improved Fixed Point Method (Gauss-Seidel style)** This method utilizes a Gauss-Seidel style update. The newly computed  $x_{k+1}$  is immediately used when calculating  $y_{k+1}$ :

$$x_{k+1} = F(x_k, y_k)$$

$$y_{k+1} = G(x_{k+1}, y_k)$$

This method is generally expected to converge faster than the Jacobi style if it converges at all. This method can also be used with either function Scheme 1 or Scheme 2. The Python function ‘fp’ (when ‘mode=1’) implements this:

```
def fp(f,g,x,y,mode=0,count=0,tol=1e-6):
    # ... (code for mode=0 omitted for brevity, as provided in the original text)
    else: # This implicitly handles mode=1 in the full function
        x_new=f(x,y)
        y_new=g(x_new,y)
```

```

d=(x-x_new)**2+(y-y_new)**2
if count>=300:
    print(f'Reached maximum iterations ({count}). Returning current approximation.')
    return x_new, y_new
elif d<tol**2:
    print(f'Converged in {count + 1} iterations.')
    return x_new, y_new
else:
    return fp(f,g,x_new,y_new,mode,count+1)

def fixed_point(f,g,x0,y0,mode=0):
    return fp(f,g,x0,y0,mode)

```

This method corresponds to ‘mode=1’ in our implementation. The iterative process for all scenarios begins with an initial guess  $(x_0, y_0)$  and continues until the squared Euclidean distance between successive approximations,  $(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2$ , falls below a specified tolerance squared (e.g.,  $\text{tol}^2$ ), or a maximum number of iterations is reached. The convergence properties vary significantly between these different choices of iteration functions and update styles.

## 1.2 Result

The fixed-point iteration was performed using an initial guess  $(x_0, y_0) = (1.0, 0.0)$ , a tolerance of  $1 \times 10^{-6}$  for the squared distance, and a maximum of 300 iterations.

### Comparison and Summary of Results

A comparative overview of the four scenarios, derived from the Python output in Figure 1, is presented in Table 1.

**Table 1 Comparison of Fixed-Point Iteration Results.** \*Max Residual refers to the maximum absolute value among  $f_0(x, y)$  and  $g_0(x, y)$  at the approximated root.

Metric	Original		Improved	
	Scheme A	Scheme B	Scheme A	Scheme B
<b>Iterations</b>	160	15	>300 (Diverged)	17
<b>Approx. Root(<math>x, y</math>)</b>	(0.9986066, -0.1055308)	(0.9986070, -0.1055307)	(1.1667598, -0.2881905)	(0.9986070, -0.1055307)
<b>Max Residual*</b>	$2.44 \times 10^{-6}$	$3.33 \times 10^{-7}$	$1.53 \times 10^0$	$3.42 \times 10^{-7}$

The experiments unequivocally demonstrate the critical influence of both the chosen iteration functions and the update strategy on the convergence of the fixed-point method.

```

PS C:\Users\20369> & E:/anaconda3/envs/py310/python.exe "c:/Users/20369/Homework/homework11/homework11root.py"
##### Original fixed point method #####
--- Without appropriate F(x,y) ---
Converged in 160 iterations.

Final approximated root: (0.998606630, -0.105530847)

--- Verification using original system (f0(x,y)=0, g0(x,y)=0) ---
f0: -2.43729369e-06
g0: -6.87129325e-07
--- With appropriate F(x,y) ---
Converged in 15 iterations.

Final approximated root: (0.998606959, -0.105530728)

--- Verification using original system (f0(x,y)=0, g0(x,y)=0) ---
f0: 1.69431615e-07
g0: -3.33045869e-07

#####
Improved fixed point method #####
--- Without appropriate F(x,y) ---
Reached maximum iterations (300). Returning current approximation.

Final approximated root: (1.166759810, -0.288190498)

--- Verification using original system (f0(x,y)=0, g0(x,y)=0) ---
f0: 1.52836758e+00
g0: -1.14728039e-01
--- With appropriate F(x,y) ---
Converged in 17 iterations.

Final approximated root: (0.998606958, -0.105530734)

--- Verification using original system (f0(x,y)=0, g0(x,y)=0) ---
f0: 1.60578685e-07
g0: -3.42295646e-07
PS C:\Users\20369>

```

**Figure 1 Detailed Python Output for All Fixed-Point Iteration Scenarios.**

## 2 Matrix Norms and Condition Numbers

This section details the calculation of different matrix norms and their corresponding condition numbers for the given matrix  $\mathbf{A}$ . The matrix under consideration is:

$$\mathbf{A} = \begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix}$$

## 2.1 Matrix Norms

Matrix norms measure the "size" of a matrix. We consider the  $L_1$ ,  $L_2$ , and  $L_\infty$  norms.

### 2.1.1 The $L_1$ Norm (Column Sum Norm)

The  $L_1$  norm of a matrix  $\mathbf{A}$  is the maximum absolute column sum:

$$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$$

For matrix  $\mathbf{A}$ :

$$\text{Column 1 sum: } |-2| + |1| + |0| = 3$$

$$\text{Column 2 sum: } |1| + |-2| + |1| = 4$$

$$\text{Column 3 sum: } |0| + |1| + |-2| = 3$$

Thus,  $\|\mathbf{A}\|_1 = \max(3, 4, 3) = 4$ .

### 2.1.2 The $L_\infty$ Norm (Row Sum Norm)

The  $L_\infty$  norm of a matrix  $\mathbf{A}$  is the maximum absolute row sum:

$$\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^m |a_{ij}|$$

For matrix  $\mathbf{A}$ :

$$\text{Row 1 sum: } |-2| + |1| + |0| = 3$$

$$\text{Row 2 sum: } |1| + |-2| + |1| = 4$$

$$\text{Row 3 sum: } |0| + |1| + |-2| = 3$$

Thus,  $\|\mathbf{A}\|_\infty = \max(3, 4, 3) = 4$ .

### 2.1.3 The $L_2$ Norm (Spectral Norm)

The  $L_2$  norm of a matrix  $\mathbf{A}$  is defined as:

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^T \mathbf{A})}$$

where  $\lambda_{\max}(\mathbf{A}^T \mathbf{A})$  is the largest eigenvalue of  $\mathbf{A}^T \mathbf{A}$ . For a symmetric matrix,  $\mathbf{A}^T = \mathbf{A}$ , so  $\mathbf{A}^T \mathbf{A} = \mathbf{A}^2$ . A symmetric matrix  $\mathbf{A}$  can be spectrally decomposed as  $\mathbf{A} = \mathbf{U} \Lambda \mathbf{U}^T$ , where  $\mathbf{U}$  is an orthogonal matrix ( $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ ) and  $\Lambda$  is a diagonal matrix containing the eigenvalues of  $\mathbf{A}$  on its diagonal. Then, we can find the spectral decomposition of  $\mathbf{A}^2$ :

$$\mathbf{A}^2 = (\mathbf{U} \Lambda \mathbf{U}^T)(\mathbf{U} \Lambda \mathbf{U}^T) = \mathbf{U} \Lambda (\mathbf{U}^T \mathbf{U}) \Lambda \mathbf{U}^T = \mathbf{U} \Lambda \mathbf{I} \Lambda \mathbf{U}^T = \mathbf{U} \Lambda^2 \mathbf{U}^T$$

The eigenvalues of  $\mathbf{A}^2$  are the diagonal entries of  $\Lambda^2$ , which are  $\lambda_k^2$ , where  $\lambda_k$  are the eigenvalues of  $\mathbf{A}$ . Therefore, the largest eigenvalue of  $\mathbf{A}^T \mathbf{A} = \mathbf{A}^2$  is  $\max_k(\lambda_k^2)$ . Substituting this into the definition of the  $L_2$  norm:

$$\|\mathbf{A}\|_2 = \sqrt{\max_k(\lambda_k^2)} = \max_k \sqrt{\lambda_k^2} = \max_k |\lambda_k|$$

Thus, for a symmetric matrix, the  $L_2$  norm simplifies to its largest absolute eigenvalue.

The eigenvalues of  $\mathbf{A}$  are found by solving  $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$ . For the given matrix  $\mathbf{A}$ :

$$\det \begin{pmatrix} -2 - \lambda & 1 & 0 \\ 1 & -2 - \lambda & 1 \\ 0 & 1 & -2 - \lambda \end{pmatrix} = 0$$

Solving this characteristic equation yields the eigenvalues:

$$\begin{aligned} \lambda_1 &= -2 \\ \lambda_2 &= -2 + \sqrt{2} \approx -0.586 \\ \lambda_3 &= -2 - \sqrt{2} \approx -3.414 \end{aligned}$$

The absolute values of the eigenvalues are:  $|\lambda_1| = 2$ ,  $|\lambda_2| \approx 0.586$ ,  $|\lambda_3| \approx 3.414$ . Therefore,  $\|\mathbf{A}\|_2 = \max(|\lambda_k|) = |-2 - \sqrt{2}| = 2 + \sqrt{2} \approx 3.414$ .

## 2.2 Condition Numbers

The condition number  $\kappa(\mathbf{A})$  measures the sensitivity of the solution of a linear system  $\mathbf{Ax} = \mathbf{b}$  to input perturbations. It is defined as:

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

We first need the inverse matrix  $\mathbf{A}^{-1}$ . The inverse of  $\mathbf{A}$  is:

$$\mathbf{A}^{-1} = \begin{pmatrix} -3/4 & -1/2 & -1/4 \\ -1/2 & -1 & -1/2 \\ -1/4 & -1/2 & -3/4 \end{pmatrix}$$

### 2.2.1 Condition Number in $L_1$ Norm, $\kappa_1(\mathbf{A})$

First, calculate  $\|\mathbf{A}^{-1}\|_1$ :

$$\text{Column 1 sum: } |-3/4| + |-1/2| + |-1/4| = 1.5$$

$$\text{Column 2 sum: } |-1/2| + |-1| + |-1/2| = 2$$

$$\text{Column 3 sum: } |-1/4| + |-1/2| + |-3/4| = 1.5$$

So,  $\|\mathbf{A}^{-1}\|_1 = \max(1.5, 2, 1.5) = 2$ . Then,  $\kappa_1(\mathbf{A}) = \|\mathbf{A}\|_1 \cdot \|\mathbf{A}^{-1}\|_1 = 4 \cdot 2 = 8$ .

### 2.2.2 Condition Number in $L_\infty$ Norm, $\kappa_\infty(\mathbf{A})$

First, calculate  $\|\mathbf{A}^{-1}\|_\infty$ :

$$\text{Row 1 sum: } |-3/4| + |-1/2| + |-1/4| = 1.5$$

$$\text{Row 2 sum: } |-1/2| + |-1| + |-1/2| = 2$$

$$\text{Row 3 sum: } |-1/4| + |-1/2| + |-3/4| = 1.5$$

So,  $\|\mathbf{A}^{-1}\|_\infty = \max(1.5, 2, 1.5) = 2$ . Then,  $\kappa_\infty(\mathbf{A}) = \|\mathbf{A}\|_\infty \cdot \|\mathbf{A}^{-1}\|_\infty = 4 \cdot 2 = 8$ .

### 2.2.3 Condition Number in $L_2$ Norm, $\kappa_2(\mathbf{A})$

The condition number in the  $L_2$  norm is defined as:

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2$$

Now let's consider  $\mathbf{A}^{-1}$ . If  $\mathbf{A}$  is invertible, its inverse can be expressed as:

$$\mathbf{A}^{-1} = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)^{-1} = (\mathbf{U}^T)^{-1}\mathbf{\Lambda}^{-1}\mathbf{U}^{-1}$$

Since  $\mathbf{U}$  is orthogonal,  $\mathbf{U}^{-1} = \mathbf{U}^T$  and  $(\mathbf{U}^T)^{-1} = \mathbf{U}$ . Thus:

$$\mathbf{A}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T$$

Applying the  $L_2$  norm definition for symmetric matrices to  $\mathbf{A}^{-1}$ :

$$\|\mathbf{A}^{-1}\|_2 = \max_k |\lambda_k(\mathbf{A}^{-1})| = \max_k \left| \frac{1}{\lambda_k(\mathbf{A})} \right| = \frac{1}{\min_{k, \lambda_k \neq 0} |\lambda_k(\mathbf{A})|}$$

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2 = \left( \max_k |\lambda_k(\mathbf{A})| \right) \cdot \left( \frac{1}{\min_{k, \lambda_k \neq 0} |\lambda_k(\mathbf{A})|} \right) = \frac{\max_k |\lambda_k(\mathbf{A})|}{\min_{k, \lambda_k \neq 0} |\lambda_k(\mathbf{A})|}$$

Using the eigenvalues calculated previously:  $\max |\lambda_k| = 2 + \sqrt{2}$   $\min |\lambda_k| = 2 - \sqrt{2}$  Thus,

$$\kappa_2(\mathbf{A}) = \frac{2 + \sqrt{2}}{2 - \sqrt{2}} = \frac{(2 + \sqrt{2})(2 + \sqrt{2})}{(2 - \sqrt{2})(2 + \sqrt{2})} = \frac{4 + 4\sqrt{2} + 2}{4 - 2} = \frac{6 + 4\sqrt{2}}{2} = 3 + 2\sqrt{2} \approx 5.828$$

### 2.3 Summary of Results

The calculated matrix norms and condition numbers for  $\mathbf{A}$  are:

- $\|\mathbf{A}\|_1 = 4$
- $\|\mathbf{A}\|_\infty = 4$
- $\|\mathbf{A}\|_2 = 2 + \sqrt{2} \approx 3.414$
- $\kappa_1(\mathbf{A}) = 8$
- $\kappa_\infty(\mathbf{A}) = 8$
- $\kappa_2(\mathbf{A}) = 3 + 2\sqrt{2} \approx 5.828$

These condition numbers indicate that the matrix  $\mathbf{A}$  is relatively well-conditioned.

## 3 Gaussian Elimination (Naive)

This section demonstrates solving a linear system using Gaussian elimination without pivoting (i.e., naive Gaussian elimination).

The linear system  $\mathbf{Ax} = \mathbf{b}$  is given by:

$$\mathbf{A} = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 1 & 9 \\ 1 & 2 & 6 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$$

We form the augmented matrix  $[\mathbf{A}|\mathbf{b}]$ :

$$\left( \begin{array}{ccc|c} 2 & 3 & 4 & 0 \\ 1 & 1 & 9 & 2 \\ 1 & 2 & 6 & 1 \end{array} \right)$$

### 3.1 Forward Elimination

#### Step 1: Eliminate $x_1$

Using  $a_{11} = 2$  as the pivot.

- $R_2 \leftarrow R_2 - (1/2)R_1: \left( \begin{array}{ccc|c} 1 & 1 & 9 & 2 \end{array} \right) - 0.5 \left( \begin{array}{ccc|c} 2 & 3 & 4 & 0 \end{array} \right) = \left( \begin{array}{ccc|c} 0 & -0.5 & 7 & 2 \end{array} \right)$
- $R_3 \leftarrow R_3 - (1/2)R_1: \left( \begin{array}{ccc|c} 1 & 2 & 6 & 1 \end{array} \right) - 0.5 \left( \begin{array}{ccc|c} 2 & 3 & 4 & 0 \end{array} \right) = \left( \begin{array}{ccc|c} 0 & 0.5 & 4 & 1 \end{array} \right)$

The matrix becomes:

$$\left( \begin{array}{ccc|c} 2 & 3 & 4 & 0 \\ 0 & -0.5 & 7 & 2 \\ 0 & 0.5 & 4 & 1 \end{array} \right)$$

#### Step 2: Eliminate $x_2$

Using  $a_{22} = -0.5$  as the pivot (no row swap, despite  $a_{32} = 0.5$  having the same magnitude).

- $R_3 \leftarrow R_3 + R_2: \left( \begin{array}{ccc|c} 0 & 0.5 & 4 & 1 \end{array} \right) + \left( \begin{array}{ccc|c} 0 & -0.5 & 7 & 2 \end{array} \right) = \left( \begin{array}{ccc|c} 0 & 0 & 11 & 3 \end{array} \right)$

The matrix is now in upper triangular form:

$$\left( \begin{array}{ccc|c} 2 & 3 & 4 & 0 \\ 0 & -0.5 & 7 & 2 \\ 0 & 0 & 11 & 3 \end{array} \right)$$

### 3.2 Backward Substitution

From the upper triangular matrix, we have the system:

$$2x_1 + 3x_2 + 4x_3 = 0$$

$$-0.5x_2 + 7x_3 = 2$$

$$11x_3 = 3$$

Solving from bottom up:

- $x_3 = 3/11$
- $-0.5x_2 + 7(3/11) = 2 \Rightarrow -0.5x_2 = 2 - 21/11 = 1/11 \Rightarrow x_2 = (1/11)/(-0.5) = -2/11$
- $2x_1 + 3(-2/11) + 4(3/11) = 0 \Rightarrow 2x_1 - 6/11 + 12/11 = 0 \Rightarrow 2x_1 + 6/11 = 0 \Rightarrow x_1 = -3/11$

### 3.3 Solution

The solution to the linear system is:

$$\mathbf{x} = \begin{pmatrix} -3/11 \\ -2/11 \\ 3/11 \end{pmatrix}$$