MT308 Numerical Analysis

Girne American University
Faculty of Engineering
Spring 2026
Dr. Kamil Dimililer

1

Solving Nonlinear Equations f(x)=0

- **Bracketing Methods**

If the function is continuous on the bracket

Bisection Method
False-Position Method

And if the bracket includes only one root, it always converges

- **Open Methods**

➢Do NOT require an interval
➢Start from one or two initial guesses
➢Usually faster than bracketing methods
➢Do NOT guarantee convergence

Fixed Point Iteration
Newton-Raphson
Secant Method

Open methods are very sensitive to the initial value(s). A poor initial guess may lead to divergence, convergence to the wrong root, or oscillation.

2

Exercise

a) Develop a fixed-point iteration to find the root of,

$$f(x) = e^{-(x-1)/4} - \frac{x}{3}$$

b) Compute the root up to the fourth iteration.
c) What is the absolute error at the fourth iteration?
d) Solve this problem using the bisection method
e) Solve this problem using the Newton-Raphson Method
f) Solve this problem using the Secant Method
g) Solve this problem using the Regula-Falsi Method

3

Systems of Equations

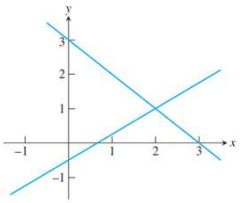
- The problem of solving several simultaneous equations in several variables arises in many scientific/engineering problems. Several methods are available for solving linear algebraic equations *by hand*. The appropriate choice depends on user preference, on the number of equations, and on the structure of the equations to be solved.
- **Direct Methods:** Direct techniques provide the exact solution to a system in a finite number of steps. The solutions are found with some round-off errors. We will learn the Gaussian Elimination Method as a direct method
- **Iterative Methods:** These methods begin with an ***initial guess*** and refine it at each step, converging to the solution vector. We will learn the Jacobi Method and the Gauss-Seidel Method

4

Systems of linear equations

- Two equations should be solved for x and y, where x=2 and y=1.

$$\begin{aligned} x + y &= 3 \\ 3x - 4y &= 2. \end{aligned}$$



- The four equations should be solved for x_1, x_2, x_3, and x_4.
- In Linear Algebra, you learned several techniques for solving such systems.

$$\begin{aligned} x_1 + x_2 + 3x_4 &= 4, \\ 2x_1 + x_2 - x_3 + x_4 &= 1, \\ 3x_1 - x_2 - x_3 + 2x_4 &= -3, \\ -x_1 + 2x_2 + 3x_3 - x_4 &= 4, \end{aligned}$$

5

A **singular problem** refers to a set of equations having either **no unique solution** or **no solution** at all. For example, the set

$$3x - 4y = 5$$
$$6x - 8y = 10$$

is singular and has no unique solution because the second equation is identical to the first equation, multiplied by 2. These equations are **linearly dependent equations**.

If the number of equations equals the number of unknowns, and if $|A^{-1}| \neq 0$, then the equation set **has a solution** and it is **unique**.

6

Matrices and Vectors

An n × m (n by m) **matrix** is a rectangular array of elements with n rows and m columns

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

The 1 × n matrix $A = [a_{11} \;\; a_{12} \;\; \cdots \;\; a_{1n}]$

is called an **n-dimensional row vector**

An n × 1 matrix

$$A = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix}$$

is called an **n-dimensional column vector**

Usually, the common subscripts are omitted for vectors as the following

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{y} = [y_1 \;\; y_2 \;\; \ldots \;\; y_n]$$

7

Matrix representation of the System of linear equations

$$[A][x] = [b]$$

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ \vdots \qquad\qquad & \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n, \end{aligned}$$

Augmented matrix:

$$[A, \mathbf{b}] = \left[\begin{array}{cccc:c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

Solution Vector:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

8

Direct Methods

Cramer's Method also solves equations using determinants. But it is better suited for small-sized problems only.

Gaussian elimination is generally faster and more efficient for solving, particularly for larger systems. Cramer's rule is $O(n^4)$, where Gaussian Elimination is $O(n^3)$.

Cramer's rule is restricted to square matrices with a non-zero determinant, while Gaussian elimination handles non-square and singular systems.

9

MATLAB

MATLAB provides the *left-division* method for solving the equation set **Ax** = **b**. The left-division method is based on Gauss elimination. To use the left-division method to solve for **x**, type `x = A\b`. For example,

```
>> A = [6, -10; 3, -4]; b = [2; 5];
>> x = A\b
x =
     7    4
```

10

MATLAB

The MATLAB command `inv(A)` computes the inverse of the matrix **A**. The following MATLAB session solves the following equations using MATLAB.

$$2x + 9y = 5$$
$$3x - 4y = 7$$

```
>>A = [2,9;3,-4];
>>b = [5;7]
>>x = inv(A)*b
x =
    2.3714
    0.0286
```

If you attempt to solve a singular problem using the `inv` command, MATLAB displays an error message.

11

Gaussian Elimination

Gaussian elimination is an algorithm used to *solve systems of linear equations*, *find the rank of a matrix*, or *compute the inverse of an invertible matrix*.

It transforms a matrix into **row echelon form** (REF) using elementary row operations based on the followings:

1) Swap one equation for another.
2) Add or subtract a multiple of one equation from another.
3) Multiply an equation by a nonzero constant.

12

Example: Solve the following system :

$$\begin{aligned} x + 2y - z &= 3 \\ 2x + y - 2z &= 3 \\ -3x + y + z &= -6. \end{aligned}$$

Solution: The augmented matrix:

$$\left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 2 & 1 & -2 & 3 \\ -3 & 1 & 1 & -6 \end{array}\right]$$

Step 1: Eliminate the column 1:

$$\left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 2 & 1 & -2 & 3 \\ -3 & 1 & 1 & -6 \end{array}\right] \xrightarrow{\text{subtract } 2 \times \text{ row 1 from row 2}} \left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 0 & -3 & 0 & -3 \\ -3 & 1 & 1 & -6 \end{array}\right]$$

$$\xrightarrow{\text{subtract } -3 \times \text{ row 1 from row 3}} \left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 0 & -3 & 0 & -3 \\ 0 & 7 & -2 & 3 \end{array}\right]$$

13

Step 2: Eliminate the column 2:

$$\left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 0 & -3 & 0 & -3 \\ 0 & 7 & -2 & 3 \end{array}\right] \xrightarrow{\text{subtract } -\frac{7}{3} \times \text{ row 2 from row 3}} \left[\begin{array}{rrr|r} 1 & 2 & -1 & 3 \\ 0 & -3 & 0 & -3 \\ 0 & 0 & -2 & -4 \end{array}\right]$$

Re-write the equations and solve the unknowns:

$$\begin{aligned} x + 2y - z &= 3 \\ -3y &= -3 \\ -2z &= -4, \end{aligned}$$

back substitution:

$$\begin{aligned} x &= 3 - 2y + z \\ -3y &= -3 \\ -2z &= -4 \end{aligned}$$

The solution is x = 3, y = 1, z = 2.

14

Steps of the Gaussian Elimination Algorithm:

(Represent the systems of equations as an augmented matrix)

Forward Elimination: Transform the matrix into upper triangular (row echelon) form:

1. **Pivot Selection**: Start at the top-left element. If it's zero, swap with a row below where the element in the same column is non-zero.
2. **Normalize Pivot Row**: Make the pivot = 1 by dividing the entire row.
3. **Eliminate Below**: Subtract multiples of the pivot row from the rows below to make all entries below the pivot zero.
4. **Move to the Next Column and Repeat** for all pivot positions.

Back Substitution: Once the matrix is in upper triangular form:

1. **Solve from the bottom row up**.
2. **Substitute the known values** back into the equations above to find all variables.

15

A very simple code to implement Gaussian elimination method

```
function x = my_gauss_elimination(A, b)
    [n, m] = size(A);
    if n ~= m
        error('Matrix A must be square.');
    end
    if length(b) ~= n
        error('Vector b must have the same number of rows as A.');
    end

    Ab = [A b];

    for i = 1:n
        for j = i+1:n
            factor = Ab(j,i) / Ab(i,i);
            Ab(j,:) = Ab(j,:) - factor * Ab(i,:);
        end
    end

    x = zeros(n, 1);
    for i = n:-1:1
        x(i) = (Ab(i,end) - Ab(i,i+1:n) * x(i+1:n)) / Ab(i,i);
    end
end
```

Normalization is not compulsory
What happens to the element Ab(i ,i) is zero?

16