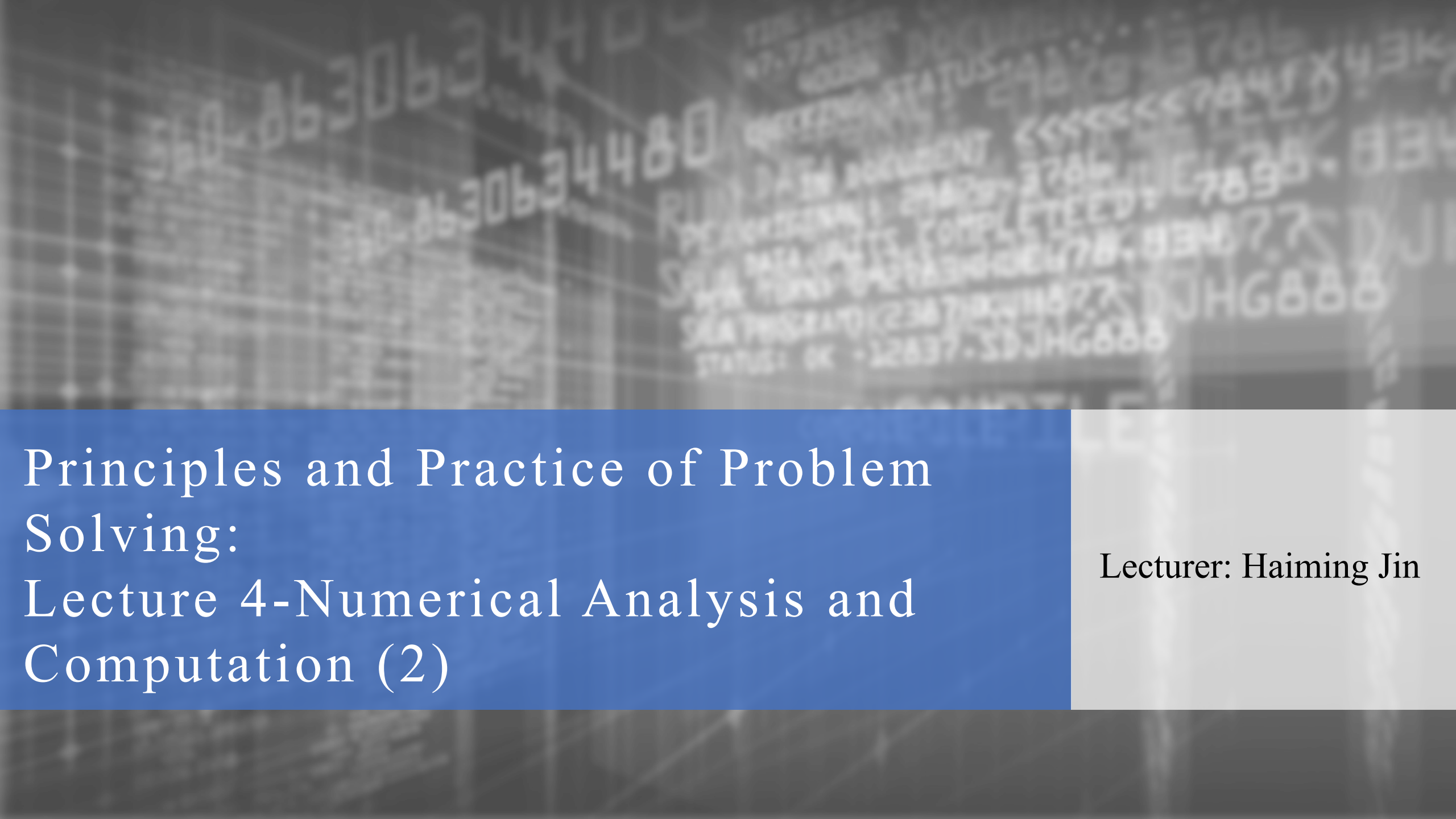


# Principles and Practice of Problem Solving: Lecture 4-Numerical Analysis and Computation (2)

Lecturer: Haiming Jin

# Solving Nonlinear Equations

- Nonlinear equation is used in many fields;
- It is typically hard to derive analytical solutions of nonlinear equations, and oftentimes they can only be solved numerically.

We focus on solving nonlinear equations in one variable.

# Typical Nonlinear Equations

## 1. Polynomials

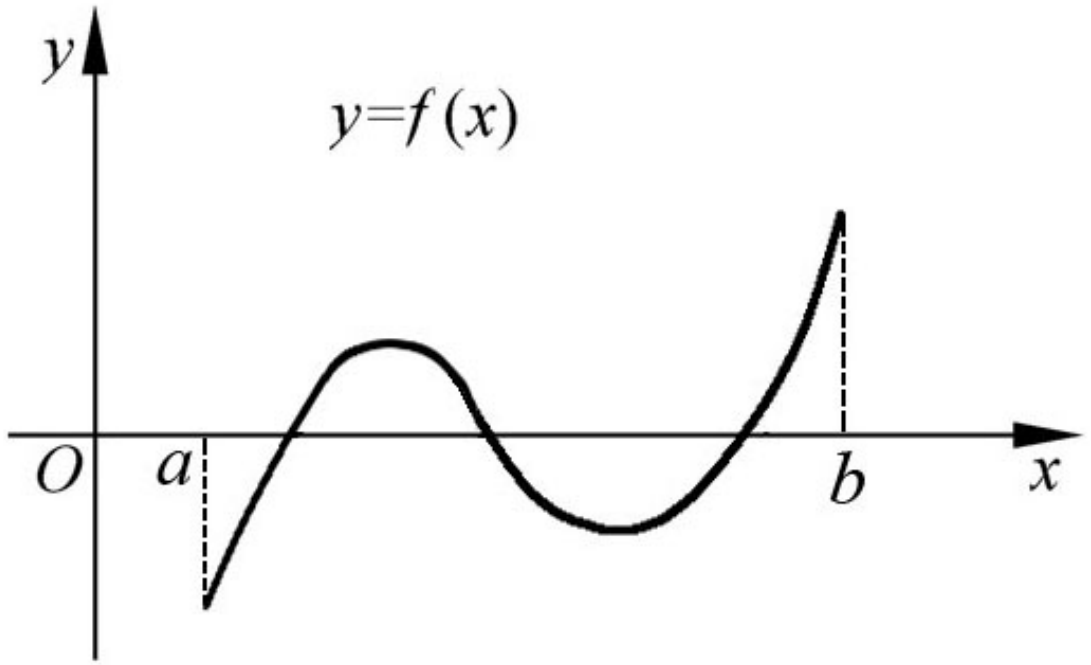
$$a_0 + a_1 x + \cdots + a_n x^n = 0$$

e.g., $x^2 - 5x + 1 = 0$

## 2. Transcendental Equation

e.g., $2^x - 5x + 2 = 0$

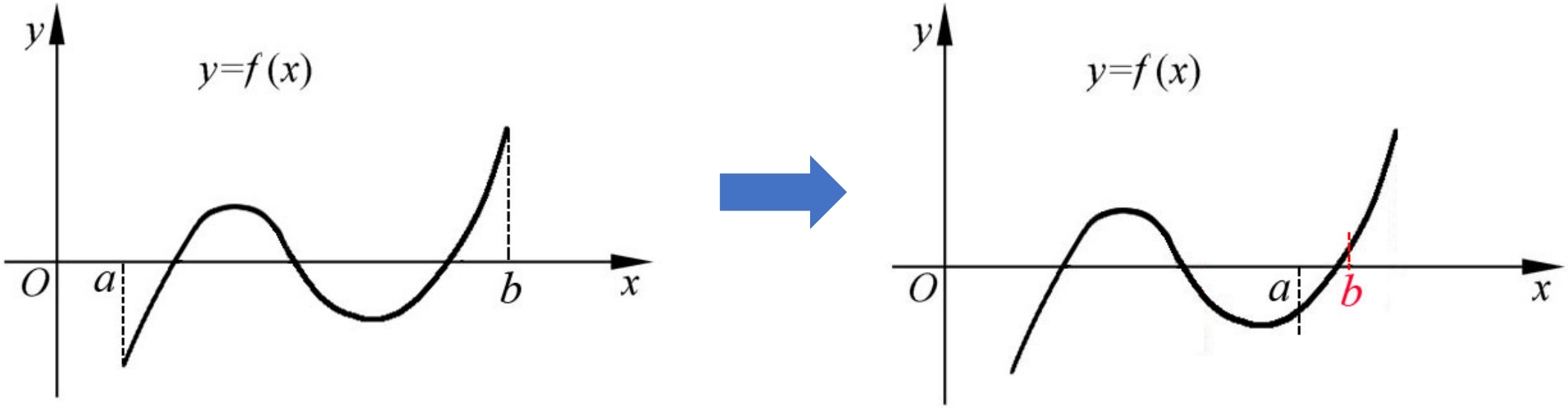
# Solving Nonlinear Equations




**Theorem** *If function* $f(x)$ *is continuous over interval* $[a, b]$*, and* $f(a)f(b) < 0$*, then function* $f(x) = 0$ *has at least one root in interval* $(a, b)$.

# Solving Nonlinear Equations-Bisection Method

Idea: By checking the signs of function values, the bisection method bisects the interval where a root must lie until the interval length is sufficiently small.



# Solving Nonlinear Equations-Bisection Method

- **The code for finding the root of f(x)=0**

```
while ( fabs(a-b)>=0.000001 )
{
        c=(a+b)/2;                  // calculate the middle point
        fa=f(a);                    // calculate f(a)
        fmid=f(c);                  // calculate f(c)
        if ( fmid==0 )              //If c is the root, break
             break;
        else if (fa*fmid>0)         //If f(a) and f(c) have the same sign, move a
             a=c;
        else                        //If f(a) and f(c) have different signs, move b
              b=c;
}

printf("Solution is: %6.6f",(a+b)/2);
```

# Solving Nonlinear Equations-Bisection Method

## Convergence and Error Estimate

**Theorem** *The sequence* $p_n = \frac{a_n+b_n}{2}$ *where* $[a_n, b_n]$ *is the interval generated by the bisection method in iteration* $n$*, satisfies that* $|p_n - p| \leq \frac{b-a}{2^n}$ *where* $p$ *is a zero point of the function* $f(x)$ *in* $(a, b)$.

**Proof Sketch**

$b_n - a_n = \frac{1}{2}(b_n - a_n) = \cdots = \frac{1}{2^{n-1}}(b_1 - a_1) = \frac{1}{2^{n-1}}(b - a)$.

$p \in [a_n, b_n]$ and $p_n = \frac{a_n+b_n}{2}$, so $|p_n - p| \leq \frac{1}{2}(b_n - a_n) \leq \frac{b-a}{2^n}$.

**Corollary** $\lim_{n\to+\infty} p_n = p$.

# Solving Nonlinear Equations-Bisection Method

## Pros

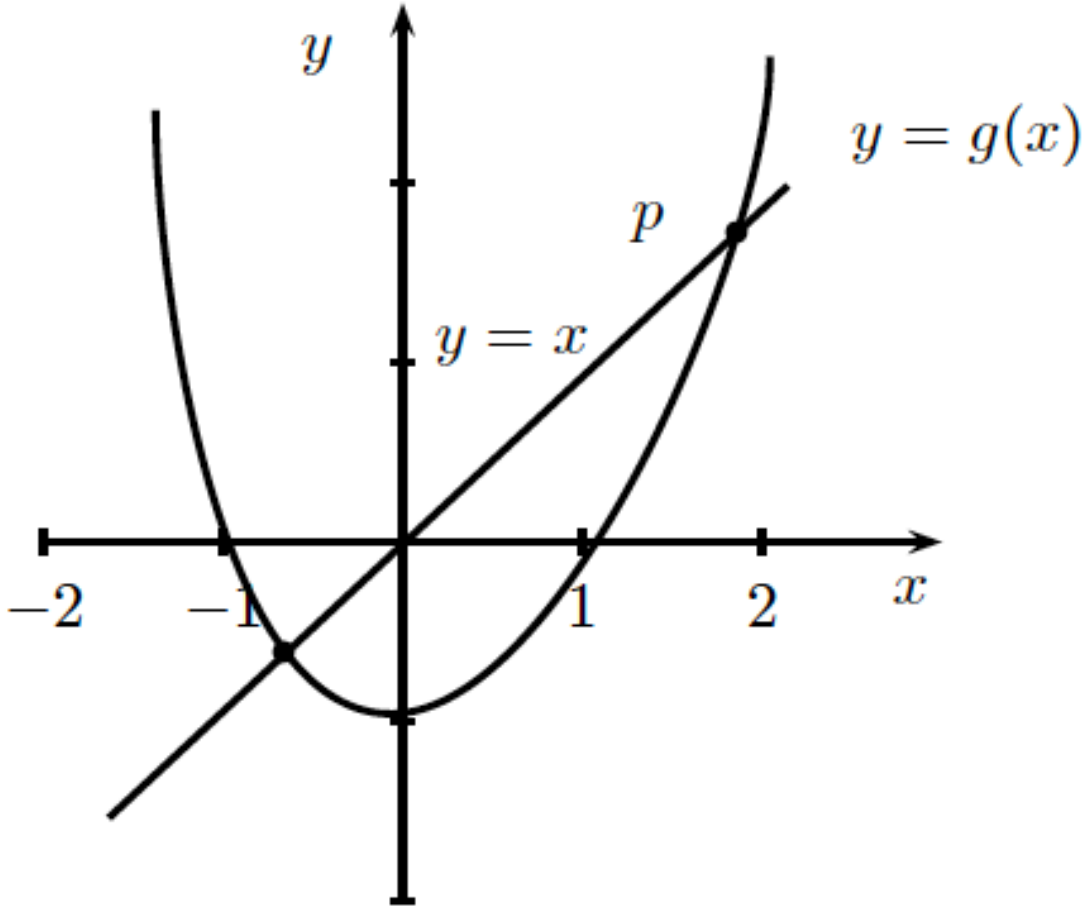
- Simple and easy to implement.
- One function evaluation per iteration.
- No knowledge of the derivative is needed.

## Cons

- Have to find an interval $(a, b)$ such that $f(a)f(b) < 0$.
- The error depends on $b - a$.
- Hard to generalize to multi-variable non-linear equations.

# Solving Nonlinear Equations-Fixed Point Iteration

**Definition** *$p$ is a fixed point of function $y = g(x)$ if $g(p) = p$.*



# Solving Nonlinear Equations-Fixed Point Iteration

Finding a root of $f(x)$ is equivalent to finding a fixed point of any of the following $g(x)$,

$g(x) = x - f(x), g(x) = x + 3f(x), g(x) = x - h(x)f(x)$ with $h(x) \neq 0, \ldots$

## Advantages of using fixed point

- Simple to analyze.
- Freedom to choose $h(x)$.
- Certain fixed point iteration methods lead to powerful root-finding techniques.

# Solving Nonlinear Equations-Fixed Point Iteration

## Examples

- Find the fixed point of $g(x) = x^2 - 2$.

$$g(x) = x \Leftrightarrow x^2 - 2 = x \Leftrightarrow x^2 - x - 2 = 0 \Leftrightarrow x = -1, x = 2.$$

- Find the fixed point of $g(x) = x^2 + 2$.

$$g(x) = x \Leftrightarrow x^2 + 2 = x \Leftrightarrow x^2 - x + 2 = 0. \quad \text{No solution!}$$

## Questions

- Conditions for the existence of a fixed point?
- Conditions for the existence of a **unique** fixed point?

# Solving Nonlinear Equations-Fixed Point Iteration

**Theorem (Existence and Uniqueness of Fixed Point)**

*(Existence) If*

*1.* $g(x) \in C[a,b]$ *(i.e.,* $g(x)$ *is continuous in* $[a,b]$*).*

*2.* $g(x) \in [a,b]$.

*then* $g(x)$ *has a fixed point in* $[a,b]$.

*(Uniqueness) In addition to assumptions 1 and 2, if*

*3.* $g'(x)$ *exists on* $(a,b)$.

*4. There exists a* $k \in [0,1]$ *with* $|g'(x)| \leq k$ *for all* $x \in [a,b]$.

*Then, the fixed point in* $[a,b]$ *is unique.*

**Proof Sketch**

Existence: Intermediate Value Theorem.

Uniqueness: Mean Value Theorem.

# Solving Nonlinear Equations-Fixed Point Iteration

## Fixed point iteration algorithm

- Suppose function $g(x)$ has a fixed point at $p$. How to find a fixed point of $g(x)$?
- Given an initial guess of $p_0$, fixed point algorithm iteration algorithm does

$$p_{n+1} = g(p_n).$$

**fixed point iteration**

## Questions

- Does the sequence $\{p_n\}$ converge?
- If it converges, what is $\lim_{n\to\infty} p_n$?

# Solving Nonlinear Equations-Fixed Point Iteration

**Theorem (Fixed-Point Theorem)**

*If*

*1.* $g(x) \in C[a,b]$.

*2.* $g(x) \in [a,b]$.

*3.* $g'(x)$ *exists on* $(a,b)$.

*4. There exists a* $k \in [0,1]$ *with* $|g'(x)| \leq k$ *for all* $x \in [a,b]$.

*Then, for any number* $p_0 \in [a,b]$, *the sequence* $p_{n+1} = g(p_n)$ *converges to the unique fixed point of* $g(x)$ *in* $[a,b]$.

# Solving Nonlinear Equations-Newton's Method

## Derivations

Suppose $p(x) \in C^2[a,b]$, and we have an that is **close** to the root $p$ of $f(x)$.

How to get an $x_{k+1}$ that is **closer** to the root $p$?

- Consider Taylor's expansion, there exists a $\xi$ between $p$ and $x_k$ such that

$$0 = f(p) = f(x_k) + (p - x_k)f'(x_k) + \frac{(p-x_k)^2}{2} f''(\xi).$$

- As $|p - x_k|$ is assumed small, $|p - x_k|^2 \ll |p - x_k|$. We neglect the term with $(p - x_k)^2$, and have

$$0 \approx f(x_k) + (p - x_k)f'(x_k).$$

Thus,

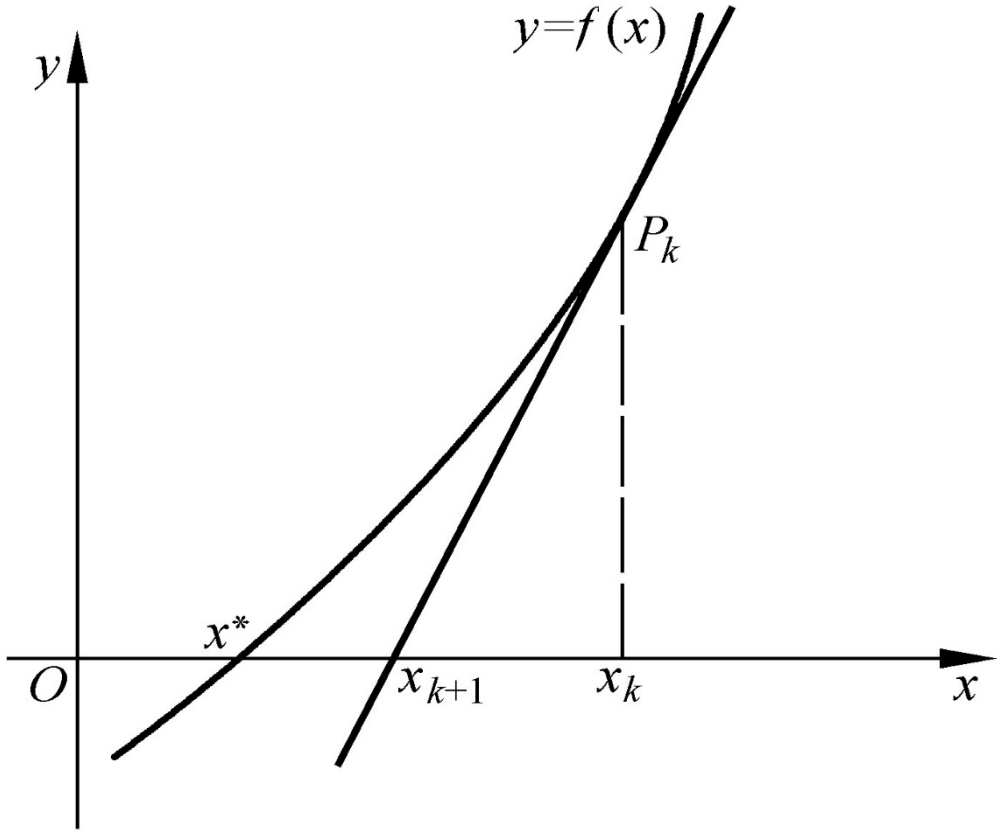
$$p \approx x_k - \frac{f(x_k)}{f'(x_k)}.$$

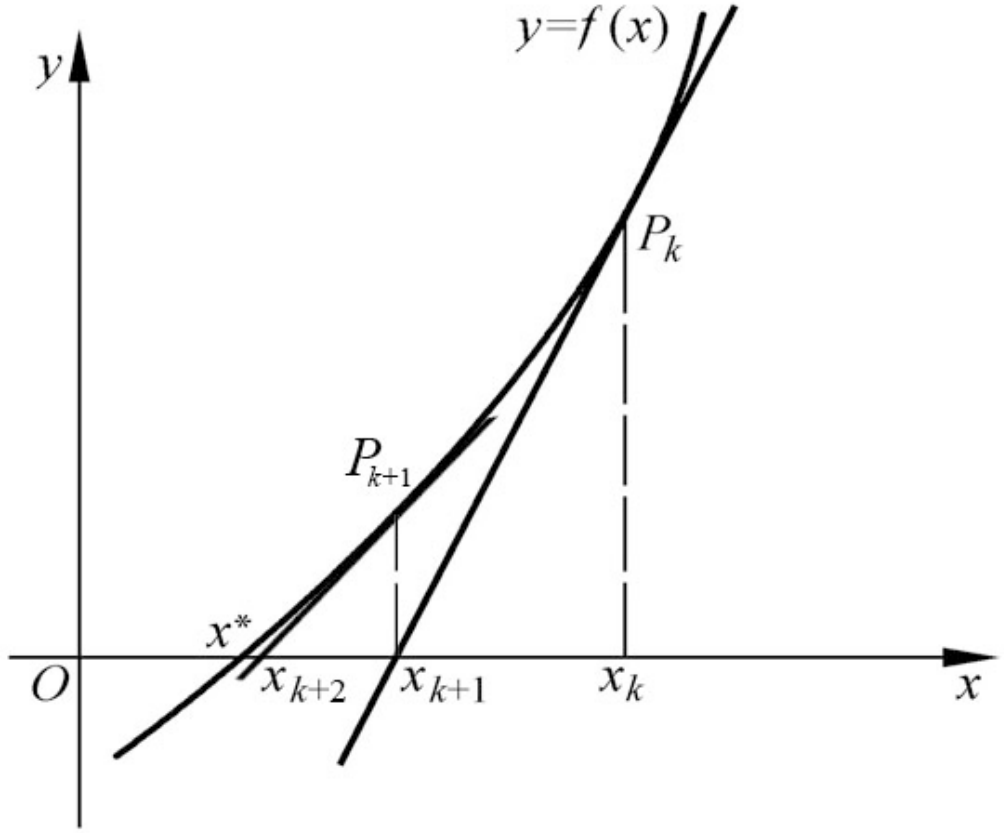
# Solving Nonlinear Equations-Newton's Method

Take any value in $[a, b]$ as $x_0$, use the iterative formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

to obtain an approximate solution.



# Solving Nonlinear Equations-Newton's Method

Take any value in $[a, b]$ as $x_0$, use the iterative formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

to obtain an approximate solution.

## Questions

- Does the sequence $\{x_k\}$ converge?
- If it converges, what is $\lim_{k\to\infty} x_k$?

## Remark

- Newton's method is equivalent to conducting fixed point iteration with function

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

# Solving Nonlinear Equations-Newton's Method

**Theorem**

*Let* $f(x) \in C^2[a,b]$. *If* $p \in (a,b)$ *satisfies that* $f(p) = 0$ *and* $f'(p) = 0$, *then there exists a* $\delta > 0$ *such that the Newton's method generates a sequence* $\{x_k\}$ *that converges to* $p$ *for any initial guess* $x_0 \in [p-\delta, p+\delta]$.

**Proof Sketch:** Fixed point theorem.

■ **Remark**

- Newton's method converges only when $x_0$ is close enough to $p$.
  - Need a good $x_0$.
  - For example, we use bisection to find a good $x_0$.
- It converges fast, it it converges.
- Need $f'(x)$
  - Secant method: Use difference to replace the derivative.

# Solving Linear Systems

Given a linear system $Ax = b$, where the coefficient matrix $A$ is invertible, and diagonal elements $a_{00}, a_{11}, \cdots, a_{(n-1)(n-1)}$ are non-zero, calculate $x$.

- **Iterative Method**

  Jacobi iteration, relaxation methods, …

- **Direct Method**

  Gaussian elimination, determinant method, …

# Jacobi Iteration Method

## ◆ Mathematical derivations

We decompose the coefficient matrix of $Ax=b$ as $A=(A-D)+D$

where $D=\begin{bmatrix} a_{00} & & & \\ & a_{11} & & \\ & & \cdots & \\ & & & a_{(n-1)(n-1)} \end{bmatrix}$

Rewrite the original equation as: $Dx=(D-A)x+b \Rightarrow x=(I-D^{-1}A)x+D^{-1}b$

Then, we have the Jacobi iteration formula: $x^{(k+1)}=(I-D^{-1}A)x^{(k)}+D^{-1}b$

Initialize $x^{(0)}=(x_0{}^{(0)},x_1{}^{(0)},\cdots,x_{n-1}{}^{(0)})^T$

Rewrite the Jacobi iteration formula as $x_i{}^{(k+1)}=\frac{1}{a_{ii}}[b_i-\sum_{\substack{j=0\\ j\neq i}}^{n-1}a_{ij}x_j{}^{(k)}] \quad i=0,1,\cdots,n-1$

# Jacobi Iteration Method

## ◆ Code

Given that we store the coefficients in a 2-D array a[][], and constants in the 1-D array b[], implement the Jacobi iteration method according to the following equation:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{\substack{j=0 \\ j\neq i}}^{n-1} a_{ij}x_j^{(k)}]$$

```
//store the solution in the previous iteration in array x2[]
//store the solution in the previous iteration in array x[]
for( i=0; i<n; i++)  {
    tmp=0.0;
    for(j=0;  j<n;   j++)  {
        if(j==i) continue;
        tmp+=a[i][j]*x2[j];
    }
    x[i]=(b[i]-tmp)/a[i][i];
}
```

$i$ corresponds to $x_i$

# Gaussian Elimination

Gaussian elimination transforms the coefficient matrix A into an **upper triangular matrix**, and solve the linear system defined by the upper triangular matrix using **back substitution**.

That is, we transform a linear system to the following form:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \cdots & \cdots \\ & & & a_{nn}^{(n)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

Back substitution:

$$\begin{cases} x_n = b_n^{(n)} \Big/ a_{nn}^{(n)} \\ x_i = \left(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j\right) \Big/ a_{ii}^{(i)} \qquad (i = n-1, \cdots, 2, 1) \end{cases}$$

# Gaussian Elimination

Deal with *i*th row

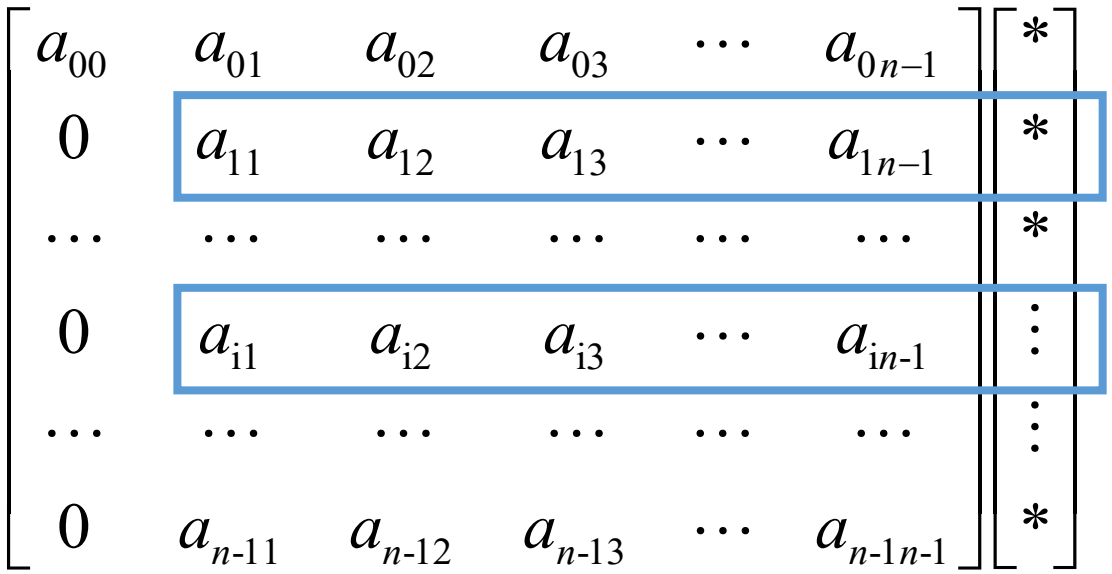
```
for(i=1; i<n; i++)
{
    temp=A[i][0]/A[0][0];
    for(j=0; j<n; j++)
    {
        A[i][j]=A[i][j]-temp*A[0][j];
    }
    b[i]=b[i]-temp*b[0];
}
```

*i*th row − 0th row* $a_{i0}/a_{00}$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & \cdots & a_{0n-1} \\ a_{10} & a_{11} & a_{12} & a_{13} & \cdots & a_{1n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i0} & a_{i1} & a_{i2} & a_{i3} & \cdots & a_{in-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n-10} & a_{n-11} & a_{n-12} & a_{n-13} & \cdots & a_{n-1n-1} \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ \vdots \\ \vdots \\ * \end{bmatrix}$$

# Gaussian Elimination

$i$th row − 1st row $*a_{i1}/a_{11}$




Deal with $i$th row

```
for(i=2; i<n; i++)
{
    temp=A[i][1]/A[1][1];
    for(j=1; j<n; j++)
    {
        A[i][j]=A[i][j]-temp*A[1][j];
    }
    b[i]=b[i]-temp*b[1];
}
```

# Gaussian Elimination

```
for(i=1; i<n; i++)  {
    temp=A[i][0]/A[0][0];
    for(j=0; j<n; j++)  {  A[i][j]=A[i][j]-temp*A[0][j];  }
    b[i]=b[i]-temp*b[0];
}
```

$$\begin{bmatrix} \# & \# & \# & \# \\ 0 & \# & \# & \# \\ 0 & \# & \# & \# \\ 0 & \# & \# & \# \end{bmatrix}$$

```
for(i=2; i<n; i++)  {
    temp=A[i][1]/A[1][1];
    for(j=1; j<n; j++)  {  A[i][j]=A[i][j]-temp*A[1][j];  }
    b[i]=b[i]-temp*b[1];
}
```

$$\begin{bmatrix} \# & \# & \# & \# \\ 0 & \# & \# & \# \\ 0 & 0 & \# & \# \\ 0 & 0 & \# & \# \end{bmatrix}$$

```
for(i=3; i<n; i++) {
    temp=A[i][2]/A[2][2];
    for(j=2; j<n; j++)  {  A[i][j]=A[i][j]-temp*A[2][j];  }
    b[i]=b[i]-temp*b[2];
}
```

$$\begin{bmatrix} \# & \# & \# & \# \\ 0 & \# & \# & \# \\ 0 & 0 & \# & \# \\ 0 & 0 & 0 & \# \end{bmatrix}$$

# Gaussian Elimination

```
for(k=0;k<n-1;k++)
{
        if(!A[k][k])    return -1;
        //elimination process
        for(i=k+1;i<n;i++)
        {
                temp=A[i][k]/A[k][k];
                for(j=k;j<n;j++)
                {  A[i][j]=A[i][j]-temp*A[k][j];     }
                b[i]=b[i]-temp*b[k];
        }
}
```

$a_{k+1k}, a_{k+2k}, \cdots, a_{(n-1)k}$ are set as zeros

# After Class

- Secant method for solving non-linear equations?
- Conditions that ensure the convergence of Newton's method given any initial point?
- Pivoting techniques for solving linear systems?

# References and Readings

- **Richard L. Burden, J. Douglas Faires, "Numerical Analysis" (9th Edition)**
  - Chapter 2 Solutions of Equations in One Variable
  - Chapter 6 Direct Methods for Solving Linear Systems