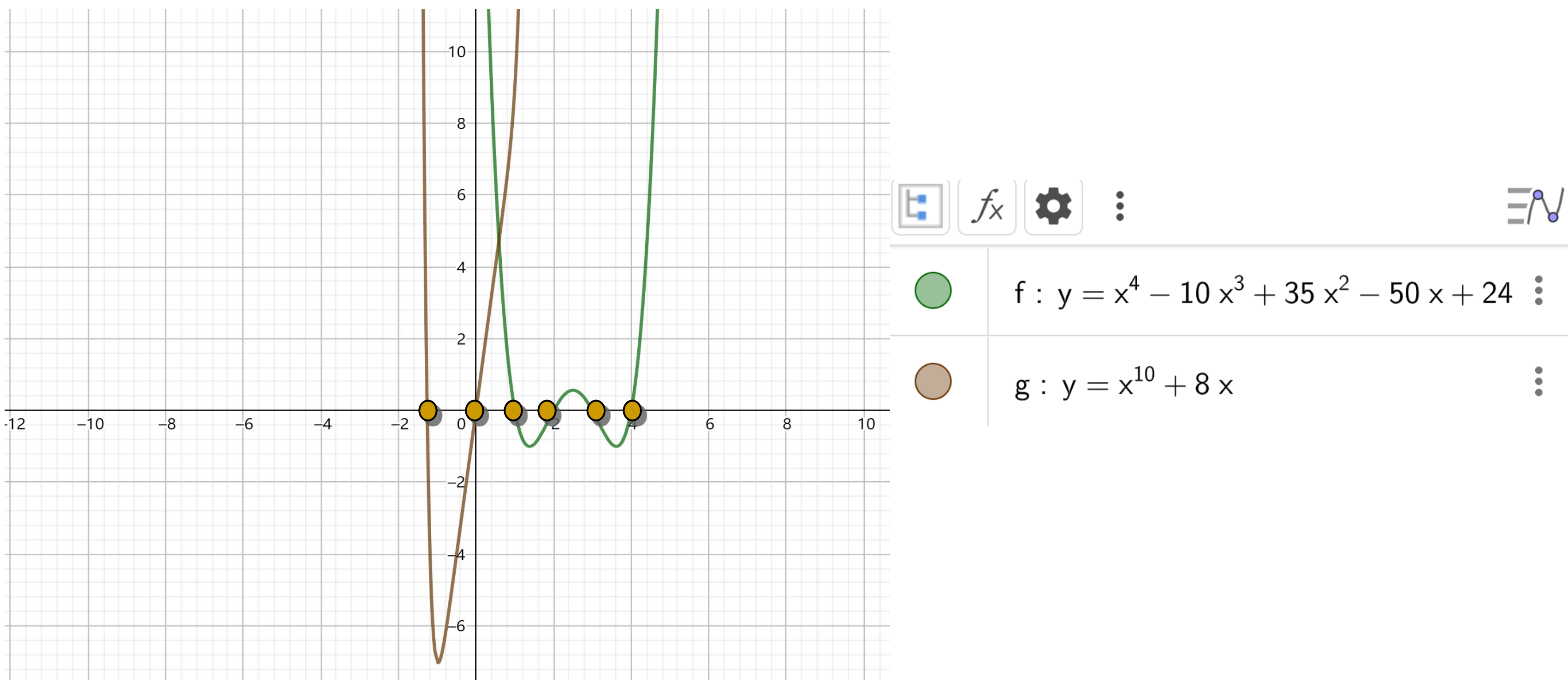
# 数值计算模块四

## 非线性方程求根

# 非线性方法求根

- 非线性科学是当今科学发展的一个重要研究方向，而非线性方程的求根也成了一个不可缺的内容。但是，非线性方程的求根非常复杂。



# 非线性方法求根

求代数方程

$$x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$$

及超越方程

$$e^{-x} - \sin(\frac{n\pi}{2}) = 0$$

的解。

- **定理：** 高于4次的代数方程无精确的求根公式。

# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

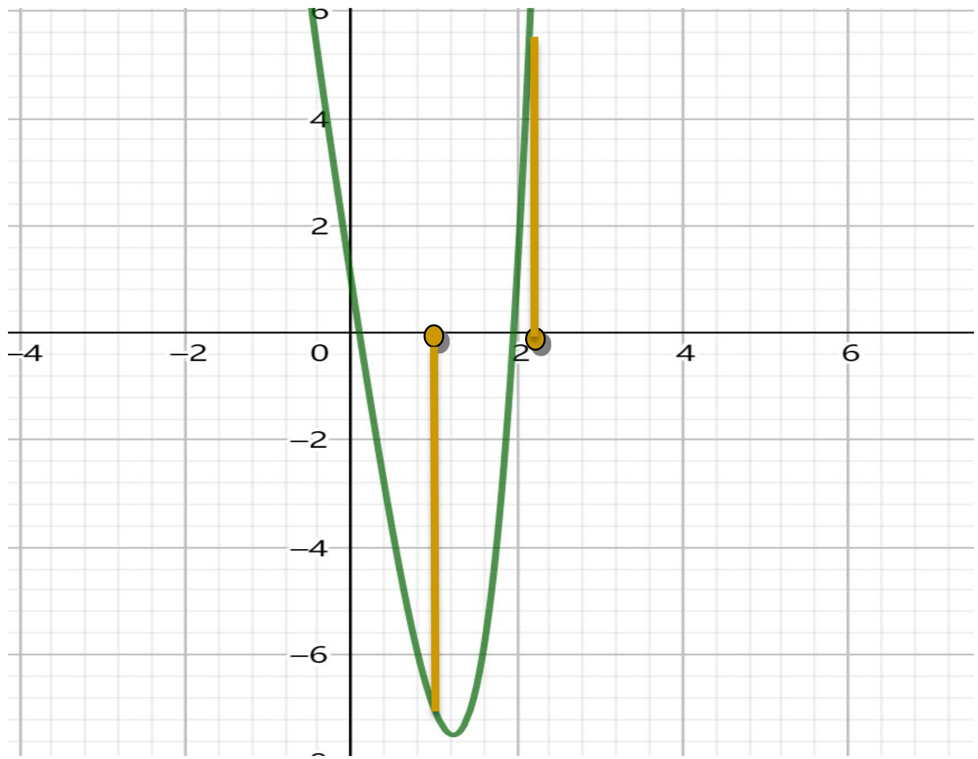
# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

# 二分法

- **用近似方法求方程的根需要知道方程的根所在区间**
- **定义：**如果在区间[a, b]内只有方程f(x)=0的一个根，则称[a, b]为隔根区间。

$$0 = x^4 + x^2 - 10x + 1$$



# 二分法

- 二分法的核心思想是利用介值定理：

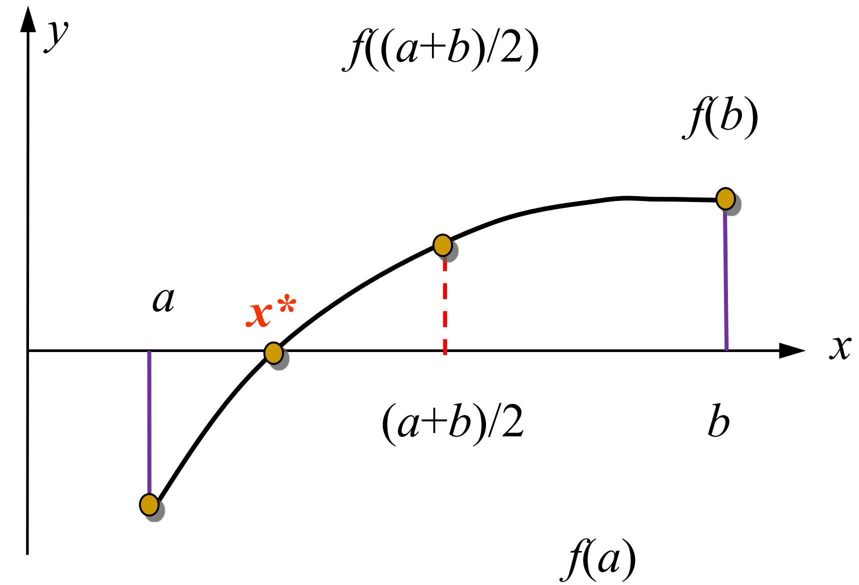
  该定理指出如果函数 $f(x)$ 在区间 $[a,b]$ 上连续，且 $f(a)$ 和 $f(b)$ 符号相反（即 $f(a)\cdot f(b)<0$），则在区间 $(a,b)$ 内至少存在一个根 $c$，使得 $f(c)=0$。

- **二分法的基本步骤：**

  通过计算隔根区间的中点，逐步将隔根区间缩小，从而可得方程的近似根。

设$f(x)=0$的隔根区间是$[a, b]$，且$f(a)<0$, $f(b)>0$。

# 二分法




(1) 输入$a$, $b$, $eps$，计算$f(a)$

(2) 计算$x=(a+b)/2$，若$f(a)\times f(x)<0$，则$b=x$，否则$a=x$

(3) 若$|a-b|<eps$，则停止循环，$x$即为所求根，否则转(2)

# 二分法

➢ **函数语句与参数说明**

**double dhrt(a, b, eps, m, f)**

| 形参与函数类型 | 参数意义 |
|---|---|
| double a | 隔根区间左边界 |
| double b | 隔根区间右边界 |
| double eps | 误差 |
| int m | 最大迭代次数 |
| double (*f)(double) | 非线性函数 |

# 二分法

```
double dhrt(double a, double b, double eps, int *m, double(*f)(double))
{

    double fa, x, fx;
    fa = (*f)(a);

    while (fabs(a - b)>=eps && *m)
    {

        x = (a + b) / 2.0;
        fx = (*f)(x);
        if (fx*fa < 0)
                b = x;
        else
                a = x;

        (*m)--;
    }

    return x;
}
```

$$x = \frac{a+b}{2}$$

$$f(a)f(b) < 0$$

$$|a-b| < eps$$

# 二分法

## ➢ 举例

求$x^3+4x^2-10=0$在[1, 2]内的根的近似解，要求绝对误差不超过0.005。

```
double f(double x)
{
    return x * x*x + 4 * x*x - 10;
}

void main()
{
        int a = 100, *m;
        m = &a;
        printf("%f\n", dhrt(1,2,0.005,m,f));
}
```

```
1.365204
请按任意键继续. . .
```

# 二分法

- **优点**
  - 运算简单，易于实现
  - 对函数$f(x)$要求不高，只要求其在$[a, b]$上连续
- **缺点**
  - 不适用于高维问题：仅适用于求解单变量非线性方程的根
  - 仅限于单根
  - 收敛速度较慢(因为每步误差是以1/2因子下降)

# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

# 不动点迭代法

- **基本思想：**

给定方程

$$f(x)= 0$$

其中$f(x)$在有根区间$[a, b]$上连续，并设$x_0$是一个近似根。

$$f(x)=0 \xleftrightarrow{\text{等价变换}} x=g(x)$$

为了求得$f(x)$的根，可由$x=g(x)$构造迭代序列

$$\begin{gathered} x_1=g(x_0), \\ x_2=g(x_1), \\ \vdots \\ x_{k+1}=g(x_k), \\ \vdots \end{gathered}$$

该方法称为迭代法，$g(x)$称为迭代函数

# 不动点迭代法

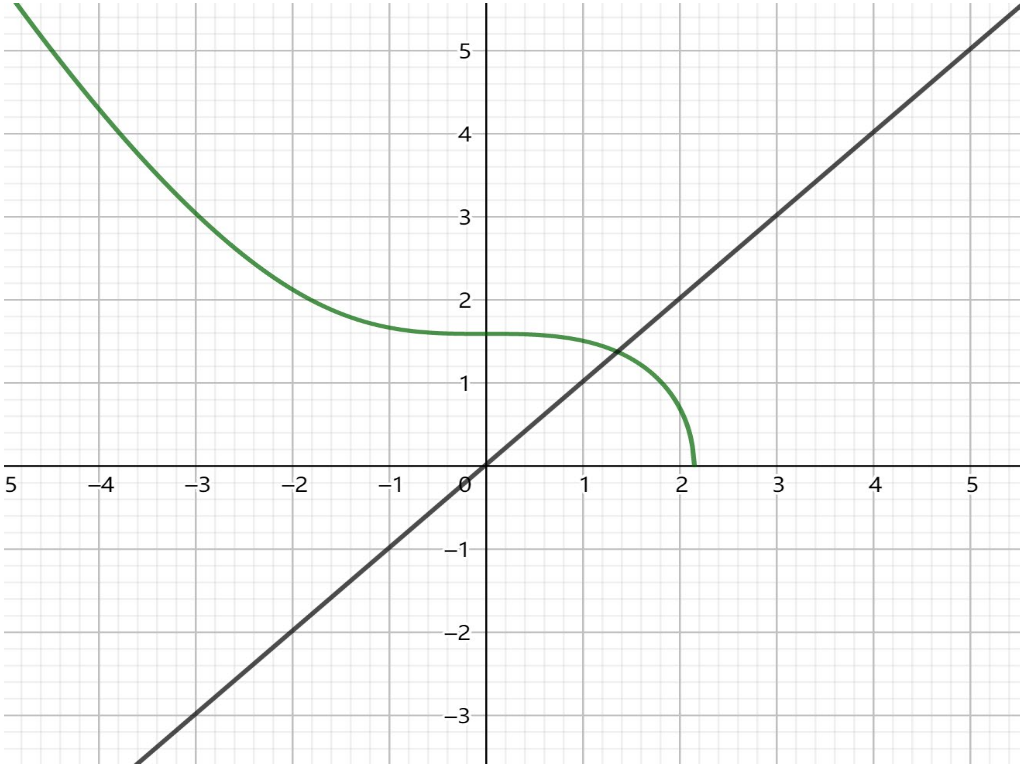
- **基本思想：**

若 $\{x_k\}_{k=0}^{\infty}$ 收敛，即存在 $x^*$ 使得 $\lim_{k\to\infty} x_k = x^*$ 则由 $\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} g(x_k)$ 可知 $x^* = g(x^*)$，即$x^*$是$g$的不动点，也就是$f$的根。

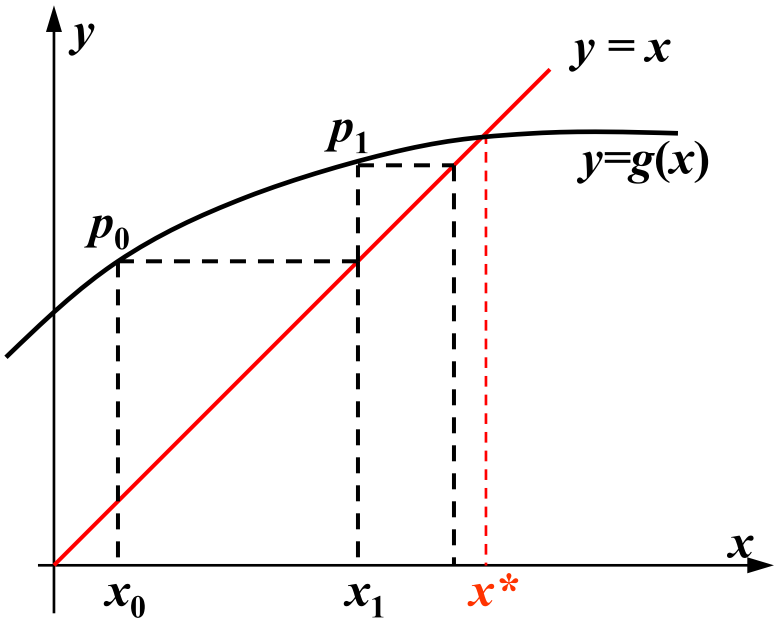
几何含义：求曲线$y=g(x)$与$y=x$的交点

# 不动点迭代法

$$x=\sqrt{\frac{10-x^3}{4}} \longrightarrow \begin{cases} y=\sqrt{\dfrac{10-x^3}{4}} \\ y=x \end{cases}$$



# 不动点迭代法




$$x_1 = g(x_0),$$
$$x_2 = g(x_1),$$
$$\vdots$$
$$x_{k+1} = g(x_k),$$

# 不动点迭代法

- **步骤**

1、给出方程的局部等价形式 $f(x)=0 \Leftrightarrow x=g(x)$

2、取合适的初值$x_0$

3、计算$x_1=g(x_0)$

4、若$|x_1-x_0|<=eps$，则停止迭代，输出$x_1$

5、若迭代次数超过m，则方法失败；否则$x_0=x_1$转3

# 不动点迭代法

➢ **函数语句与参数说明**

**double dhrt2(x0, eps, m, g)**

| 形参与函数类型 | 参数意义 |
|---|---|
| double x0 | 初值 |
| double eps | 误差 |
| int m | 最大迭代次数 |
| double (*g)(double) | 迭代函数 |

# 不动点迭代法

```c
double dhrt2(double x0, double eps, int *m, double(*g)(double))
{
	double x1;
	x1 = (*g)(x0);
	while (fabs(x1 - x0) >= eps && *m)
	{
		x0 = x1;
		x1 = (*g)(x0);
		(*m)--;
	}
	return x1;
}
```

$$
\begin{aligned}
x_1 &= g(x_0),\\
x_2 &= g(x_1),\\
&\vdots\\
x_{k+1} &= g(x_k),
\end{aligned}
$$

# 不动点迭代法

- **举例**

求$x^3+4x^2-10=0$在[1, 2]内的根的近似解，要求绝对误差不超过0.005。

- **构造迭代公式：**

方程的等价形式

$$x=((10-x^3)/4)^{0.5}$$

迭代公式为

$$x_{k+1}=((10-x_k{}^3)/4)^{0.5}$$

# 不动点迭代法

- **举例**

选取初值为1.5

```
double g(double x)
{
    return pow((10-x*x*x)/4.0, 1.0 / 2.0);
}
void main()
{
      int a = 100, *m;
      m = &a;
      printf("%f\n", dhrt2(1.5,0.005,m,g));
 }
```

```
1.363887
请按任意键继续. . .
```

# 不动点迭代法

- **优点**
  - 运算简单，易于实现
  - 若迭代公式收敛，只要迭代次数足够，可使结果达到指定精度
- **缺点**
  - 需选取合适的迭代次数

# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

# 牛顿迭代法（切线法）

➢ **基本思想：**

将$f(x)$在初值处作Taylor展开

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots$$

取线性部分作为$f(x)$的近似，有：

$$f(x_0)+f'(x_0)(x-x_0)\approx 0$$

若 $f'(x_0)\neq 0$ ，则有

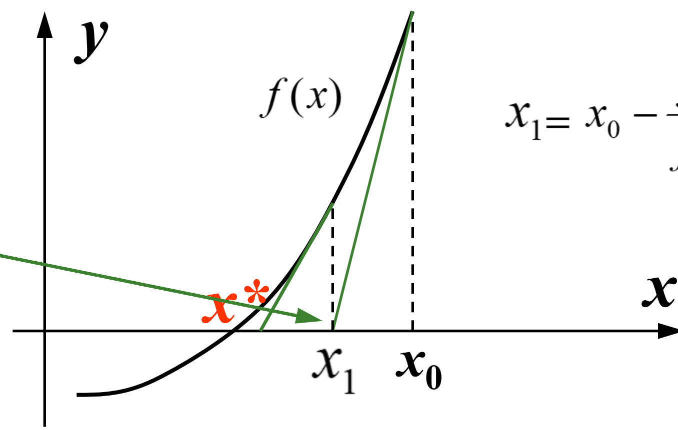
$$x=x_0-\frac{f(x_0)}{f'(x_0)}$$

记为$x_1$

类似，我们可以得到

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}$$

切线公式：$y=f(x_0)+f'(x_0)(x-x_0)$




$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$

# 牛顿迭代法

➢ **基本思想：**

这样一直下去，我们可以得到迭代序列

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

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

$$\lim_{n\to\infty} f(x^*) = 0$$

$$\lim_{n\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) = x^*$$

# 牛顿迭代法

➢ **步骤**

1、取合适的初值$x_0$，计算 $f(x_0)$，$f'(x_0)$

2、迭代得到新值

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

计算 $f(x_1)$，$f'(x_1)$

3、若$|x_1 - x_0| < eps$且$|f(x_1)| < eps$，则停止迭代，输出$x_1$

4、$x_0 = x_1$，$f(x_0) = f(x_1)$，$f'(x_0) = f'(x_1)$，转2

# 牛顿迭代法

➢ **函数语句与参数说明**

**double newt(x0, eps, m, f, ff)**

| 形参与函数类型 | 参数意义 |
|---|---|
| double x0 | 初值 |
| double eps | 误差 |
| int m | 最大迭代次数 |
| double (*f)(double) | 非线性函数函数 |
| double (*ff)(double) | 非线性函数的导数计算函数 |

# 牛顿迭代法

```
double  newt(double x0, double eps, int *m,  double(*f)(double),double(*ff)(double))
{
      double x1, f0, ff0;
      f0 = (*f)(x0);
      ff0 = (*ff)(x0);
      x1 = x0-f0/ff0;
      while ((fabs(x1 - x0) >= eps || fabs(f0) >= eps) && *m)
      {
           x0 = x1;
           f0 = (*f)(x0);
           ff0 = (*ff)(x0);
           x1 = x0 - f0 / ff0;
            (*m)--;
      }
      return x1;
}
```

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

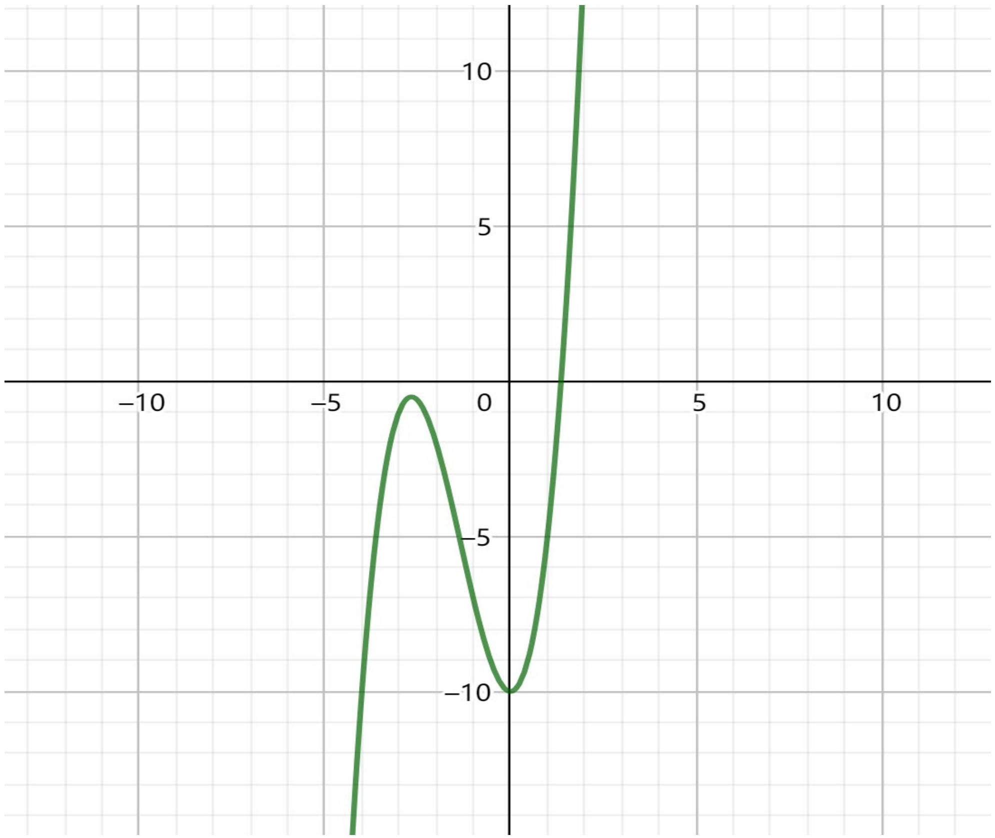
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$\lim_{n\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) = x^*$$

# 牛顿迭代法

- **举例**

求$x^3+4x^2-10=0$在[1, 2]内的根的近似解，要求绝对误差不超过0.005。




- **求导:** $f'(x)=3x^2+8x$

# 牛顿迭代法

```
1.365230
请按任意键继续. . .
```

➢ **举例**　　选取初值为1.5

```
double f(double x)
{
    return x * x*x + 4 * x*x - 10;
}
double ff(double x)
{
    return 3 * x*x + 8 * x;
}
void main()
{
    int a = 10000, *m;
    m = &a;
    printf("%f\n", newt(1.5,0.005,m,f,ff));
}
```

# 牛顿迭代法

- 优点
  - 具有二阶收敛速度

假设 $x^*$ 是方程 $f(x)=0$ 的根，且 $f(x)$ 在 $x^*$ 附近二阶连续可导。将 $f(x_n)$ 和 $f'(x_n)$ 在 $x^*$ 处进行泰勒展开：

$$f(x_n)=f(x^*)+f'(x^*)(x_n-x^*)+\frac{1}{2}f''(x^*)(x_n-x^*)^2+O((x_n-x^*)^3)$$

由于 $f(x^*)=0$，上式简化为：

$$f(x_n)=f'(x^*)(x_n-x^*)+\frac{1}{2}f''(x^*)(x_n-x^*)^2+O((x_n-x^*)^3)$$

同样，对 $f'(x_n)$ 进行泰勒展开：

$$f'(x_n)=f'(x^*)+f''(x^*)(x_n-x^*)+O((x_n-x^*)^2)$$

将这些展开式代入牛顿迭代法的公式：

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

代入展开式并化简，可以得到误差项的表达式：

$$e_{n+1}=x_{n+1}-x^*=\frac{1}{2}\frac{f''(x^*)}{f'(x^*)}(e_n)^2+O((e_n)^3)$$

- 缺点
  - 对初值非常敏感
  - 需要求导数

# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

# 弦截法

- 基本思想：将*Newton*迭代中的导数，用差商代替

$$\boxed{L_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)} = 0$$

点斜式

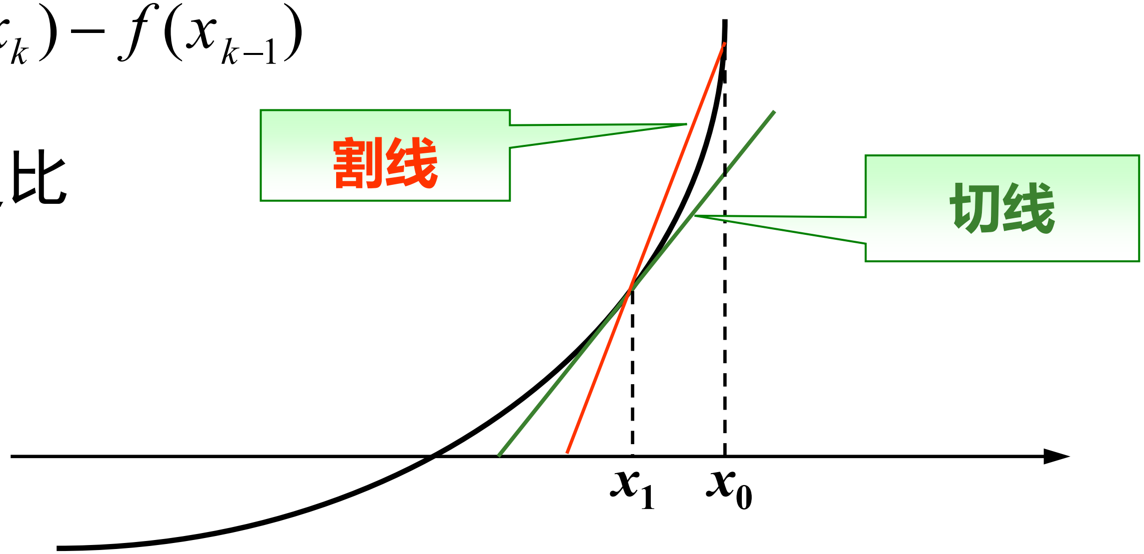
下一个猜测值

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

*Newton*迭代法：

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

是2步格式。收敛速度比*Newton*迭代慢



# 弦截法

- **步骤**

1、取合适的初值$x_0$，$x_1$，计算 $f(x_0)$，$f(x_1)$

2、迭代得到新值

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

计算 $f(x_2)$

3、若$|x_2-x_1|<eps$且$|f(x_2)|<eps$，则停止迭代，输出$x_2$

4、$x_0=x_1$，$f(x_0)=f(x_1)$，$x_1=x_2$，$f(x_1)=f(x_2)$转2

# 弦截法

- 函数语句与参数说明

**double newt(x0, x1, eps, m, f)**

| 形参与函数类型 | 参数意义 |
|---|---|
| double x0 | 初值 |
| double x1 | 初值 |
| double eps | 误差 |
| int m | 最大迭代次数 |
| double (*f)(double) | 非线性函数 |

# 弦截法

```
double newt2(double x0, double x1, double eps, int *m, double(*f)(double))
{
    double x2, f0, f1, f2;
    f0 = (*f)(x0);
    f1 = (*f)(x1);
    x2 = x1 - f1 * (x1 - x0) / (f1 - f0);
    while ((fabs(x2 - x1) >= eps || fabs(f1) >= eps) && *m)
    {
        f2 = (*f)(x2);
        x0 = x1;
        f0 = f1;
        x1 = x2;
        f1 = f2;
        x2 = x1 - f1 * (x1 - x0) / (f1 - f0);
        (*m)--;
    }
    return x2;
}
```

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

# 弦截法

## ➤ 举例

求$x^3+4x^2-10=0$在[1, 2]内的根的近似解，要求绝对误差不超过0.005。选取初值为1.5, 1.6。

```
double f(double x)
{
    return x * x*x + 4 * x*x - 10;
}
void main()
{
     int a = 10000, *m;
     m = &a;
     printf("%f\n", newt2(1.5,1.6,0.005,m,f));
}
```

```
1.365230
请按任意键继续. . .
```

# 弦截法

- **优点**
  - 不需要求导数

- **缺点**
  - 对初值非常敏感
  - 比牛顿法收敛慢

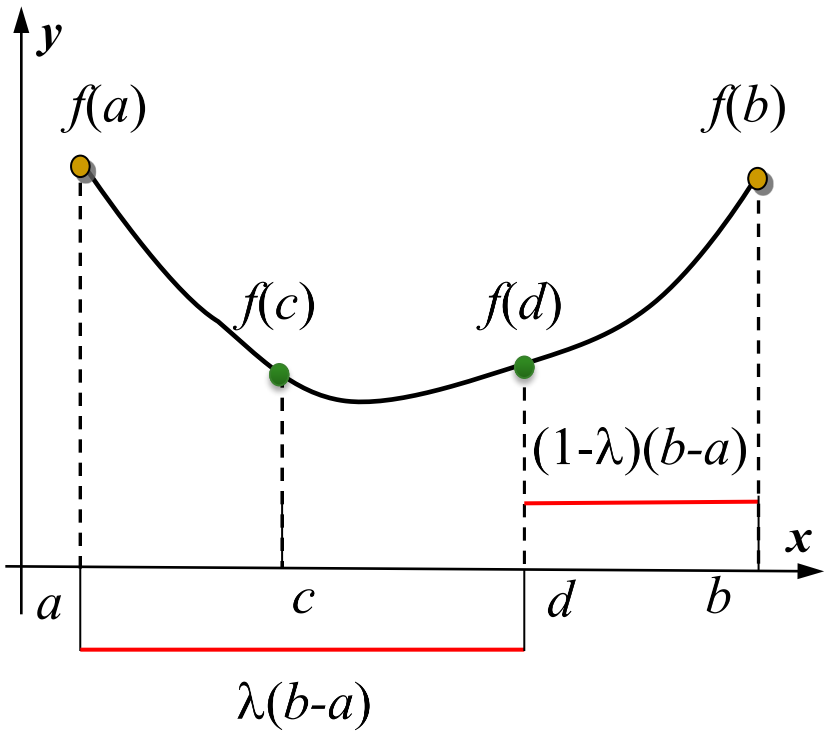
# 非线性方法求根

- 二分法
- 不动点迭代法
- 牛顿迭代法
- 弦截法
- 黄金值分割法

# ➢黄金值分割法

- **基本思想：（：把一条线段分割为两部分，较短部分与较长部分长度之比等于较长部分与整体长度之比）**

在区间$[a, b]$上取两点$c$和$d$，$c<d$使得 $\dfrac{ad}{ab}=\dfrac{db}{ad}$




$$\Rightarrow \frac{\lambda(b-a)}{b-a}=\frac{(1-\lambda)(d-a)}{\lambda(b-a)}$$

$$\Rightarrow \lambda^2+\lambda-1=0$$

$$\Rightarrow \lambda=\frac{-1\pm\sqrt{5}}{2}$$

$$\Rightarrow \because c<d$$

$$\Rightarrow \lambda=\frac{\sqrt{5}-1}{2}\approx 0.618$$

如果$f(c)<f(d)$，则抛弃db段继续搜索，否则抛弃ac段

# 黄金值分割法

- **步骤**

1、给定区间[a, b]及误差eps;

2、计算c=a+0.382(b−a), d=a+0.618(b−a);

3、若f(c)>f(d)转(4) 否则转(5);

4、若d-c<*eps*，则转(6)

否则a=c, c=d, d=a+0.618(b-a)，转(3);

5、若d-c<*eps*，则转(6)

否则b=d, d=c, c=a+0.382(b-a)，转(3)。

6、若f(c)<f(d)输出c, 否则输出d

# 黄金值分割法

- **函数语句与参数说明**

**double gss(a, b, eps, m, f)**

| 形参与函数类型 | 参数意义 |
|---|---|
| double a | 左边界 |
| double b | 有边界 |
| double eps | 误差 |
| int m | 最大迭代次数 |
| double (*f)(double) | 非线性函数 |

# 黄金值分割法

```
double gss(double a, double b, double eps, int *m, double(*f)(double))
{
	double fc,fd,c,d;
	c = a + 0.382*(b - a);d = a + 0.618*(b - a);
	fc = (*f)(c);fd = (*f)(d);
	while (d-c >= eps && *m)
	{
		if (fc > fd)
		{
				a = c;c = d;d= d = a + 0.618*(b - a);
		}
		else
		{
				b = d;d = c;c= c = a + 0.382*(b - a);
		}
		fc = (*f)(c);fd = (*f)(d);
		(*m)--;
	}
	return fc<fd?c:d;
}
```

c=a+0.382(b−a)，d=a+0.618(b−a)

如果$f(c)<f(d)$，则抛弃db段继续搜索，否则抛弃ac段

# 黄金值分割法

- **举例**

求$x^3+4x^2-10=0$在[1, 2]内的根的近似解，要求绝对误差不超过0.005。

```
double f(double x)
{
    return x * x*x + 4 * x*x - 10;
}
void main()
{
     int a = 10000, *m;
     m = &a;
     printf("%f\n", gss(1.0,2.0,0.005,m,f4));
}
```

# 实验内容

(1) 编程用不动点迭代法求解$x^3+2x^2+2x+1=0$。

(2) 编程用牛顿迭代法求$x^3-2x^2+4x+1=0$在$x=0$附近的一个实根。

(3) 编程用二分法求$x^3-6x-1=0$在$x=2$附近的一个实根。

(4) 编程用弦截法求$x^3-2x^2+7x-4=0$的一个实根，要求误差不超过$10^{-6}$。

(5) 编程用黄金值搜索法求$f(x)=x^2-4x+4$在[0,4]之间的最小值。

**作业：代码包**

# 大作业

- 将数值计算的所有功能集成为一个大系统。
- 要求：系统有不同方法的调用界面和结果展示。