零点

- 二分法
- 牛顿法
  - 收敛速度
  - 重根

方程组

- 高斯消去法
  - 三角矩阵
  - 原理
  - 误差分析
- 范数
- 迭代法
  - Jacobi迭代
  - Gauss-Seidel迭代
  - 逐次松弛迭代
- 共轭梯度法

零点

二分法

$$x_{n+1} - x^* = \frac{x_0 - x^*}{2^{n+1}}$$

牛顿法

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

收敛速度判断

由于

$$\begin{aligned}\phi(x) &= x - \frac{f(x)}{f^{(1)}(x)} \\ \phi^{(1)}(x) &= 1 - \frac{f^{(1)}(x)^2 - f(x)f^{(2)}(x)}{f^{(1)}(x)^2} \\ &= \frac{f(x)f^{(2)}(x)}{f^{(1)}(x)^2}\end{aligned}$$

且在零点 x^* 处

$$f(x^*) = 0$$

得

$$\phi^{(1)}(x^*) = 0$$

根据泰勒展开，将$\phi(x_n)$在零点x^*展开

$$\phi(x_n) = \phi(x^*) + \phi^{(1)}(x^*)(x_n - x^*) + \frac{1}{2!}\phi^{(2)}(x^*)(x_n - x^*)^2 + \ldots$$

且

$$\begin{gathered} x_{n+1} = \phi(x_n) \\ x^* = \phi(x^*) \\ \phi^{(1)}(x^*) = 0 \end{gathered}$$

得

$$\begin{gathered} x_{n+1} - x^* = \phi(x_n) - \phi(x^*) \\ \approx \phi^{(2)}(x^*)(x_n - x^*)^2 \\ \approx \phi^{(2)}(x^*)(\phi^{(2)}(x^*)(x_{n-1} - x^*)^2 - x^*)^2 \\ = \ldots\ldots \\ \approx (x_0 - x^*)^{2^{n+1}} \end{gathered}$$

即

$$x_{n+1} - x^* \approx (x_0 - x^*)^{2^{n+1}}$$

所以，牛顿法满足2阶收敛，收敛速度较快

重根问题

$f(x)$有m个重根，则$f(x)$可写作

$$f(x) = (x - x^*)^m$$

且

$$f^{(1)}(x^*) = f^{(2)}(x^*) = \ldots = f^{(m-1)}(x^*) = 0$$

此时，对于$\phi(x)$，有

$$\begin{gathered} \phi(x) = x - \frac{f(x)}{f^{(1)}(x)} \\ = x - \frac{x - x^*}{m} \\ \phi^{(1)}(x) = 1 - \frac{1}{m} \end{gathered}$$

由于$\phi^{(1)}(x)$不为0，牛顿法退化为线性收敛(二分法)

改进

可改写递推式：

$$\phi(x) = x - m\frac{f(x)}{f^{(1)}(x)} = x^*$$
$$\phi^{(1)}(x) = 0$$

方程组求解

高斯消去法

上三角矩阵U与下三角矩阵L

计算

$$Ux = b$$
$$x_k = b_k - \sum_{j=k+1}^{n} U_{k,j}x_j$$

```
for (int k = n; 1 <= k; k--)
    for (int j = k+1; j <= n; j++)
        x[k] = b[k] - U[k][j] * x[j]
```

计算

$$Lx = b$$
$$x_k = b_k - \sum_{j=1}^{k-1} L_{k,j}x_j$$

```
for (int k = 1; k <= n; k++)
    for (int j = 1; j <= k-1; j++)
        x[k] = b[k] - L[k][j] * x[j]
```

原理

对于

$$Ax = b$$

使用高斯变换，乘以下三角矩阵L

$$LAx = Lb$$
$$Ux = Lb$$
$$x = U^{-1}Lb$$

因此关键在于，将系数矩阵A转化为上三角矩阵U

$$LA = U$$

即，将矩阵A分解为下三角矩阵L和上三角矩阵之积U

$$A = LU$$

称为矩阵LU分解

$$a_{k,p} = a_{k,p} - \frac{a_{k,j}}{a_{j,j}} * a_{j,p}$$

```
gauss_lu(double* a, int n)--> void {
    for(j = 1; j <= n; ++j){
        for(k = j+1; k <= n; ++k){
            temp = a[k][j] = a[k][j]/a[j][j];
            for(p =j+1; p <= n; ++p)
                a[k][p] = a[k][p] - a[j][p] * temp;
        }
    }
}
```

列选主元高斯消去法

交换行：选取一列中最大值，然后交换。避免很大的数字对结果产生影响

```
guass_lu(double* a, int* p, int n)--> void
{
    p[1,n] = {1, 2, · · · , n};
    for(j = 1; j <= n; ++j) {
        // 找到a[j][j]到a[n][j]的最大值a[t][j]
        swap(a[j], a[t]);
        swap(p[j], p[t]);
        for(k = j+1; k <= n; ++k) {
            temp = a[k][j] = a[k][j]/a[j][j];
            for(s = j+1; s <= n; ++s)
                a[k][s] = a[k][s] - a[j][s] * temp;
        }
}
```

完全选主元高斯消去法

交换行和列

误差分析

具体内容见讲义

前向误差

由高斯法求解得到x的近似值

$$x^*$$

判断相对误差

$$\frac{\delta(x)}{x} = \frac{x^* - x}{x}$$

的上界

向后舍入误差分析

存在ΔA满足

$$(A+\Delta A)x^* = b$$

上三角矩阵方程组数组解的舍入误差分析

$$|\Delta b| = |Ux^* - b|$$

列选主元高斯消去法*LU*分解的舍入误差分析

$$|\Delta A| = |A - LU|$$

完全选主元高斯消去法*LU*分解的舍入误差分析

$$|\Delta A| = |A - LU|$$

列选主元高斯消去法解方程组舍入误差分析

$$|\Delta b| = |Ax^* - b|$$

范数

具体内容见讲义

将向量和矩阵映射到实数范围，比较大小

向量范数

壹范数、贰范数、*p*范数、无穷范数

矩阵范数

行范数(无穷范数)、列范数(壹范数)、贰范数(欧几里得范数、谱范数)、*p*范数、*F*范数

迭代法

计算

$$Ax = b$$

使用迭代法

收敛判断

对于

$$x = Gx + b$$

$$\begin{cases} x^0 = \text{随机选取} \\ x^{k+1} = Gx^k + b \end{cases}$$

上面迭代格式的充要条件是$\rho(G) < 1$

Jacobi迭代

$$
A = L + U + D \\
x = -D^{-1}(L+U)x + D^{-1}b
$$

计算格式

$$
x_0 = \text{初始向量} \\
x_{k+1} = D^{-1}(b-(L+U)x_k), k = 0,1,2...
$$

如果A主对角线元行占优(或列占优)，则上面迭代格式收敛

Gauss-Seidel迭代

$$
A = L + U + D \\
(L+D)x = -Ux + b
$$

计算格式

$$
x_0 = \text{初始向量} \\
x_{k+1} = D^{-1}(b - Ux_k - Lx_{k+1}), k = 0,1,2...
$$

如果A

1. 主对角线元占优；或
2. 对称正定

则上面迭代格式收敛

代码

```
见讲义
```

逐次松弛迭代

使用*Gauss-Seidel*迭代的求解方向，并使用过松弛以加快收敛速度

令*w*是一个实数，将x_{k+1}定义为*w*乘上*Gauss-Seidel*公式和$1-w$乘上当前估计x_k的平均。

*w*称为松弛参数，$w>1$时称为过松弛

$$
Dx_{k+1} = (1-w)Dx_k + w(b - Lx_{k+1} - Ux_k)
$$

计算格式

$$
x_0 = \text{初始向量} \\
x_{k+1} = (wL+D)^{-1}[(1-w)Dx_k - wUx_k] + w(D+wL)^{-1}, b, k = 0,1,2...
$$

如果*A*

1. 主对角线占优并且$w\epsilon(0,1]$；或者
2. 对称正定并且$w\epsilon(0,2)$

则上面迭代格式收敛

代码

```
见讲义
```

用于对称正定矩阵的方法

对称正定矩阵

对称矩阵只有一半数量的独立元素

$$A^T=A$$

是否能以一半的计算代价，并且仅仅使用一半的内存来求解

楚列斯基分解

如果*A*是对称正定矩阵，则存在上三角矩阵*R*满足

$$A=R^TR$$

共轭梯度方法

共轭

定义*A*内积

$$(v,w)_A=v^TAw$$

当$(v,w)_A=0$时，向量*v*和*w*为*A*共轭

计算格式

$$\begin{gathered}x_0=\text{初始估计}\\p_0=r_0=b-Ax_0\end{gathered}$$

$$
\begin{aligned}
&for\ k = 0, 1, 2, \ldots \\
&\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k} \\
&x_{k+1} = x_k + \alpha_k p_k \\
&r_{k+1} = r_k - \alpha_k A p_k \\
&\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \\
&p_{k+1} = r_{k+1} + \beta_k p_k
\end{aligned}
$$

向量x_k是第k步时的近似解. 向量r_k表示近似解x_k的余项

变量p_k表示用于更新x_k得到改进的x_{k+1}时所使用的新的搜索方向

对于α_k和β_k的选择

1. 选择α_k使得新的余项r_{k+1}和方向p_k正交，保证下一余项向量和前面所有的余项向量都正交，即$(r^{k+1}, r^k) = 0$

   $r^0, r^1, \ldots, r^{n-1}$是正交的
2. 选择β_k，保证p_{k+1}和p_k A共轭，即$(p_{k+1}, p_k)_A = 0$

   $p^0, p^1, \ldots, p^{n-1}$是$A$共轭的

代码

```
见讲义
```