



计算数学所linux小组第七次活动

2005年6月24日·蓝白楼报告厅

linux下C++编程实例

稀疏矩阵的存储、算法和应用

殷俊锋

ICMSEC,AMSS,CAS

yinjf@lsec.cc.ac.cn

内容提要



第一章 背 景



第二章 C++面向对象编程



第二章 稀疏矩阵的存储



第三章 稀疏矩阵的算法



第四章 C++的实现和应用



第五章 实例分析和相关软件介绍



演 示



致 谢

1 背景

👉 高性能科学计算的发展

- ★ 稀疏矩阵思想 Versus 稠密矩阵思想;
- ★ 直接算法 Versus 迭代算法;
- ★ C/C++ Versus Fortan。

👉 软件包概述

- ★  SPARSKIT
- ★  SparseLib++
- ★  pARMS
- ★ PETSc



前人的工作

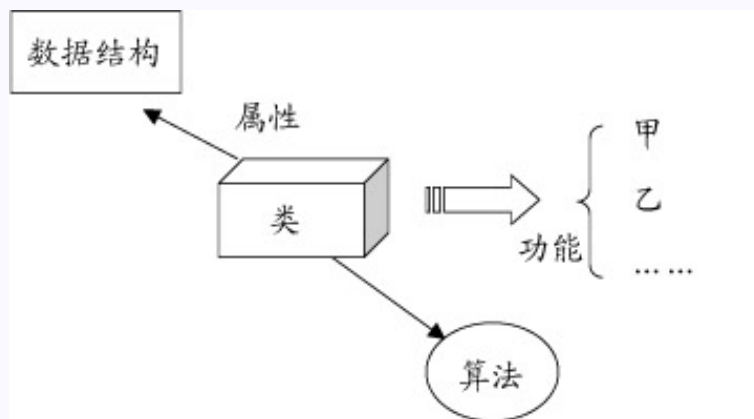
2 C++面向对象编程

思想

- ① 对象=算法+数据结构;
- ② 抽象模型—类;
- ③ 封装性和拓展性。

实现

- ① 私有属性—数据存储;
- ② 构造函数;
- ③ 析构函数。
- ④ 功能函数—算法实现。



3 一个简单的例子(A)

```
class person
{
    private:
        int hand;
        int foot;
    public:
        person() {hand = 2; foot = 2;}
        ~person(void) ;
        % 成员函数
        int swear();
        void laugh();
}
```



4 一个简单的例子(B)

```
% person是一个抽象类  
% person本身不能使用  
% 可以用person定义实例  
person ZhangSan;  
person LiSi;  
% 张三、李四都有person的属性和  
功能  
ZhangSan.laugh();  
LiSi.swear();
```



5 稀疏矩阵的存储

☒ 最常见的存储方法

- ① 坐标存储法(Coordinate Storage);
- ② 压缩稀疏行法(Compressed Sparse Row Storage);
- ③ 压缩稀疏列法(Compressed Sparse Column Storage)。

☒ 优缺点

- ① 顺序和排序;
- ② 存取矩阵元素;
- ③ 其它存储方法。

6 坐标存储法

✈ 一个非对称稀疏矩阵(n^2)

$$A = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 0 & 4 & 5 & 0 \\ 6 & 0 & 7 & 8 \\ 0 & 9 & 0 & 10 \end{pmatrix}$$

✈ 最直接简单的存储——坐标存储法($3nnz$)

val	1	2	3	4	5	6	7	8	9	10
row_ind	1	1	1	2	2	3	3	3	4	4
col_ind	1	2	4	2	3	1	3	4	2	5

Remark 对矩阵元素的顺序没有限制，存储表达不唯一。

7 压缩稀疏行法

⊗ 一个非对称稀疏矩阵(n^2)

$$A = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 0 & 4 & 5 & 0 \\ 6 & 0 & 7 & 8 \\ 0 & 9 & 0 & 10 \end{pmatrix}$$

⊗ 最直接行压缩存储—压缩稀疏行存储法($\leq 3nnz$)

row_ptr	1	4	6	9	11					
val	1	2	3	4	5	6	7	8	9	10
col_ind	1	2	4	2	3	1	3	4	2	4

Remark 行存取比较方便，列存取比较困难。

8 压缩稀疏列法

① 一个非对称稀疏矩阵(n^2)

$$A = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 0 & 4 & 5 & 0 \\ 6 & 0 & 7 & 8 \\ 0 & 9 & 0 & 10 \end{pmatrix}$$

② 最直接列压缩存储—压缩稀疏列存储法($\leq 3nnz$)

col_ptr	1	3	6	8	11					
val	1	6	2	4	9	5	7	3	8	10
row_ind	1	3	1	2	4	2	3	1	3	4

Remark 列存取比较方便，行存取比较困难。

9 顺序和排序

☞ 顺序

- ① C/C++数组按行存储;
- ② Fortran数组按列存储;
- ③ 压缩行、压缩列存储法具有明显的行列优势。

☞ 排序

- ① 坐标存储法需要排序;
- ② 结构排序;
- ③ 数量排序。

10 其它存储方法

✂ 结构存储方法

- ① 对称存储;
- ② 带状存储;
- ③ 块状存储。

✂ 类型存储方法

- ① 链表存储;
- ② matlab存储方法;
- ③ Harwell-Boeing类型。

11 稀疏矩阵的算法

✂ 稀疏矩阵分解

- ★ 三角分解（LU分解）
- ★ 正交分解
- ★ 不完全分解

✂ 稀疏矩阵迭代

- ★ 简单迭代
- ★ Krylove子空间迭代
- ★ 预优处理

12 三角分解(一)

✂ 不同的三角分解

- ★ Gaussian elimination

- ★ Sherman分解

- ★ Doolittle分解

- ★ Pickett,Black分解

- ★ Crout分解

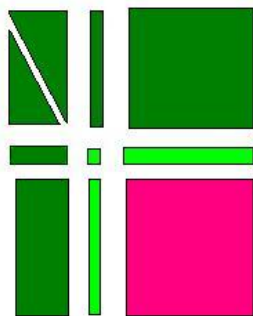
13 Gaussian elimination

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$

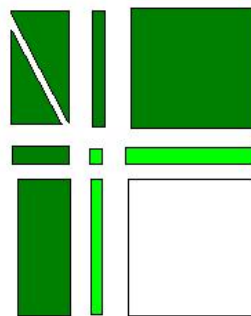
14 Sherman 分解

$$\begin{pmatrix} A_{11} & a_{1k} \\ a_{1k}^T & a_{kk} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ l_{kl}^T & a_{kk} \end{pmatrix} \begin{pmatrix} U_{11} & u_{1k} \\ 0 & u_{kk} \end{pmatrix}$$

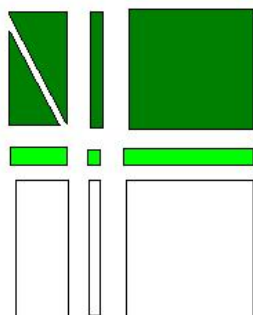
15 三角分解(二)



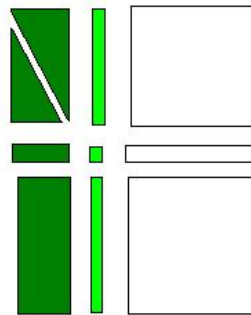
Dense Method



Crout Method



Doolittle Method



Pickett, Black Method

16 简单迭代方法

✂ 稀疏求解

$$x_0 = U^{-1}L^{-1}b;$$

where $A = LU$ is sparse factorization

$$r_0 = b - Ax_0;$$

...

$$r_{k+1} = U^{-1}L^{-1}r_k;$$

$$x_{k+1} = x_k + r_{k+1};$$

17 Krylove子空间迭代

✂ 共轭梯度法

```
template < class Matrix, class Vector, class Preconditioner, class Real >
int CG(const Matrix &A, Vector &x, const Vector &b,
const Preconditioner &M, int &iter, Real &tol) {
    Real resid;
    Vector p, z, q;
    ...
    Vector r = b - A*x;
    ...
    for (int i = 1; i != iter; i++) {
        z = M.solve(r);
        rho(0) = dot(r, z);
        ...
        q = A*p;
        alpha(0) = rho(0) / dot(p, q);
        x += alpha(0) * p;
        r -= alpha(0) * q;
        ...
    } }
```

18 C++的实现和应用(一)矩阵数据存取和转换

- 定义

```
1 CompRow_Mat_double Arow;  
2 CompCol_Mat_double Acol;  
3 Coord_Mat_double Acoord;  
4 VECTOR_double b;
```

- 读取

```
5 readtxtfile_mat(Aname, &Acoord);  
6 readtxtfile_vec(bname, &b);
```

- 转换

```
7 Arow = Acoord;  
8 Acol = Acoord;
```

19 C++的实现和应用(一)预处理迭代求解

- 构造预处理

```
1 double cpu = 1.e-6, last = 1.e-6,  
2 last = clock();  
3 CompRow_ILUPreconditioner_double Milu(Arow);  
4 cpu = (clock()-last)/CLOCKS_PER_SEC;
```

- 共轭梯度法


```
5 result = CG(Arow, cx, b, Milu, maxit, tol);
```

- GMRES

```
6 result = GMRES(Arow, cx, b, Milu, H, restart, maxit, tol);
```

20 实例分析和相关软件介绍

● 实例分析



1	1	0.95012929
2	1	1.00000000
3	1	1.00000000
1	2	0.82140716
2	2	0.38970942
3	2	1.00000000
4	2	1.00000000
1	3	0.93546970
2	3	-0.00835685
3	3	-0.81641206
4	3	1.00000000
5	3	1.00000000
1	4	0.13889088
2	4	0.74408079
3	4	-0.14691837



4.81210228
4.53098047
0.73582840
1.33172278
3.98174722
2.64151248
2.10258023
3.02695888

`./test mtest.dat btest.dat`

● 相关软件介绍

- SparseLib++ (稀疏矩阵函数库) ;
- cygwin (linux模拟器) ;
- xemacs (linux超强编辑器) 。

21 演示和展望

✌ 演示

- 数据文件的存取（数据封装）；
- 预优子空间算法（实现简单）。

✌ 未来的工作

- 存储的优化；
- 算法的改进；
- 并行化；
- 需要实际应用来检验。

Thank you! Thank you!
Thank you! Thank you! Thank you!
Thank you! Thank you! Thank you! Thank you! Thank
you! Thank you! Thank you! Thank you! Thank you!
Thank you! Thank you! Thank you! Thank you! Thank
you! Thank you! Thank you! Thank you! Thank
you! Thank you! Thank you! Thank you! Thank
you! Thank you! Thank you! Thank you!
Thank you! Thank you! Thank you!
Thank you! Thank you! Thank
you! Thank you! Thank
you! Thank
you!

感谢大家的参与！感谢崔涛的组织！欢迎提问！

Email: yinjf@lsec.cc.ac.cn