稀疏矩阵简介

科学计算小组学习报告

2010年3月

稀疏矩阵与偏微分方程数值解

$m \times n$矩阵M中有s个非零元素。$e = \frac{s}{m*n}$称为矩阵的稀疏因子。通常认为$e \leq 0.05$时称之为稀疏矩阵。稀疏矩阵的一个重要产生途径是偏微分方程数值解， 本质上是由离散空间的基函数支集的局部性造成的

稀疏矩阵与偏微分方程数值解

$m \times n$矩阵M中有s个非零元素。$e = \frac{s}{m*n}$称为矩阵的稀疏因子。通常认为$e \le 0.05$时称之为稀疏矩阵。稀疏矩阵的一个重要产生途径是偏微分方程数值解， 本质上是由离散空间的基函数支集的局部性造成的：

- 三对角矩阵：一维问题的中心差分或线性元；
- 块三对角矩阵： 基于结构网格的差分或有限元；
- 一般的稀疏矩阵：基于非结构网格的有限元法或有限体积法离散， 每一行非零元的个数与网格的分布以及有限元空间结构都有关。

图论中与稀疏矩阵相关的一些知识

- 邻接图：对应n阶稀疏矩阵M存在一个与之对应的图G， 其定点为$V = 1, 2, \cdots, n$，同时每个$a_{i,j} \neq 0$对应与G中的一条边$(i, j) \in E$,
- 带宽(bandwidth)：邻接图中任何相邻两点(以规则f)编号之差的最大值:

$$B_f(G) := \max |f(v) - f(u)| : uv \in E,$$

- 轮廓宽(profile width)P_j: 对所有的$k < i < j$而言，$a_{kj} = 0, a_{i,j} \neq 0$， 那么$P_j = j - i$。很明显，带宽是轮廓宽中的最大值。矩阵的轮廓称为skyline。
- 稀疏矩阵行列Cuthill-McKee排序算法能减小带宽。求带 宽的最小值是一个完全NP问题 (Papadimitriou, 1976; Garey, Johnson & Stockmeyer, 1974; Lin & Yuan, 1994)。

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;
- 坐标存储法，Intel MKL Sparse BLAS支持；

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;
- 坐标存储法，Intel MKL Sparse BLAS支持；
- CSR(Compressed Sparse Row)存储法，Intel MKL Sparse BLAS、deal.II、AFEPack等支持；

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;
- 坐标存储法，Intel MKL Sparse BLAS支持；
- CSR(Compressed Sparse Row)存储法，Intel MKL Sparse BLAS、deal.II、AFEPack等支持；
- 轮廓线（Skyline）存储法，MKL支持；

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;
- 坐标存储法，Intel MKL Sparse BLAS支持；
- CSR(Compressed Sparse Row)存储法，Intel MKL Sparse BLAS、deal.II、AFEPack等支持；
- 轮廓线（Skyline）存储法，MKL支持；
- Sherman's存储法：往往用于直接法，Fill-in、Reordering；

几种常用的存储格式

- 对角线存储法,只存储n条对角线元素;
- 压缩矩阵存储法，也叫Ellpack-Itpack存储法;
- 坐标存储法，Intel MKL Sparse BLAS支持；
- CSR(Compressed Sparse Row)存储法，Intel MKL Sparse BLAS、deal.II、AFEPack等支持；
- 轮廓线（Skyline）存储法，MKL支持；
- Sherman's存储法：往往用于直接法，Fill-in、Reordering；
- 超矩阵存储法：CSR存储法的分块矩阵推广。

适合Gauss消去法的单、双链表存储结构

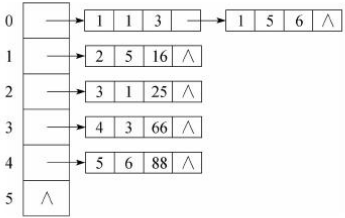

图 2-22 稀疏矩阵 A 的带行指针数组的三元组链表

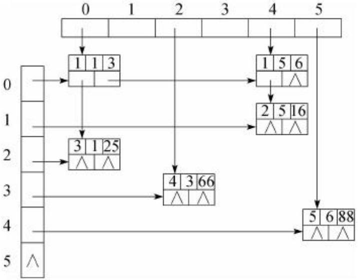

图 2-23 稀疏矩阵 A 的三元组十字链表

I、压缩矩阵存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

若$*$表示任意$1-6$的数，可以用如下的压缩存储方式：

$$AC = \begin{bmatrix} 11 & 13 & 0 & 0 \\ 22 & 21 & 24 & 0 \\ 33 & 32 & 35 & 0 \\ 44 & 43 & 46 & 0 \\ 55 & 51 & 54 & 0 \\ 66 & 61 & 62 & 65 \end{bmatrix}, KA = \begin{bmatrix} 1 & 3 & * & * \\ 2 & 1 & 4 & * \\ 3 & 2 & 5 & * \\ 4 & 3 & 6 & * \\ 5 & 1 & 4 & * \\ 6 & 1 & 2 & 5 \end{bmatrix}$$

II、坐标存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

基于坐标的存储法

$$\begin{aligned} AR &= \{11, 21, 51, 61, 22, 32, 62, 13, 33, 43, 24, 44, 54, 35, 55, 65, 46, 66\} \\ IA &= \{1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6\} \\ JA &= \{1, 2, 5, 6, 2, 3, 6, 1, 3, 4, 2, 4, 5, 3, 5, 6, 4, 6\} \end{aligned}$$

II、坐标存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

基于坐标的存储法

$$\begin{aligned} AR &= \{11,21,51,61,22,32,62,13,33,43,24,44,54,35,55,65,46,66\} \\ IA &= \{1,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6\} \\ JA &= \{1,2,5,6,2,3,6,1,3,4,2,4,5,3,5,6,4,6\} \end{aligned}$$

- Kunths的改进使得行列扫描效率大大提高;
- Rheinboldt和Mesztenyi改进到KRM循环方法,循环时间增;
- 刘万勋、刘长学等1981年的著作进一步提高KRM的效率;
- Larcombe在矩阵对称正定情况下做了适当简化。

III、CSR(Compressed Sparse Row)存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

结合前两种优点的CSR存储：

$$\begin{aligned} AR &= \{11, 13, 22, 21, 24, 33, 32, 35, 44, 43, 46, 55, 51, 54, 66, 61, 62, 65\} \\ IA &= \{1, 3, 6, 9, 12, 15, 19\} \\ JA &= \{1, 3, 2, 1, 4, 3, 2, 5, 4, 3, 6, 5, 1, 4, 6, 1, 2, 5\} \end{aligned}$$

III、CSR(Compressed Sparse Row)存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

结合前两种优点的CSR存储：

$$\begin{aligned} AR &= \{11, 13, 22, 21, 24, 33, 32, 35, 44, 43, 46, 55, 51, 54, 66, 61, 62, 65\} \\ IA &= \{1, 3, 6, 9, 12, 15, 19\} \\ JA &= \{1, 3, 2, 1, 4, 3, 2, 5, 4, 3, 6, 5, 1, 4, 6, 1, 2, 5\} \end{aligned}$$

同理也可以按列的存储格式，请自行给出。有如下特点：

- 迭代运算或与向量乘法实现简单高效（cg、gmres等）
- 矩阵转置、矩阵与矩阵的乘实现较为复杂（AMG）

IV、轮廓线存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

方案一：从对角线出发（Diagonal-Out）

$AU = \{11, 22, 33, 0, 13, 44, 0, 24, 55, 0, 35, 66, 0, 46\}$

$IDU = \{1, 2, 3, 6, 9, 12, 15\}$

$AL = \{*, *, 21, *, 32, *, 43, *, 54, 0, 0, 51, *, 65, 0, 0, 62, 61\}$

$IDL = \{1, 2, 4, 6, 8, 13, 19\}$

IV、轮廓线存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

方案一：从对角线出发（Diagonal-Out）

$AU = \{11, 22, 33, 0, 13, 44, 0, 24, 55, 0, 35, 66, 0, 46\}$

$IDU = \{1, 2, 3, 6, 9, 12, 15\}$

$AL = \{*, *, 21, *, 32, *, 43, *, 54, 0, 0, 51, *, 65, 0, 0, 62, 61\}$

$IDL = \{1, 2, 4, 6, 8, 13, 19\}$

方案二：从轮廓线出发（Profile-In）

$AU = \{11, 22, 13, 0, 33, 24, 0, 44, 35, 0, 55, 46, 0, 66\}$

$IDU = \{1, 2, 5, 8, 11, 14, 15\}$

$AL = \{*, 21, *, 32, *, 43, *, 51, 0, 0, 54, *, 65, 61, 62, 0, 0, 65, *\}$

$IDL = \{1, 3, 5, 7, 9, 12, 19\}$

IV、轮廓线存储法

$$\begin{bmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 35 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 51 & 0 & 0 & 54 & 55 & 0 \\ 61 & 62 & 0 & 0 & 65 & 66 \end{bmatrix}$$

方案一：从对角线出发（Diagonal-Out）

$AU = \{11, 22, 33, 0, 13, 44, 0, 24, 55, 0, 35, 66, 0, 46\}$

$IDU = \{1, 2, 3, 6, 9, 12, 15\}$

$AL = \{*, *, 21, *, 32, *, 43, *, 54, 0, 0, 51, *, 65, 0, 0, 62, 61\}$

$IDL = \{1, 2, 4, 6, 8, 13, 19\}$

方案二：从轮廓线出发（Profile-In）

$AU = \{11, 22, 13, 0, 33, 24, 0, 44, 35, 0, 55, 46, 0, 66\}$

$IDU = \{1, 2, 5, 8, 11, 14, 15\}$

$AL = \{*, 21, *, 32, *, 43, *, 51, 0, 0, 54, *, 65, 61, 62, 0, 0, 65, *\}$

$IDL = \{1, 3, 5, 7, 9, 12, 19\}$

Reverse Cuthill-McKee ording是很有必要的！！！

稀疏矩阵以及相关迭代法的实现

实践内容：

1. CSR、Skyline或结构网格下简化存储格式;
2. Jacobian、Gauss-Sediel、cg;
3. 转置、与其他稀疏矩阵的乘法以及AMG;

建议时限:

稀疏矩阵以及Gauss-Sediel、cg实现——1周；

AMG以及相关算法实现——1周。

欢迎访问 http://10.13.91.107/forum/的数值代数模块讨论相关问题， 参考材料见讨论小组论坛。

测试建议

以sparsematrix.pdf中的Possion方程求解为例，做如下的测试：

- 支持从数据文件导入或从偏微分方程离散；
- 比较GS迭代与cg；
- 测试自由度数为1000,...,1000000时各种迭代法的运行效率；
- 比较几种实现方式的时间、空间复杂度的优劣；
- 根据需要选择实现的方式。

That's All