第9章　稀 疏 矩 阵

在许多问题中提到了含有大量0元素的矩阵，这样的矩阵称为稀疏矩阵。比如求解普通或者部分微分方程的数值解。为了节省存储空间和计算时间，　MATLAB考虑到矩阵的稀疏性，在对它运算时有特殊的命令。

9.1　矩阵为什么稀疏

一个稀疏矩阵中有许多元素等于零，这便于矩阵的计算和保存。如果 MATLAB把一个矩阵当作稀疏矩阵，那么只需在 $m\times3$的矩阵中存储 m个非零项。第1列是行下标，第2列是列下标，第3列是非零元素值，不必保存零元素。如果存储一个浮点数要 8个字节，存储每个下标要4个字节，那么整个矩阵在内存中存储需要 $16\times m$个字节。

■ **例**9.1

```
A=eye(1000);
```

得到一个1000×1000的单位矩阵，存储它需要8 Mb空间。如果使用命令：

```
B=speye(1000);
```

用一个1000×3的矩阵来代表，每行包含有一个行下标、列下标和元素本身。现在只需16Kb的空间就可以存储1000×1000的单位矩阵，它只需要满单位矩阵的 0.2%存储空间。对于许多的广义矩阵也可这样来作。

■

稀疏矩阵的计算速度更快，因为 MATLAB只对非零元素进行操作，这是稀疏矩阵的第二个突出的优点。

■ **例**9.2

假设矩阵**A**、**B**和例9.1中的矩阵一样。计算 2*A需要一百万次的浮点运算，而计算 2*B只需要2000次浮点运算。

■

因为MATLAB不能自动创建稀疏矩阵，所以要用特殊的命令来得到稀疏矩阵，在下一节中将给出这些命令。前面章节中的算术和逻辑运算都适用于稀疏矩阵。

9.2　创建和转换稀疏矩阵

在MATLAB中，用命令`sparse`来创建一个稀疏矩阵。

命令集87　创建稀疏矩阵

`sparse(A)`	由非零元素和下标建立稀疏矩阵 **A**。如果**A**已是一个稀疏矩阵，则返回**A**本身。
`sparse(m,n)`	生成一个$m\times n$的所有元素都是0的稀疏矩阵。

`sparse(u,v,a)`	生成一个由长度相同的向量 **u**，**v**和**a**定义的稀疏矩阵。其中**u**和**v**是整数向量，**a**是一个实数或者复数向量。(u_i, v_i)对应值a_i，如果**a**中有零元素，则将这个元素排除在外。稀疏矩阵的大小为$\max(u)\times\max(v)$。
`sparse(u,v,a,m,n)`	生成一个$m\times n$的稀疏矩阵，(u_i, v_i)对应值a_i。向量**u**，**v**和**a**必须长度相同。
`sparse(u,v,a,m,n, nzmax)`	生成一个$m\times n$的含有$nzmax$个非零元素的稀疏矩阵。(u_i, v_i)对应值a_i。$nzmax$的值必须大于或者等于向量**u**和**v**的长度。
`find(x)`	返回向量**x**中非零元素的下标。如果 **x**=**X**是一个矩阵，那么**X**的向量就作为一个长向量来考虑。
`[u,v]=find(A)`	返回矩阵**A**中非零元素的下标。
`[u,v,s]=find(A)`	返回矩阵**A**中非零元素的下标。用向量**s**中元素的值及**u**和**v**中相应的下标，实际上就是向量**u**、**v**和**s**作为命令sparse的参数。
`spconvert(D)`	将一个有三列的矩阵转换成一个稀疏矩阵。**D**中的第1列作为行的下标，第2列作为列的下标，最后一列作为元素值。

而且可以使用命令full将稀疏矩阵转换成一个满矩阵。

命令集88　转换成满矩阵

`full(S)`	将稀疏矩阵**S**转换成一个满矩阵。

■ **例**9.3

(a) 创建一个5×5的单位矩阵：

```
A=eye(5)
```

将矩阵**A**转换成稀疏矩阵**B**：

```
B = sparse(A)
```

```
B =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
```

(b) 假设MATLAB中给出如下的向量：

```
ind1 = [1 2 3 3 4 2];
ind2 = [1 2 1 4 5 3];
number = [0 1 2 3 0 5];
```

这样就有了行向量，但是也可使用列向量。运行命令`Smatrix=sparse(ind1,ind2,number)`，结果为：

```
Smatrix =
   (3,1)        2
```

```
   (2,2)        1
   (2,3)        5
   (3,4)        3
```

其中有去掉了两个零元素。将这个矩阵转换成满矩阵，输入：

```
Fullmatrix=full(Smatrix)
```

得到的结果为：

```
Fullmatrix =
     0     0     0     0     0
     0     1     5     0     0
     2     0     0     3     0
     0     0     0     0     0
```

注意，稀疏矩阵和得到的满矩阵的大小是分别是由 **ind1**和**ind2**中最大元素值确定的，即使相应的值是零，并且在列出的稀疏矩阵中去掉这个值。

输入命令whos可得到：

```
  Name             Size          Bytes      Class

  A                5x5             200      double array
  B                5x5              84      sparse array
  Fullmatrix       4x5             160      double array
  Smatrix          4x5              96      sparse array
  ind1             1x6              48      double array
  ind2             1x6              48      double array
  number           1x6              48      double array

Grand total is 74 elements using 684 bytes
```

可以看出虽然两个矩阵的大小相同，但是其中稀疏矩阵需要的存储空间更小些。

(c) 在处理稀疏矩阵时find命令很有用。命令对于稀疏矩阵或者满矩阵都返回相同的结果。返回得到的三个向量直接用来重新创建一个稀疏矩阵。令Smatrix定义在(b)中，运行命令：

```
[ind1,ind2,number] = find(Smatrix);
Smaller = sparse(ind1,ind2,number)
```

得到的结果为：

```
Smaller =
   (3,1)        2
   (2,2)        1
   (2,3)        5
   (3,4)        3
```

用下面命令得到的矩阵和(b)中得到的矩阵是不一样的：

```
Fullsmall = full(Smaller)

Fullsmall =
     0     0     0     0
     0     1     5     0
     2     0     0     3
```

■

9.3 稀疏矩阵运算

MATLAB中对满矩阵的运算和函数同样可用在稀疏矩阵中。结果是稀疏矩阵还是满矩阵，这取决于运算符或者函数及下列的操作数：

• 当函数用一个矩阵作为输入参数，输出参数为一个标量或者一个给定大小的向量时，输出参数的格式总是返回一个满阵形式，如命令 `size`。

• 当函数用一个标量或者一个向量作为输入参数，输出参数为一个矩阵时，输出参数的格式也总是返回一个满矩阵，如命令`eye`。还有一些特殊的命令可以得到稀疏矩阵，如命令`speye`。

• 对于单参数的其他函数来说，通常返回的结果和参数的形式是一样的，如 `diag`。

• 对于双参数的运算或者函数来说，如果两个参数的形式一样，那么也返回同样形式的结果。在两个参数形式不一样的情况下，除非运算的需要，均以满矩阵的形式给出结果。

• 两个矩阵的组和[A B]，如果**A**或**B**中至少有一个是满矩阵，则得到的结果就是满矩阵。

• 表达式右边的冒号是要求一个参数的运算符，遵守这些运算规则。

• 表达式左边的冒号不改变矩阵的形式。

■ **例**9.4

假设有：

```
A = eye(5); B = sparse(A); h = [1;2;0;4;5];
```

这是一个5×5的单位满矩阵和相应的稀疏矩阵。

(a) `C=5*B`，结果为：

```
C =
   (1,1)        5
   (2,2)        5
   (3,3)        5
   (4,4)        5
   (5,5)        5
```

这是一个稀疏矩阵。

(b) `D=A+B`，给出的结果为：

```
D =
     2     0     0     0     0
     0     2     0     0     0
     0     0     2     0     0
     0     0     0     2     0
     0     0     0     0     2
```

这是一个满矩阵。

(c) `x=B\h`，结果为：

```
x =
     1
     2
     0
     4
     5
```

■

这是一个满向量。

有许多命令可以对非零元素进行操作。

命令集89 矩阵的非零元素

nnz(A)	求矩阵**A**中非零元素的个数。它既可求满矩阵也可求稀疏矩阵。
spy(A)	画出稀疏矩阵**A**中非零元素的分布。也可用在满矩阵中，在这种情况下，只给出非零元素的分布。
spy(A,cstr,size)	用指定的颜色**cstr**(见表13-1)和在*size*规定的范围内画出稀疏矩阵**A**中非零元素的分布。
nonzeros(A)	按照列的顺序找出矩阵A中非零的元素。
spones(A)	把矩阵**A**中的非零元素全换为1。
spalloc(m,n,nzmax)	产生一个$m\times n$阶只有*nzmax*个非零元素的稀疏矩阵。这样可以有效地减少存储空间和提高运算速度。
nzmax(A)	给出为矩阵**A**中非零元素分配的内存数。不一定和nnz(A)得到的数相同；参见sparse或者spalloc。
issparse(A)	如果矩阵**A**是稀疏矩阵，则返回1；否则返回0。
spfun(fcn,A)	用**A**中所有非零元素对函数**fcn**求值，如果函数不是对稀疏矩阵定义的，同样也可以求值。
spfun(A)	求稀疏矩阵 **A**的结构秩。对于所有的矩阵来说，都有sprank(A≥rank(A)。

■ **例**9.5

用下面的命令定义稀疏矩阵：

```
A = sparse(diag(ones(5,1),1)) + sparse(diag(ones(5,1),-1));
```

现在创建一个大矩阵：

```
Big=kron(A, A)
```

这个矩阵**Big**是什么样子呢？Kronecker张量积给出一个大矩阵，它的元素是矩阵 A的元素之间可能的乘积。因为参量都是稀疏矩阵，所以得到的矩阵也是一个稀疏矩阵。可以用命令whos和issparse来确认一下。

查看矩阵**Big**的结构图，可输入spy(Big)，结构如图9-1所示。

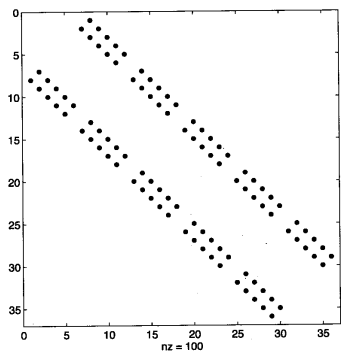

图9-1 用spy命令显示的矩阵结构图

可以看出Big是一个块双对角矩阵。

■

9.4 稀疏矩阵的特例

MATLAB中有四个基本稀疏矩阵，它们是单位矩阵、随机矩阵、对称随机矩阵和对角矩阵。

命令集90 单位稀疏矩阵

`speye(n)`	生成$n\times n$的单位稀疏矩阵。
`speye(m,n)`	生成$m\times n$的单位稀疏矩阵。

命令speye(A) 得到的结果和`sparse(eye(A))`是一样的，但是没有涉及到满阵的存储。

命令集91 随机稀疏矩阵

`sprand(A)`	生成与A有相同结构的随机稀疏矩阵，且元素服从均匀分布。
`sprand(m,n,dens)`	生成一个$m\times n$的服从均匀分布的随机稀疏矩阵，有$dens\times m\times n$个非零元素，$0\le dens\le 1$。参数*dens*是非零元素的分布密度。
`sprand(m,n,dens, rc)`	生成一个近似的条件数为1/*rc*、大小为$m\times n$的随机稀疏矩阵。如果*rc*=**rc**是一个长度为$l\le l(\min(m,n))$的向量，那么矩阵将rc_i作为它*l*个奇异值的第一个，其他的奇异值为0。
`sprandn(A)`	生成与A有相同结构的随机稀疏矩阵，且元素服从正态分布。
`sprandn(m,n,dens, rc)`	生成一个$m\times n$的服从正态分布的随机稀疏矩阵，和sprand一样。
`sprandsym(S)`	生成一个随机对称稀疏矩阵。它的下三角及主对角线部分与**S**的结构相同，矩阵元素服从正态分布。
`sprandsym(n,dens)`	生成一个$m\times n$的随机对称稀疏矩阵。矩阵元素服从正态分布，分布密度为*dens*。
`sprandsym(n,dens, rc)`	生成一个近似条件数为1/*rc*的随机对称稀疏矩阵。元素以0对称分布，但不是正态分布。如果*rc*=**rc**是一个向量，则矩阵有特征值rc_i。也就是说，如果**rc**是一个正向量，则矩阵是正定矩阵。
`sprandsym(n,dens, rc,k)`	生成一个正定矩阵。如果$k=1$，则矩阵是由一正定对称矩阵经随机Jacobi旋转得到的，其条件数正好等于 1/*rc*；如果$k=2$，则矩阵为外积的换位和，其条件数近似等于 1/*rc*。
`sprandsym(S,dens, rc,3)`	生成一个与矩阵**S**结构相同的稀疏矩阵，近似条件数为1/*rc*。参数*dens*被忽略，但是这个参数在这个位置以便函数能确认最后两个参数的正确与否。

■ **例9.6**

(a) 假设有矩阵A：

$$\mathbf{A}=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix}$$

输入Random=sprandn(A)，可得到随机稀疏矩阵：

```
Random =
   (2,1)       -0.4326
   (1,2)       -1.6656
   (3,2)        0.1253
   (4,3)        0.2877
```

矩阵中随机数的位置和矩阵**A**中非零元素的位置相同。

(b) 对于(a)中的矩阵**A**，输入：

```
B=sprandsym(A)
```

结果为：

```
B =
   (2,1)       -1.1465
   (1,2)       -1.1465
   (3,2)        1.1909
   (2,3)        1.1909
   (4,3)        1.1892
   (3,4)        1.1892
```

这是一个用矩阵**A**的下三角及主对角线部分创建的对称矩阵，在非零元素的位置用随机数作为元素值。

■

用命令spdiags可以取出对角线元素，并创建带状对角矩阵。假设矩阵 **A**的大小为$m\times n$，在p个对角线上有非零元素。 **B**的大小为$\min(m\times n)\times p$，它的列是矩阵**A**的对角线。向量**d**的长度为p，其整型分量给定了A的对角元：

$d_i<0$ 主对角线下的对角线。举例如$d_i=-1$，用第一个下对角线

$d_i=0$ 用主对角线

$d_i>0$ 主对角线上的对角线

命令集92 对角稀疏矩阵

[B,d]=spdiags(A)	求出**A**中所有的对角元，对角元保存在矩阵 **B**中，它们的下标保存在向量**d**中。
spdiags(A,d)	生成一个矩阵，这个矩阵包含有矩阵**A**中向量**d**规定的对角元。
spdiags(B,d,A)	生成矩阵**A**，用矩阵**B**中的列替换**d**定义的对角元。
A=spdiags(B,d,m,n)	用保存在由**d**定义的**B**中的对角元创建稀疏矩阵**A**。

例11.4给出了如何使用spdiags命令来解普通微分方程组。

9.5 系数阵为稀疏矩阵的线性方程组

在许多实际应用中要保留稀疏矩阵的结构，但是在计算过程中的中间结果会减弱它的稀疏性，如LU分解。这就会导致增加浮点运算次数和存储空间。为了避免这种情况发生，在

MATLAB中用命令对矩阵进行重新安排。这些命令都列在下面的命令集 93中。通过help命令可以得到每个命令更多的帮助信息，也可见 helpdesk。

命令集93 矩阵变换

colmmd(A)	返回一个变换向量，使得矩阵 **A**列的秩为最小。
symmmd(A)	返回使对称矩阵秩为最小的变换。
symrcm(A)	矩阵**A**的Cuthill-McKee逆变换。矩阵**A**的非零元素在主对角线附近。
colperm(A)	返回一个矩阵**A**的列变换的向量。列按非零元素升序排列。有时这是LU因式分解前有用的变换：lu(A(:, j))。如果**A**是一个对称矩阵，对行和列进行排序，这有利于Cholesky分解：chol(A(j, j)) 。
randperm(n)	给出正数1，2，…，*n*的随机排列，可以用来创建随机变换矩阵。
dmperm(A)	对矩阵A进行Dulmage-Mendelsohn分解，输入help dmperm可得更多信息。

■ **例**9.7

创建一个秩为4的变换矩阵，可输入：

```
i = [1 2 3 4]; aa = ones(1,4); perm = randperm(4)
P = sparse(i,perm,aa)
```

一旦运行perm=randperm(4)，就会得到：

```
perm =
     4     1     3     2
```

给出的变换矩阵为：

```
P =
   (2,1)        1
   (4,2)        1
   (3,3)        1
   (1,4)        1
```

如果矩阵A为：

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

输入命令：

```
RowChange = P*A, ColChange = A*P
```

运行结果为：

```
RowChange =
     5     6     3     1
     5     2     4     2
     8     1     1     2
     7     4     3     4
```

```
ColChange =
     4     4     7     3
     2     2     5     4
     1     6     5     3
     2     1     8     1
```

■

有两个不完全因式分解命令，它们是用来在解大线性方程组前进行预处理的。用helpdesk命令可得更多信息。

命令集94 不完全因式分解

cholinc(A,opt)	进行不完全Cholesky分解，变量opt取下列值之一：
	droptol 指定不完全分解的舍入误差，0给出完全分解。
	michol 如果michol=1，则从对角线上抽取出被去掉的元素。
	rdiag 用sqrt(droptol*norm(X(:,j)))代替上三角分解因子中的零元素，*j*为零元素所在的列。
[L,U,P]= luinc(X,opt)	返回矩阵**X**的不完全分解得到的三个矩阵**L**、**U**和**P**，变量**opt**取下列值之一：
	droptol 指定分解的舍入误差。
	milu 改变分解以便从上三角角分解因子中抽取被去掉的列元素。
	udiag 用droptol值代替上三角角分解因子中的对角线上的零元素。
	thresh 中心极限。

解稀疏线性方程组既可用左除运算符解，也可用一些特殊命令来解。

命令集95 稀疏矩阵和线性方程组

spparms(keystr,op)	设置稀疏矩阵算法的参数，用help spparms可得详细信息。
spaugment(A,c)	根据[c*l A; A'0] 创建稀疏矩阵，这是二次线性方程组的最小二乘问题。参见7.7节。
symbfact(A)	给出稀疏矩阵的Cholesky和LU因式分解的符号分解因子。用help symbfact可得详细信息。

稀疏矩阵的范数计算和普通满矩阵的范数计算有一个重要的区别。稀疏矩阵的欧几里德范数不能直接求得。如果稀疏矩阵是一个小矩阵，则用 norm(full(A))来计算它的范数；但是对于大矩阵来说，这样计算是不可能的。然而 MATLAB可以计算出欧几里德范数的近似值，在计算条件数时也是一样。

命令集96 稀疏矩阵的近似欧几里德范数和条件数

normest(A)	计算A的近似欧几里德范数，相对误差为10^{-6}。
normest(A,tol)	计算A的近似欧几里德范数，设置相对误差 *tol*，而不用缺省时的10^{-6}。

`[nrm,nit]= normest(A)`	计算近似*nrm*范数，还给出计算范数迭代的次数*nit*。
`condest(A)`	求矩阵**A**条件数的1-范数中的下界估计值。
`[c,v]= condest(A,tr)`	求矩阵**A**的1-范数中条件数的下界估计值c和向量**v**，使得 ‖**Av**‖=(‖**A**‖ · ‖**v**‖)/c。如果给定*tr*，则给出计算的过程。*tr*=1，给出每步过程；*tr*=－1，给出商`c/rcond(A)`。

■ **例**9.8

假设给出：

```
Sprs = speye(4); Sprs(4,1) = 19; Sprs(3,2) = 4;
```

用normApprox=normest(Sprs)计算出：

```
normApprox =
   19.0525
```

用theNorm=norm(full(Sprs))得：

```
theNorm =
   19.0525
```

为了找到它们之间的差别，计算`difference=theNorm-normApprox`，得：

```
difference =
   8.5577e-09
```

在许多应用中，`normest`计算得到的近似值是一个很好的近似欧几里德范数，它的计算步数要比`norm`要少得多；可参见7.6节。

■

用`etree`命令来找到稀疏对称矩阵的消元树，用向量 **f** 来描述消元树，还可用`etreeplot`命令画出来。元素f_i是矩阵的上三角Cholesky分解因子中*i*行上第1非零元素的列下标。如果有非零元素，则f_i=0。消元树可以这样来建立：

节点*i*是f_i的孩子，或者如果f_i=0，则节点*i*是树的根节点。

命令集97　矩阵的消元树

`etree(A)`	求**A**的消元树向量 **f**，这个命令有可选参数；输入 `help etree`获取帮助。
`etreeplot(A)`	画出向量**f**定义的消元树图形。
`treeplot(p,c,d)`	画出指针向量**p**的树图形，参数*c*和*d*分别指定节点的颜色和分支数。`etreeplot`可以调用这个命令。
`treelayout`	显示树的结构，`treeplot`可以调用这个命令。

■ **例**9.9

假设有对称稀疏矩阵**B**：

$$\mathbf{B}=\begin{pmatrix}5&1&0&0&1&0&0&1\\1&5&0&0&0&0&1&0\\0&0&5&0&1&1&1&0\\0&0&0&5&0&0&1&1\\1&0&1&0&5&0&0&0\\0&0&1&0&0&5&1&0\\0&1&1&1&0&1&5&0\\1&0&0&1&0&0&0&5\end{pmatrix}$$

运行命令btree=etree(B)，结果为：

```
btree =
     2     5     5     7     6     7     8     0
```

开始的数字2不难理解，它是矩阵的第1列上第1个非零元素的行数，它决定了在Cholesky分解因子的第1行第2列处有一个非零元素。当缩减第1列的元素时就得到第2列的数字5。**B**在缩减后，在(5,2)位置的元素是非零的，这样消元树向量中第2个元素的值为5。spy(chol(B))给出了Cholesky分解因子的结构图，如图9-2所示：

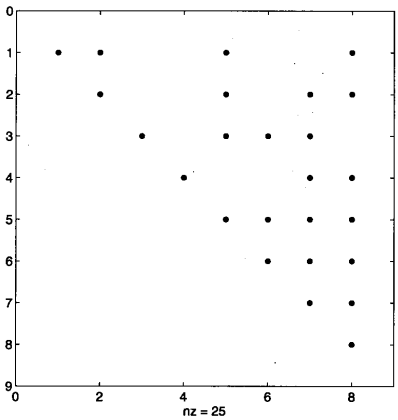

图9-2 Cholesky分解结构图

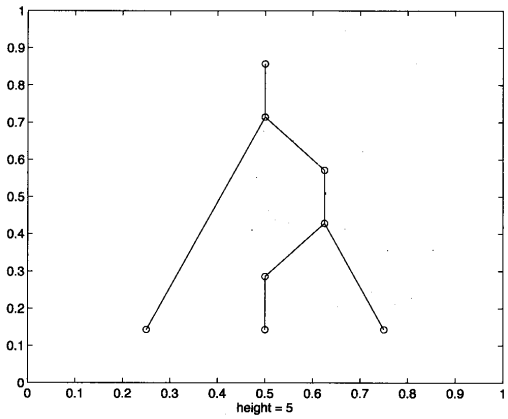

图9-3 矩阵B的消元树

这个向量消元树可以这样来建立：上三角中只有一行有非零元素，节点 8，因此这就是树唯一的根。节点1是节点2的孩子，节点2和3是节点5的孩子，而节点5是节点6的孩子。节点4

和6是节点7的孩子，而节点7又是节点8的孩子，即根的孩子。

命令etreeplot(B)给出了树的结构图，如图9-3所示。

消元树的形状取决于列和行序，它可以用来分析消元过程。

■

用gplot命令可以画出坐标和矩阵元素间的联系图形。必须在$n\times2$的矩阵中给出n个坐标，矩阵的每一行作为一个点。这样就创建出点点之间连接的 $n\times n$矩阵，如果点4连接到点8，则(4, 8)的值为1。由于是一个大矩阵，而且非零元素较少，所以它应该被建成稀疏矩阵。

这个图可以说明网络问题，如传递问题。它还包含有线性方程组中未知量之间的相关信息。

命令集98 网络图形

gplot(A,K)	如果矩阵**A**的$a(i, j)$不为0，则将点k连接到点kj。**K**是一个$n\times2$的坐标矩阵，**A**是一个$n\times n$的关联矩阵。
gplot(A,K,str)	用字符串str给定的颜色和线型画出的同上图形。字符串 **str**的取值参见表13-1。
[X,A]=unmesh(E)	求网格边界矩阵**E**的Laplace矩阵**A**和网格点的坐标矩阵**X**。

■ **例**9.10

假设有下面的坐标矩阵**K**和关联矩阵**A**：

$$\mathbf{K}=\begin{pmatrix}0 & 1\\ 1 & 0.2\\ 1.3 & 0.9\\ 2 & 0\\ 2 & 1.9\\ 3 & 2\\ 4 & 1\end{pmatrix}\qquad \mathbf{A}=\begin{pmatrix}0&1&1&0&1&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&1&0&0&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&1&1\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0\end{pmatrix}$$

矩阵A在稀疏化后，用命令gplot(A,K)画出图9-4，给出了点(0, 1)和点(4, 1)之间所有可能的路径。

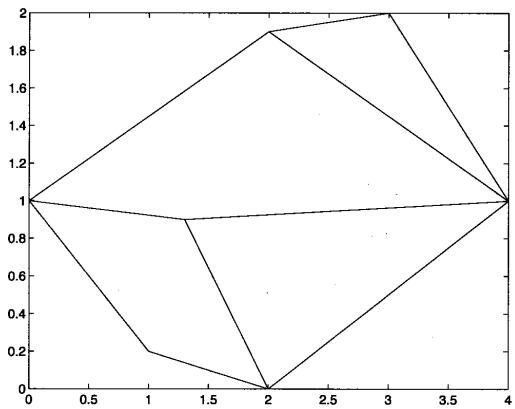

图9-4 使用gplot的一个例子

■