Matrice rare şi bandă

R. Trîmbiţaş
UBB

24 martie 2021

1 Matrice rare şi bandă

Matricele rare şi bandă apar frecvent în calcule ştiinţifice şi tehnice. Raritatea (sparsity) unei matrice este proporţia de elemente nule. Funcţia MATLAB nnz numără elementele nenule dintr-o matrice, deci raritatea lui A este dată de

```
density = nnz(A)/prod(size(A))
sparsity = 1 - density
```

O *matrice rară* este o matrice a cărei raritate este apropiată de 1. *Lăţimea de bandă* a unei matrice este distanţa maximă a elementelor nenule de diagonala principală.

```
[i,j] = find(A)
bandwidth = max(abs(i-j))
```

O matrice bandă este o matrice a cărei lăţime de bandă este mică.

Aşa cum se poate vedea, raritatea şi lăţimea de bandă sunt noţiuni subiective. O matrice diagonală $n \times n$ fără nici un zero pe diagonala principală are raritatea $1 - 1/n$ şi lăţimea benzii 0, deci este un exemplu extrem de matrice şi bandă şi rară. Pe de altă parte o matrice $n \times n$ fără elemente nenule, cum ar fi una creată cu `rand(n,n)`, are raritatea egală cu zero şi lăţimea benzii egală cu $n - 1$, deci departe de a se califica în oricare dintre categorii.

Structura MATLAB pentru matrice rare (sparse) memorează elementele nenule împreună cu informaţii despre indicii lor. Această structură gestionează eficient şi matricele bandă şi din acest motiv MATLAB nu are o clasă separată pentru matrice bandă. Instrucţiunea

```
S = sparse(A)
```

converteşte o matrice într-una rară. Instrucţiunea

```
A = full(S)
```

realizează operaţia inversă. Totuşi, cele mai multe matrice rare au ordine atât de mari încât este nepractic să fie memorate ca matrice dense. Mai frecvent, matricele rare sunt create prin

```
S = sparse(i,j,x,m,n)
```

Aceasta produce o matrice `S` cu

```
[i,j,x] = find(S)
[m,n] = size(S)
```

Cele mai multe operaţii şi funcţii matriciale din MATLAB sunt aplicabile atât matricelor rare cât şi celor dense. Factorul dominat în determinarea timpului de execuţie şi a necesarului de memorie este numărul de elemente nenule, `nnz(S)`, pe care le au diversele matrice.

O matrice cu lăţimea benzii egală cu 1 se numeşte matrice tridiagonală. Merită să implementăm o funcţie specializată pentru rezolvarea unui sistem liniar cu o astfel de matrice:

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_1 & b_1 & c_2 & & & \\ & a_2 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

Algoritmul corespunzător se numeşte algoritmul lui Thomas. Funcţia

```
x = Thomas(a,b,c,d)
```

rezolvă sistemul tridiagonal cu subdiagonala `a`, diagonala `b`, superdiagonala `c` şi membrul drept `d`. Principiul este ca la eliminarea gaussiană. În multe situaţii practice ce presupun matrice tridiagonale, elementele diagonale domină elementele nediagonale, deci pivotarea nu este necesară. Membrul drept este prelucrat în acelaşi timp cu matricea.

```
function x =Thomas(a,b,c,d)
x = d;
n = length(x);
for j = 1:n-1
   mu = a(j)/b(j);
   b(j+1) = b(j+1) - mu*c(j);
   x(j+1) = x(j+1) - mu*x(j);
end
x(n) = x(n)/b(n);
for j = n-1:-1:1
   x(j) = (x(j)-c(j)*x(j+1))/b(j);
end
```

Deoarece algoritmul nu utilizează pivotarea, rezultatul ar putea fi imprecis dacă `abs(b)` este mult mai mic decât `abs(a)+abs(c)`. Alternative mai robuste, dar lente cu pivotare ar putea fi generarea unei matrice dense cu `diag`

```
T = diag(a,-1) + diag(b,0) + diag(c,1)
x = T\d
```

sau generarea unei matrice tridiagonale cu `spdiags`

```
S = spdiags([[a; 0] b [0; c]],[-1 0 1],n,n)}
x = S\d
```