

Matrice rare și bandă

R. Trîmbițaș
UBB

March 14, 2016

1 Matrice rare și bandă

Matricele rare și bandă apar frecvent în calcule științifice și tehnice. Raritatea (sparsity) unei matrice este proporția de elemente nule. Funcția 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. *Lățimea de bandă* a unei matrice este distanța maximă a elementelor nenule de diagonală principală.

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

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

Așa cum se poate vedea, raritatea și lățimea de bandă sunt noțiuni subiective. O matrice diagonală $n \times n$ fără nici un zero pe diagonală principală are raritatea $1 - 1/n$ și lățimea benzii 0, deci este un exemplu extrem de matrice și bandă și 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 și lățimea 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 informații despre indicii lor. Această structură gestionează eficient și matricele bandă și din acest motiv MATLAB nu are o clasă separată pentru matrice bandă. Instrucțiunea

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

convertește o matrice într-una rară. Instrucțiunea

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

realizează operația inversă. Totuși, 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 operații și funcții matriciale din MATLAB sunt aplicabile atât matricelor rare cât și celor dense. Factorul dominant în determinarea timpului de execuție și a necesarului de memorie este numărul de elemente nenule, `nnz(S)`, pe care le au diversele matrice.

O matrice cu lățime benzii egală cu 1 se numește matrice tridiagonală. Merită să implementăm o funcție 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 numește algoritmul lui Thomas. Funcția

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

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