Scientific Computing

A practical Companion

7th Notebook

Author:

Walter Hoffmann (Korteweg-de Vries Institute for Mathematics, UvA)

January - March, 2008

Systems of linear equations III
Iterative solution methods by

fixed-point iteration, part 2

Better parallelism with special ordering:

"Red Black" ordering

Consider the discretization of the diffusion operator using the well known 'five-point' stencil. Now apply a sort of 'leap frog' ordering to obtain the situation that points are devided in two classes: 'Reds and Blacks' such that in each five-point stencil a central-point is surrounded by points from the other class.

In the periodic case, this does not always work; for instance in the choice of grid (a 5 × 4 grid) we have used before, it doesn't work:

```
18     9    19    10    20   (18)
 6    16     7    17     8   ( 6)
13     4    14     5    15   (13)
 1    11     2    12     3   ( 1)
```

If on this grid the number of different points in x-direction would have been even, then it would work, as we can see for instance in the 4× 3 situation (presuming that we stick to a grid that in each direction contains from boundary to boundary the same number of points) :

```
 5    11     6    12   ( 5)
 9     3    10     4   ( 9)
 1     7     2     8   ( 1)
```

When we now consider the matrix for this problem we observe the following structure:

$$\begin{array}{r|rrrrrrrrrrrr}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
1 & -4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & -4 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
4 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
5 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 1 & 1 \\
7 & 1 & 1 & 1 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\
8 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\
9 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & -4 & 0 & 0 \\
11 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -4 & 0 \\
12 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -4
\end{array}$$

Symbolically we can write a linear system for this matrix in partitioned form as:

$$\begin{bmatrix} D_R & U \\ L & D_B \end{bmatrix} \begin{bmatrix} x_R \\ x_B \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Both the right-hand side vector and the vector of unknowns have been splitted in two parts. For these parts the equations read:

$$\begin{aligned} D_R\, x_R + U\, x_B &= b_1\,, \\ L\, x_R + D_B\, x_B &= b_2\,. \end{aligned}$$

If we look closely at the first of these two systems, we observe that the unknown part x_R has a diagonal matrix as coefficientmatrix and therefore it can easily be expressed in terms of x_B:

$$x_R = {D_R}^{-1}\, b_1 - {D_R}^{-1}\, U\, x_B\,.$$

Substitution in the second system yields

$$L\left({D_R}^{-1}\, b_1 - {D_R}^{-1}\, U\, x_B\right) + D_B\, x_B = b_2.$$

which defines the solution of the system of linear equations

$$\left(D_B - L\, {D_R}^{-1}\, U\right) x_B = \left(b_2 - L\, {D_R}^{-1}\, b_1\right).$$

Defining the coefficientmatrix G and right-hand side vector h , we call this the linear system

$$G\, x_B = h.$$

Going back to the expression we had before and after rearrangement and putting iteration indices to it, we arrive at :

$$\begin{aligned} D_B\, {x_B}^{(i+1)} &= b_2 - L\,(\,{D_R}^{-1}\, b_1 - {D_R}^{-1}\, U\, {x_B}^{(i)}\,) \\ &= h + L\, {D_R}^{-1}\, U\, {x_B}^{(i)} \end{aligned}$$

which can be recognized as Jacobi iteration in its fixed-point formulation for calculating the solution of .

$$G\, x_B = h.$$

We should emphasize that the diagonal of G is not necessarily equal to D_B ; nevertheless it can be used as a splitting matrix for an iteration in defect-correction form as:

$$D_B\left({x_B}^{(i+1)} - {x_B}^{(i)}\right) = h - G\, {x_B}^{(i)}$$

where the right-hand side is clearly the residual vector for the above system of linear equations, calculated for the approximate solution ${x_B}^{(i)}$. This formulation fits our earlier description of Jacobi iteration.

Convergence of this iteration is determined by the size of $\| I - {D_B}^{-1}\, G \|$.

After having calculated x_B, vector x_R follows from

$$D_R\, x_R = b_1 - U\, x_B\,.$$

▽ Example of implementation

The following example shows an implementation of the above treatment for the same problem we treated before. Because of the periodicity in the considered diffusion problem, we saw that an odd number of gridpoints must be taken in x-direction (identification of first and last point on a line that is parallel to the X-axis as consequence of the periodicity).

In[697]:=

```
AR = {{4, 0, 0, 0, 0, 0, −1, −1, −1, 0, 0, 0},
    {0, 4, 0, 0, 0, 0, −1, −1, 0, −1, 0, 0},
    {0, 0, 4, 0, 0, 0, −1, 0, −1, −1, −1, 0},
    {0, 0, 0, 4, 0, 0, 0, −1, −1, −1, 0, −1},
    {0, 0, 0, 0, 4, 0, 0, 0, −1, 0, −1, −1},
    {0, 0, 0, 0, 0, 4, 0, 0, 0, −1, −1, −1},
    {−1, −1, −1, 0, 0, 0, 4, 0, 0, 0, 0, 0},
    {−1, −1, 0, −1, 0, 0, 0, 4, 0, 0, 0, 0},
    {−1, 0, −1, −1, −1, 0, 0, 0, 4, 0, 0, 0},
    {0, −1, −1, −1, 0, −1, 0, 0, 0, 4, 0, 0},
    {0, 0, −1, 0, −1, −1, 0, 0, 0, 0, 4, 0},
    {0, 0, 0, −1, −1, −1, 0, 0, 0, 0, 0, 4}};
AR // MatrixForm
```

Out[698]//MatrixForm=

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & -1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & -1 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix}$$

In[699]:=

```
D1 = {{4, 0, 0, 0, 0, 0}, {0, 4, 0, 0, 0, 0}, {0, 0, 4, 0, 0, 0},
     {0, 0, 0, 4, 0, 0}, {0, 0, 0, 0, 4, 0}, {0, 0, 0, 0, 0, 4}};
```

In[700]:=
```
D2 = D1;
```

In[701]:=
```
D2 // MatrixForm
```

Out[701]//MatrixForm=
$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix}$$

In[702]:=
```
UR = {{−1, −1, −1, 0, 0, 0}, {−1, −1, 0, −1, 0, 0},
       {−1, 0, −1, −1, −1, 0}, {0, −1, −1, −1, 0, −1},
       {0, 0, −1, 0, −1, −1}, {0, 0, 0, −1, −1, −1}};
```

In[703]:=
```
LR = UR;
```

In[704]:=
```
LDU = LR.Inverse[D1].UR;
```

The amplificationmatrix for the residualvector in this iteration is given by

$$\begin{aligned} &(I - G D_B{}^{-1}) \\ &= (I - (D_B - L D_R{}^{-1} U) D_B{}^{-1}) \\ &= L D_R{}^{-1} U \; D_B{}^{-1} \end{aligned}$$

We calculate its norm

In[705]:=
```
q = Norm[LR.Inverse[D1].UR.Inverse[D2]]
```

Out[705]=
$$\frac{1}{8} \sqrt{17 + 12 \sqrt{2}}$$

```
In[706]:=
  N[q]
Out[706]=
  0.728553
In[707]:=
  Log[0.1]/Log[q]
Out[707]=
  7.27069
In[708]:=
  G = D2 – LDU;
  N[G] // MatrixForm
Out[709]//MatrixForm=
```

$$\begin{pmatrix} 3.25 & -0.5 & -0.5 & -0.5 & -0.25 & 0. \\ -0.5 & 3.25 & -0.5 & -0.5 & 0. & -0.25 \\ -0.5 & -0.5 & 3. & -0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & -0.5 & 3. & -0.5 & -0.5 \\ -0.25 & 0. & -0.5 & -0.5 & 3.25 & -0.5 \\ 0. & -0.25 & -0.5 & -0.5 & -0.5 & 3.25 \end{pmatrix}$$

```
In[710]:=
  b1 = {0, 0, 0, 0, 1, 1};
  b2 = b1;
In[712]:=
  h = b2 – LR.Inverse[D1].b1
Out[712]=
```

$$\left\{0, 0, \frac{1}{4}, \frac{1}{4}, \frac{3}{2}, \frac{3}{2}\right\}$$

```
In[713]:=
  z = LinearSolve[G, h]
Out[713]=
```

$$\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}\right\}$$

The y-axis has been divided into 4 intervals; the solution of the diffusion equation shows a linear behaviour from y = 0 until y = 1. Of the six points that we find with this solution, two are on the line at distance 1/4, two on the line at 1/2 and two on the line at distance 3/4.

Now for the (standard) Jacobi iteration

```
In[714]:=
x = {0., 0., 0., 0., 0., 0.};
i = 0;
res2 = 1;
```

```
In[717]:=
While[res2 > 0.01 && i < 50,
     r = h - G.x;
     res2 = r.r;
     i = i + 1;
     d = LinearSolve[D1, r];
     x = x + d;
   ];
res2
i
x
```

```
Out[718]=
0.00574493
Out[719]=
10
Out[720]=
{0.232019, 0.232019, 0.474571,
 0.474571, 0.732019, 0.732019}
```

And for the approximation of x_B we find with the application of

$$D_R\, x_R = b_1 - U\, x_B\,.$$

```
In[721]:=
y = Inverse[D1] .(b1 - UR.x)
```

```
Out[721]=
{0.234652, 0.234652, 0.478295,
 0.478295, 0.734652, 0.734652}
```

▽ Use of modified Jacobi

▽ Turbo Jacobi

In[722]:=

```
x = {0., 0., 0., 0., 0., 0.};
r = 1. * h;
i = 0;
res2 = 1;
```

In[726]:=

```
While[res2 > 0.01 && i < 50,
     res2 = r.r;
     d = LinearSolve[D1, r];
     c = G.d;
     rho = c.r / c.c;
     r = r - rho c;
     x = x + rho d;
     i = i + 1;
  ];
res2
i
x
```

Out[727]=

```
0.00516007
```

Out[728]=

```
6
```

Out[729]=

```
{0.237948, 0.237948, 0.489012,
 0.489012, 0.737948, 0.737948}
```

```
In[730]:=
y = Inverse[D1] .(b1 - UR.x)
Out[730]=
{0.241227, 0.241227, 0.48848,
 0.48848, 0.741227, 0.741227}
```

Close all sections