4/ Jacobi

Grzegorz Studzinski / 306504 / IS / gr. 7

Date: 10.04.2020

Task source: http://home.agh.edu.pl/~byrska/src/MN_2020/5_Jacobi.pdf

Additional source: https://www3.nd.edu/~zxu2/acms40390F12/Lec-7.3.pdf

1. Jacobi method overview

Two assumptions made on Jacobi Method:

1. The system given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n &= b_n \end{aligned}$$

has a unique solution.

2. The coefficient matrix A has no zeros on its main diagonal, namely, a11, a22, ann, are nonzeros

Main idea of Jacobi:

To begin, **solve**:

the 1st equation for x1

The 2nd equation for x2

...and so on

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots a_{1n}x_n)$$
$$x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots a_{2n}x_n)$$
$$\vdots$$
$$x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots a_{n,n-1}x_{n-1})$$

To obtain the rewritten equations.

Then, **make a initial guess of the solution**

$$x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \ldots x_n^{(0)}).$$

Susbitute these values(inital guess) into the right hand side of the rewritten equations to obtain *the first approximation.*

$$\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots x_n^{(1)}\right).$$

This accomplishes one iteration.

The second approximation is computed by **substituting the first approximation's x-values into the right hand side of the reqritten equations.**

$$\left(x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \ldots x_n^{(2)}\right).$$

Then, continue with next iterations...

The sequence of approximations:

$$x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \ldots x_n^{(k)}\right)^t, \quad k = 1,2,3, \ldots$$

Real example:

$$\begin{aligned} 5x_1 - 2x_2 + 3x_n &= -1 \\ -3x_1 + 9x_2 + x_n &= 2 \\ 2x_1 - x_2 - 7x_n &= 3 \end{aligned}$$

Rewritten equations:

$$x_1 = \frac{-1}{5} + \frac{2}{5}x_2 - \frac{3}{5}x_3$$
$$x_2 = \frac{2}{9} + \frac{3}{9}x_1 - \frac{1}{9}x_3$$
$$x_3 = -\frac{3}{7} + \frac{2}{7}x_1 - \frac{1}{7}x_2$$

Make a inital guess:

the initial guess $x_1 = 0, x_2 = 0, x_3 = 0$

The first approximatin:

$$x_1^{(1)} = \frac{-1}{5} + \frac{2}{5}(0) - \frac{3}{5}(0) = -0.200$$
$$x_2^{(1)} = \frac{2}{9} + \frac{3}{9}(0) - \frac{1}{9}(0) = 0.222$$
$$x_3^{(1)} = -\frac{3}{7} + \frac{2}{7}(0) - \frac{1}{7}(0) = -0.429$$

Contiune iteration for k=2,3...

n	$k = 0$	$k = 1$	$k = 2$	$k = 3$
$x_1^{(k)}$	0.000	-0.200	0.146	0.192
$x_2^{(k)}$	0.000	0.222	0.203	0.328
$x_2^{(k)}$	0.000	-0.429	-0.517	-0.416

To sum up - unambiguous method notation:

For each $k \geq 1$, generate the components $x_i^{(k)}$ of $\boldsymbol{x}^{(k)}$ from $\boldsymbol{x}^{(k-1)}$ by

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{\substack{j=1, \\ j \neq i}}^{n} \left(-a_{ij} x_j^{(k-1)}\right) + b_i \right], \qquad \text{for } i = 1,2,, \dots n$$

Matrix Form:

nxn size matrix:

$$A\boldsymbol{x} = \boldsymbol{b} \text{ with } A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \text{ and } \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \text{ for } \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

We split A into

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 & 0 \\ -a_{21} & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & & \vdots \\ \vdots & \vdots & \ddots & -a_{n-1,n} \\ 0 & 0 & \cdots & 0 \end{bmatrix} = D - L - U$$

$A\boldsymbol{x} = \boldsymbol{b}$ is transformed into $(D - L - U)\boldsymbol{x} = \boldsymbol{b}$

$$D\boldsymbol{x} = (L + U)\boldsymbol{x} + \boldsymbol{b}$$

Assume D^{-1} exists and $D^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix}$

Then

$$\boldsymbol{x} = D^{-1}(L + U)\boldsymbol{x} + D^{-1}\boldsymbol{b}$$

The matrix form of Jacobi iterative method is

$$\boldsymbol{x}^{(k)} = D^{-1}(L + U)\boldsymbol{x}^{(k-1)} + D^{-1}\boldsymbol{b} \qquad k = 1,2,3, \dots$$

Define $T = D^{-1}(L + U)$ and $\boldsymbol{c} = D^{-1}\boldsymbol{b}$, Jacobi iteration method can also be written as

$$\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \boldsymbol{c} \qquad k = 1,2,3, \dots$$

2. Source code

2.1 Collecting user input

jacobi.txt is to be filled with data in the following order:

1st line: number of equations

Next lines: a[0][0] a[0][1] ……. b[0]   and so on

The last line: number of iterations

So, we read the matrix from file:

$$Ax = b \text{ with } A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \text{ and } b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \text{ for } x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Firstly we have to check if the matrix is diagonally dominant.

```
double diagonal;
double others;
int test1=0;
int test2=0;

for (int i = 0; i < n; i++) {
    others = 0;
    for (int j = 0; j < n; j++) {

        if (i == j) diagonal = a[i][j];
        else others += a[i][j];

    }
    if (abs(diagonal) >= abs(others)) test1++;
    if (diagonal > others) test2++;

}
if (test1 == n && test2 > 0) {
    cout << "Matrix is diagonally dominant." << endl;
}
else {
    cout << "Matrix is not diagonally dominant. Use another matrix." << endl;
    return;
}
```

2. Split A into D,L,U

We split A into

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 & 0 \\ -a_{21} & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & & \vdots \\ \vdots & \vdots & \ddots & -a_{n-1,n} \\ 0 & 0 & \cdots & 0 \end{bmatrix} = D - L - U$$

```cpp
    d = new double*[n];
    d_inv = new double*[n];
    l = new double*[n];
    u = new double*[n];
    m = new double*[n];


    for (int i = 0; i < n; ++i) {
    d[i] = new double[n];
    d_inv[i] = new double[n];
    l[i] = new double[n];
    u[i] = new double[n];
    m[i] = new double[n];
    }

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        d[i][j] = 0.0;
        d_inv[i][j] = 0.0;
        l[i][j] = 0.0;
        u[i][j] = 0.0;
    }
}
```

```cpp
    //D and D_inv
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        d[i][i] = a[i][i];
        d_inv[i][i] = 1 / a[i][i];
    }
}

//L
for (int i = 1; i < n; i++) {
    for (int j = 0; j < i; j++) {
        if (a[i][j] != 0) l[i][j] = a[i][j];
    }
}


//U
int temp = 0;
for (int i = 0; i < n  ; i++) {
    for (int j = n-1; j >temp ; j--) {
        if (a[i][j] != 0.0) u[i][j] = a[i][j];
    }
    temp++;
}
```

Displaying matrices:

```
cout << "The matrix D\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {

        if (j > n - 1) cout << "| ";
        cout << d[i][j] << "   ";
    }
    cout << "\n";
}
cout << "The matrix L\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {

        if (j > n - 1) cout << "| ";
        cout << l[i][j] << "   ";
    }
    cout << "\n";
}
cout << "The matrix U\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {

        if (j > n - 1) cout << "| ";
        cout << u[i][j] << "   ";
    }
    cout << "\n";
}

//Matrix D^(-1)
cout << "The matrix D^(-1)\n";
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
```

Function to calculate x[i]

```
for (int i = 1; i < ilimit+1; i++) {


    x[i] = sumVECTOR(multiplySQplusVECTOR(multiplySQ(multiplySQ(d_inv, m,n), sumSQ(l, u, n), n), x[i-1], n), multiplySQplusVECTOR(d_inv, b, n), n);

}
```

The matrices operations are handled by MatrixOperations.h

Displaying the results:

```
for (int i = 0; i < ilimit; i++) {
    cout << "Iteration no " << i + 1 << endl << "Solution: ";
    for (int j = 0; j < n; j++) {
        cout << x[i][j] << " ";
    }
    cout << endl << endl;
}
```

Output for the file from website:

```
The matrix from file
6   1   1   1   1   | 10
1   12   1   0   1   | 15
0   1   32   1   0   | 34
1   1   0   4   0   | 6
1   0   1   0   3   | 5
Matrix is diagonally dominant.
The matrix D
6   0   0   0   0
0   12   0   0   0
0   0   32   0   0
0   0   0   4   0
0   0   0   0   3
The matrix L
0   0   0   0   0
1   0   0   0   0
0   1   0   0   0
1   1   0   0   0
1   0   1   0   0
The matrix U
0   1   1   1   1
0   0   1   0   1
0   0   0   1   0
0   0   0   0   0
0   0   0   0   0
The matrix D^(-1)
0.166667   0   0   0   0
0   0.0833333   0   0   0
0   0   0.03125   0   0
0   0   0   0.25   0
0   0   0   0   0.333333
```

```
Iteration no 1
Solution: 0 0 0 0 0

Iteration no 2
Solution: 1.66667 1.25 1.0625 1.5 1.66667

Iteration no 3
Solution: 0.753472 0.883681 0.976563 0.770833 0.756944

Iteration no 4
Solution: 1.102 1.04275 1.0108 1.09071 1.08999

Iteration no 5
Solution: 0.960959 0.983102 0.995829 0.963813 0.962402

Iteration no 6
Solution: 1.01581 1.00673 1.00166 1.01398 1.0144

Iteration no 7
Solution: 0.99387 0.997344 0.999353 0.994364 0.994177

Iteration no 8
Solution: 1.00246 1.00105 1.00026 1.0022 1.00226

Iteration no 9
Solution: 0.999039 0.999585 0.999899 0.999122 0.999094

Iteration no 10
Solution: 1.00038 1.00016 1.00004 1.00034 1.00035

Iteration no 11
Solution: 0.99985 0.999935 0.999984 0.999863 0.999859

Iteration no 12
Solution: 1.00006 1.00003 1.00001 1.00005 1.00006

Iteration no 13
Solution: 0.999976 0.99999 0.999998 0.999979 0.999978

Iteration no 14
Solution: 1.00001 1 1 1.00001 1.00001

Iteration no 15
Solution: 0.999996 0.999998 1 0.999997 0.999997
```

```
Iteration no 16
Solution: 1 1 1 1 1

Iteration no 17
Solution: 0.999999 1 1 0.999999 0.999999

Iteration no 18
Solution: 1 1 1 1 1

Iteration no 19
Solution: 1 1 1 1 1

Iteration no 20
Solution: 1 1 1 1 1

Iteration no 21
Solution: 1 1 1 1 1

Iteration no 22
Solution: 1 1 1 1 1

Iteration no 23
Solution: 1 1 1 1 1

Iteration no 24
Solution: 1 1 1 1 1

Iteration no 25
Solution: 1 1 1 1 1

Iteration no 26
Solution: 1 1 1 1 1

Iteration no 27
Solution: 1 1 1 1 1

Iteration no 28
```

..and so on until it hits the last iteration.