Programming Assignment 06

Iterative Jacobi Method for Solving Linear Systems

Nick Handelman
4/18/2018

1 Project Description

In this project, I implemented the matrix form of Jacobi's iterative method for solving linear systems of equations in jacobi.m (Appendix B). The function is called from proj06.m (Appendix A), and the results returned from the function are output in proj06.m.

2 Analysis

jacobi.m accepts a matrix A, solution vector b, initial approximation vector x_0 and a tolerance level TOL. The function decomposes A into components D, L and U (equation (1)). D is the diagonal of A. L and U are the negation of the lower and upper triangle of A, respectively, with the diagonal set to 0 in both. Matrix T_j is calculated by equation (2). Matrix C_j is calculated by equation (3). Jacobi's iterative method approximates the solution of $Ax = b$ by repetitively calculating $x^{(k)}$ in equation (4) where k is the iteration.

$$A = D - L - U \tag{1}$$

$$Tj = D^{-1}(L + U) \tag{2}$$

$$Cj = D^{-1}b \tag{3}$$

$$x^{(k)} = T_j x^{(k-1)} + C_j \tag{4}$$

In each of the following systems, x_0 is a zero vector and $TOL = 10^{-3}$. The iterations continue until equation (5) is true.

$$\frac{||x^{(k)} - x^{(k-1)}||_\infty}{||x^{(k)}||_\infty} < 10^{-3} \tag{5}$$

2.1 Linear System 1

A and b are loaded from files "A.txt" and "b.txt". jacobi.m requires 4 iterations to terminate (see Table 1). The approximate solution after 4 iterations is given in Table 2.

iteration	1	2	3	4
error	1.000000	0.104891	0.006416	0.000775

Table 1: Errors

i	1	2	3	4	5	6	7	8	9
x_i	0.3450723	1.3286782	0.7898246	0.0043637	1.9462537	0.2590304	1.8707790	1.6833014	0.7404628
i	10	11	12	13	14	15	16	17	18
x_i	1.7232881	1.3901596	0.8851502	1.8593649	1.9696130	0.1367843	1.4005939	0.2416643	2.0448782
i	19	20							
x_i	-0.0297488	1.9362999							

Table 2: Solution Vector x

2.2 Linear System 2

jacobi.m requires 12 iterations to terminate (see Table 3). The approximate solution after 12 iterations is given by vector $x_{approximation}$. The exact solution is given by vector x_{exact}.

$$A = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 6 & 3 \\ 1 & 3 & 7 \end{bmatrix} \quad b = \begin{bmatrix} 19 \\ 44 \\ 83 \end{bmatrix} \quad x_{approximation} = \begin{bmatrix} 4.0086 \\ 3.0077 \\ 9.9917 \end{bmatrix} \quad x_{exact} = \begin{bmatrix} 4 \\ 3 \\ 10 \end{bmatrix}$$

i	1	2	3	4	5	6	7	8	9
x_i	1.000000	0.623984	0.227803	0.127518	0.054234	0.030149	0.013739	0.007867	0.003782
i	10	11	12						
x_i	0.002259	0.001141	0.000711						

Table 3: Errors

2.3 Linear System 3

jacobi.m requires 9 iterations to terminate (see Table 4). The approximate solution after 9 iterations is given by vector $x_{approximation}$. The exact solution is given by vector x_{exact}.

$$A = \begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix} \quad b = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix} \quad x_{approximation} = \begin{bmatrix} 0.99967 \\ 2.00045 \\ -1.00037 \\ 1.00062 \end{bmatrix} \quad x_{exact} = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

i	1	2	3	4	5	6	7	8	9
x_i	1.000000	0.576821	0.164319	0.080380	0.028696	0.013511	0.005027	0.002355	0.000888

Table 4: Errors

Appendix A proj06.m

```
A = load('-ascii','A.txt');
b = load('-ascii','b.txt');
x0 = zeros(20,1);
x1 = jacobi(A, b, x0, 10^-3)

A = [3,-1,1;-1,6,3;1,3,7];
b = [19;44;83];
x0 = zeros(3,1);
x1 = jacobi(A, b, x0, 10^-3)

A = [10,-1,2,0;-1,11,-1,3;2,-1,10,-1;0,3,-1,8];
b = [6;25;-11;15];
x0 = zeros(4,1);
x1 = jacobi(A, b, x0, 10^-3)
```

Appendix B jacobi.m

```
%
% Jacobi's iterative method for solving linear systems of equations
%
%  Method used from equation 7.2 (pg 293)
% Faires, J. Douglas, and Richard Burden.
%  "Numerical Methods" fourth ed., (2013).
%
function [x1] = jacobi (A, b, x0, TOL)

% calculate size of array A
nsize = size(A);
n = nsize(1);

% zero out new x vector  x^(k)
%  x0 = x^(k-1)
x1 = zeros(n,1);

% initialize distance between x0 and x1
dist = 1.0;
iter = 1;

%A = D - L - U
D = diag(A); %extract diagonal elements into a vector

if(prod(D) == 0) %D is not invertible if there is a 0 on the diagonal
    return;
endif

D = diag(D); %create diagonal matrix
L = -tril(A, -1);
U = -triu(A, 1);

%D inverse, Tj and Cj
Dinv = inv(D);
Tj = Dinv*(L+U);
Cj = Dinv*b;

while (dist > TOL)

    % Jacobi iterative method goes here
    x1=Tj*x0+Cj;

    % after the iteration, calculate the distance
    % between the vectors
    dist = max( abs(x1 - x0)) / max(abs(x1));

    % print out some useful information
    printf('%d %f\n', iter, dist);

    % assign x^k = x^(k-1)
    x0 = x1;
```

```
    % increment iteration counter
    iter = iter + 1;

endwhile

endfunction
```