The Jacobi Method

Susanne Brenner and Li-Yeng Sung
(modified by Douglas B. Meade)

Department of Mathematics

Overview

This lab, and the next two labs, examine iterative methods for solving a linear system $\mathbf{Ax} = \mathbf{b}$. When the matrix $\mathbf{A}$ satisfies some general criteria and the iterations are selected appropriately, these methods can be very efficient - much faster than the $O(n^3)$ expected from basic Gaussian elimination.

New `MATLAB` commands introduced in this lab are `zeros`, to create a zero matrix, and the timing commands `tic` and `toc`.

Part I

The general iterative method for solving $\mathbf{Ax} = \mathbf{b}$ is defined in terms of the following iterative formula:

$$\mathbf{Sx}^{\mathbf{new}} = \mathbf{b} + \mathbf{Tx}^{\mathbf{old}}$$

where $\mathbf{A} = \mathbf{S} - \mathbf{T}$ <u>and</u> it is fairly easy to solve systems of the form $\mathbf{Sx} = \mathbf{b}$.

The Jacobi method exploits the fact that diagonal systems can be solved with one division per unknown, i.e., in $O(n)$ flops. To implement Jacobi's method, write $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$ where $\mathbf{D}$ is the $n \times n$ matrix containing the diagonal of $\mathbf{A}$, $\mathbf{L}$ is the $n \times n$ matrix containing the lower triangular part of $\mathbf{A}$, and $\mathbf{U}$ is the $n \times n$ matrix containing the upper triangular part of $\mathbf{A}$. (Note that this is *not* the $\mathbf{L}$–$\mathbf{D}$–$\mathbf{U}$ factorization of $\mathbf{A}$.) Let $\mathbf{S}$ be the diagonal part of $\mathbf{A}$, $\mathbf{S} = \mathbf{D}$. Then, $\mathbf{T} = \mathbf{S} - \mathbf{A} = -(\mathbf{L} + \mathbf{U})$.

<u>A 5×5 Example</u>

Consider $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} = \begin{bmatrix} 6 & 1 & 1 & 1 & 1 \\ 1 & 7 & 1 & 1 & 1 \\ 1 & 1 & 8 & 1 & 1 \\ 1 & 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 1 & 10 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} -10 \\ -6 \\ 0 \\ 8 \\ 18 \end{bmatrix}$.

The exact solution for $\mathbf{Ax} = \mathbf{b}$ is $\mathbf{x} = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$.

The following `MATLAB` commands compute the first ten (10) Jacobi iterations with initial guess $\mathbf{x} = \mathbf{0}$:

```
≫ S = diag( diag(A) )          % diagonal matrix formed from diag(A)
≫ T = S - A                    % A = S − T
≫ x = zeros(size(b))           % initial guess is x = 0
≫ for k=1:10,
≫   x = S \ (b+T*x)            % perform one Jacobi iteration
≫ end
```

It should be apparent that the vectors `x` are converging to the exact solution of $\mathbf{Ax} = \mathbf{b}$.

Iterations Controlled by Preset Tolerance

The number of iterations needed before it is possible to conclude that an iterative method has converged depends on the particular system and the selection of the matrices **S** and **T**. An estimate of the relative error can be used to detect convergence before the maximum number of iterations have been executed.

Use the MATLAB Editor to create jacobi1.m that contains the following commands (the comments are not necessary):

```
% jacobi1.m - MATLAB script file for Lab 09

% MATLAB script that executes iterations of Jacobi's method to solve Ax = b.
% The matrix A and vector b are assumed to already be assigned values in the
% MATLAB session. The iterates are stored in the matrix X.

clear X                                  % remove existing value of X
tol = 1.e-8;                             % preset tolerance (10^-8)
maxiter = 100;                           % maximum number of iterations

relerr = inf;                            % initialize relative error to large value
niter = 1;                               % initialize iteration counter

S = diag( diag(A) );                     % diagonal of A
T = S - A;                               % off-diagonal of A

X(:,1) = zeros(size(b));                 % initial guess (x = 0)
while relerr > tol & niter< maxiter,     % iterate until convergence, or maximum iterations
   X(:,niter+1) = S \ (b+T*X(:,niter));
   relerr = norm(X(:,niter+1)-X(:,niter),inf)/norm(X(:,niter+1),inf);
   niter = niter+1;                      % increment iteration counter
end

fprintf( '\n\nAfter %g iterations of Jacobi''s method the relative error is %g.\n', niter, relerr )
```

Be sure you save this file as jacobi1.m.

Return control to the MATLAB Command Window and type the following:

```
>> format long g                         % select format for output
>> jacobi1                               % execute script in M-file
```

This script stores the vectors **x** for each iteration in the columns of the matrix X. If you wish to see the matrix of all iterations of Jacobi's method, enter the following MATLAB command:

```
>> X                                     % display matrix of iterations
```

Timing MATLAB Commands

To determine the elapsed time to execute the jacobi1 script execute the script as follows:[1]

```
>> tic, jacobi1, toc
```

Note that the count of floating-point operations can be obtained similarly:

```
>> tic, flops(0), jacobi1, flops, toc
```

Clear all variables before you begin to work on Part II.

[1] On faster computers the elapsed time for this problem is likely to be reported as 0 seconds. To increase the likelihood a positive elapsed time will be reported, work with a larger problem or a lower tolerance.