The Gauss–Seidel Method

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

Overview

The investigation of iterative solvers for $\mathbf{Ax} = \mathbf{b}$ continues with a look at the Gauss–Seidel method. Each Gauss–Seidel iteration requires $O(n^2)$ flops. (Jacobi's method requires $O(n)$ flops per iteration; a more detailed comparison of these two methods is the focus of Lab 11.)

New `MATLAB` commands introduced in this lab include `tril` and `triu`, to extract the lower- and upper-triangular parts of a matrix, and `sparse` and `full`, to work with matrices with lots of zeros (sparse matrices).

Part I

As with the Jacobi method, write $\mathbf{A} = \mathbf{L}+\mathbf{D}+\mathbf{U}$ where $\mathbf{L}$ is the $n \times n$ matrix containing all components below the main diagonal of $\mathbf{A}$, $\mathbf{D}$ is the $n \times n$ matrix containing the diagonal components of $\mathbf{A}$, and $\mathbf{U}$ is the $n \times n$ matrix containing all components above the main diagonal of $\mathbf{A}$.

The Gauss–Seidel method uses $\mathbf{S} = \mathbf{L} + \mathbf{D}$ and $\mathbf{T} = \mathbf{S} - \mathbf{A}$. Thus, in each step forward substitution is needed to solve

$$\mathbf{S}\mathbf{x}^{\mathbf{new}} = \mathbf{b} + \mathbf{T}\mathbf{x}^{\mathbf{old}}.$$

A 5×5 Example

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) Gauss–Seidel iterations with initial guess $\mathbf{x} = \mathbf{0}$.

```
>> S = tril( A )                % Gauss-Seidel method
>> 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 Gauss-Seidel iteration
>> end
```

It should be apparent that the vectors `x` are converging to the exact solution of $\mathbf{Ax} = \mathbf{b}$. (Compare the iterates and convergence with those obtained with Jacobi's method for this same problem in Lab 9.)

Block Construction of Matrices

As matrices become larger it becomes increasingly tedious to enter every component of the matirx. In some cases it is possible to decompose a matrix into simpler blocks. For example,

$$\mathbf{A} = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 5 \\ 1 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 8 \end{bmatrix} = \begin{bmatrix} 8\mathbf{I} & \mathbf{B} \\ \mathbf{B} & 8\mathbf{I} \end{bmatrix}$$

where $\mathbf{I} = \mathbf{I}_{5\times 5}$ and $\mathbf{B}$ is the 5×5 diagonal matrix with $b_{kk} = k$ for k=1, 2, 3, 4, 5.

The following commands can be used to enter $\mathbf{A}$ in `MATLAB`:

```
>> I = eye(5,5)                    % I_{5x5}
>> B = diag(1:5)
>> A = [ 9*I B ; B 9*I ]           % block definition of A
```

Sparse Matrices

A sparse matrix is a matrix with very few non-zero components. Typically, the number of non-zero components in a sparse matrix is of order n. Sparse matrices are particularly well-suited for iterative methods. The savings typically arises from efficient implementations of matrix multiplication that avoids any multiplication by zero. `MATLAB` can store sparse matrices economically and can perform matrix operations involving sparse matrices efficiently.

The `sparse` commands converts an existing `MATLAB` matrix into the corresponding sparse representation.

```
>> Asp = sparse( A )               % create sparse version of A
```

Note that the matrix $A_{\mathbf{sp}}$ is presented in terms of the non-zero components and their indices. (See `help sparse` to learn more about the `sparse` command and sparse matrices in `MATLAB`.)

One benefit of using sparse matrices is illustrated by matrix multiplication:

```
>> b = ones(10,1)
>> flops(0), Asp*b,   flops        % sparse matrix-vector product
>> flops(0), A*b,     flops        % full matrix-vector product
>> flops(0), Asp*Asp, flops        % sparse matrix-matrix product
>> flops(0), A*A,     flops        % full matrix-matrix product
```

The sparse format of a matrix is preserved by matrix operations such as transpose, `diag`, `tril`, and `lu`.

```
>> Asp'                            % (sparse) transpose of A
>> diag(Asp)                       % (sparse) diagonal comp of A
>> [Lsp,Usp] = lu(Asp)             % (sparse) L-U factorization of A
```

The `full` command converts from sparse format to the full representation of a matrix.

```
>> L = full( Lsp )                 % full representation of L
```

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