Sparse Matrices — Basic Iterative Methods
Matrix Computations — CPSC 5006 E

Julien Dompierre

Department of Mathematics and Computer Science
Laurentian University

Sudbury, December 6, 2010

Sparse Matrices — Basic Iterative Methods

- Jacobi.
- Gauss-Seidel.
- SOR.
- SSOR.
- Saad. Iterative Methods for Sparse Linear System sections 4.1 and 4.2.
- Golub-Van Loan, section 10.1.
- Barrett et al. Templates for the Solution of Linear Systems, section 2.2.

Jacobi (p. 510)

Perhaps the simplest iterative scheme is the **Jacobi iteration**. It is defined for matrices that have non zero diagonal elements[1]. The method can be motivated by rewritting the 3-by-3 system $Ax = b$ as follows:

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

Suppose $x^{(k)}$ is an approximation to $x = A^{-1}b$. A natural way to generate a new approximation $x^{(k+1)}$ is to compute:

$$\begin{aligned} x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})/a_{11} \\ x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})/a_{22} \\ x_3^{(k+1)} &= (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})/a_{33} \end{aligned}$$

[1] This assumption can be reduced

Jacobi (p. 510)

The general equation is

$$x_i^{(k+1)} = \left(b_i - \sum_{\substack{j=1,\ldots,n \\ j\neq i}} a_{ij} x_j^{(k)} \right) / a_{ii}$$

The sum can be divided in two parts, the first part up to $i-1$ and the second part starting à $i+1$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}$$

Jacobi Update

Note that in the Jacobi iteration one does not use the most recently available information when computing $x_i^{(k+1)}$. For example $x_1^{(k)}$ is used in the calculation of $x_2^{(k+1)}$ even though component $x_1^{(k+1)}$ is known.

This allows to update the vector $x^{(k+1)}$, from the vector $x^{(k)}$, looping over the component in any order. It also allows easy parallelization on multiple processors.

Jacobi (p. 510)

For general n, we have the Jacobi algorithm:

Algorithm 1 Jacobi. $A \in \mathbb{R}^{n \times n}$ with non zero diagonal elements. $b \in \mathbb{R}^n$, $x^{(0)}$, an initial guess, ε a stopping criterion and *MaxIter*, the maximum number of iterations if it does not converge.

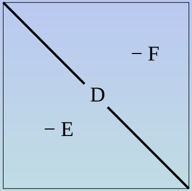
```
for k = 1 : MaxIter
    for i = 1 : n
        x_i^(k+1) = ( b_i − Σ_{j=1}^{i−1} a_ij x_j^(k) − Σ_{j=i+1}^{n} a_ij x_j^(k) ) a_ii^{−1}
    end
    if ‖x^(k+1) − x^(k)‖ < ε then
        Break
    end
end
```

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) a_{ii}^{-1}$$

Matrix Form of the Jacobi Method (p. 511)

Relaxation schemes: based on the decomposition

$$A = D - E - F$$



$D = \text{diag(A)};$
$E =$ negative of the strict lower part of A;
$F =$ negative of the strict upper part of A.

Simplest method for solving $Ax = b$: Jacobi iteration

$$\begin{aligned} Dx^{(k+1)} &= (E+F)x^{(k)} + b \\ x^{(k+1)} &= D^{-1}((E+F)x^{(k)} + b) \\ x^{(k+1)} &= D^{-1}(E+F)x^{(k)} + D^{-1}b \end{aligned}$$

Analysed using iteration matrix $M_{Jac} = D^{-1}(E+F)$.

Jacobi (Matrix Form) (p. 510)

For general n, we have the matrix form of the Jacobi algorithm:

Algorithm 2 Jacobi. $M_{Jac} = D^{-1}(E+F) \in \mathbb{R}^{n\times n}$. $b_{Jac} = D^{-1}b \in \mathbb{R}^n$, $x^{(0)}$, an initial guess, ε a stopping criterion and *MaxIter*, the maximum number of iterations if it does not converge.

---

for $k = 1 : MaxIter$
    $x^{(k+1)} = M_{Jac}x^{(k)} + b_{Jac}$
    if $\|x^{(k+1)} - x^{(k)}\| < \varepsilon$ **then**
        Break
    end
end

Note: This algorithm contains only a matrix-vector update.

Jacobi:

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}$$

Idea: Use the newest value of x_i^{k+1} as soon as it is available.

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}$$

This gives the Gauss-Seidel method.

Gauss-Seidel Update

Because the Gauss-Seidel method uses the most recently available information when computing $x_i^{(k+1)}$, it usually converge faster than the Jacobi method.

However, because we cannot compute $x_i^{(k+1)}$ before $x_{i-1}^{(k+1)}$ is computed, parallelization on multiple processors is not easy. Also, the Gauss-Seidel method depends on the traversal order of the unknowns. The classics are the forward loop from 1 to n, the reverse loop from n to 1, a combination of both, a random traversal, etc.

Gauss-Seidel (p. 510)

For general n, we have the Gauss-Seidel algorithm:

Algorithm 3 Gauss-Seidel. $A \in \mathbb{R}^{n\times n}$ with non zero diagonal elements. $b \in \mathbb{R}^n$, $x^{(0)}$, an initial guess, ε a stopping criterion and *MaxIter*, the maximum number of iterations if it does not converge.

for $k = 1 : MaxIter$
    for $i = 1 : n$
        $x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(\mathbf{k+1})} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) a_{ii}^{-1}$
    end
    if $\|x^{(k+1)} - x^{(k)}\| < \varepsilon$ **then**
        Break
    end
end

Gauss-Seidel

Idea: correct the i-th component of the current approximate solution, $i = 1, 2, ..n$, to zero out $i - th$ component of residual.

Gauss-Seidel: $$x_i^{new} = \frac{1}{a_{ii}} \left[b_i - \sum_{j<i} a_{ij} x_j^{new} - \sum_{j>i} a_{ij} x_j^{old} \right]$$

Matrix form of Gauss-Seidel:

$$(D - E)x^{(k+1)} = Fx^{(k)} + b$$

Analysed using iteration matrix $M_{GS} = (D - E)^{-1}(F)$.

Can also define a backward Gauss-Seidel Iteration:

$$(D - F)x^{(k+1)} = Ex^{(k)} + b$$

and a Symmetric Gauss-Seidel Iteration: forward sweep followed by backward sweep.

Relaxation

Relaxation is based on relaxing the Gauss-Seidel iteration:

- $x_i^{(k+1)} = \xi_i^{(k)} + \omega(\xi_i^{\mathrm{GS}} - \xi_i^{(k)})$
- $0 < \omega < 1 \;\Leftrightarrow$ Under-relaxation.
- $\omega = 1 \;\Leftrightarrow$ Gauss-Seidel.
- $1 < \omega < 2 \;\Leftrightarrow$ Over-relaxation.

Over-relaxation is based on the decomposition:

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega) D)$$

$\longrightarrow$ successive overrelaxation, (SOR):

$$(D - \omega E) x^{(k+1)} = [\omega F + (1 - \omega) D] x^{(k)} + \omega b$$

corresponding to iteration matrix

$$M_{\omega SOR} = (D - \omega E)^{-1} (\omega F + (1 - \omega) D).$$

Iteratives Matrices

<u>Iteration matrices</u>

In Jacobi, Gauss-Seidel, or SOR, the iteration is of the form,

$$x^{(k+1)} = Mx^{(k)} + f$$

where

- $M_{Jac} = D^{-1}(E + F) = I - D^{-1}A$
- $M_{GS}(A) = (D - E)^{-1}F = I - (D - E)^{-1}A$
- $$\begin{aligned} M_{\omega SOR}(A) &= (D - \omega E)^{-1}(\omega F + (1 - \omega)D) \\ &= I - (\omega^{-1}D - E)^{-1}A \end{aligned}$$

Convergence

Jacobi and Gauss-Seidel converge for diagonal dominant matrices

SOR converges for $0 < \omega < 2$ for SPD matrices

Optimal ω known for 'consistently ordered matrices' (eig-vals of $\alpha^{-1}D^{-1}E + \alpha D^{-1}F$ indep. of α):

$$\omega_{\text{optimal}} = \frac{2}{1 + \sqrt{1 - \rho(M_{Jac})^2}}.$$