Reference: http://www.numerical-methods.com/linalg.htm

Jacobi & Gauss-Seidel methods for Linear Systems of Equations

by numerical-methods.com

A system of linear algebraic equations has the form

a[1,1] x[1] + a[1,2] x[2] + ... + a[1,N] x[N] = b[1]
a[2,1] x[1] + a[2,2] x[2] + ... + a[2,N] x[N] = b[2]
a[M,1] x[1] + a[M,2] x[2] + ... + a[M,N] x[N] = b[M]

where the a[i,j] and the b[i] are known. The x[j] are unknown and the purpose of solving the system is to find the x[j].

The system is often written in the form A x = b where A is an MxN matrix and x is an N-vector and b M-vectors.

Tutorials on Matrices

Solvability of the Linear System

Whether the solution is possible and the performance of the numerical solution methods all hinges on the nature of the matrix A.

Non-square matrix

If the matrix has more columns than rows (N>M) then there is a vector space of solutions (no unique solution); there are more unknowns than equations!
If the matrix has more rows than columns then the linear system is said to be "overdetermined". This can be confusing however. For example some of the equations may be linearly dependent or even simply repeated; for example if all the rows were the same then this does not help us determine a solution.
Often in an overdetermined system, there is no solution x that satisfies all the rows exactly but there are solutions x such that Ax-b is a vector of "small values" that are within working accuracy.

Square matrix (M=N)

If the matrix A has rows that can be formed from a linear combination of other rows then it is said to be singular. In this case the linear system either has no solution or no unique solution.
If the matrix A is nearly-singular (or ill-conditioned) then there will be a range of "numerical solutions" such that A x - b is within working numerical accuracy and a precise solution will be difficult to determine.
Otherwise the system has a well-defined unique solution.

Methods of Solution

There are two distinct approached to solving a system of equations; direct methods and iterative methods. In direct methods a finite series of operations are carried out (usually O(N^3)) and the solution is arrived at. The most well-known method of this type is Gaussian elimination. LU factorisation followed by back-substitution is a related method. Iterative methods involve a sequence of matrix-vector multiplications. Starting with an initial guess at the solution, each multiplication or iteration returns a new estimate of the solution and clearly takes O(N^2) operations. Hopefully the estimates get closer and closer to the solution so that after k iterations say a satisfactory solution is arrived at (after O(kN^2) operations).

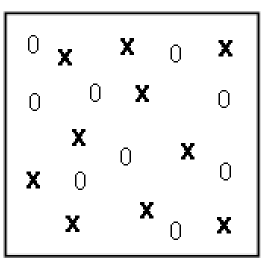
Software

Several Interactive Programs have become available, enabling users to carry out linear algebra without resorting to traditional programming languages. Leading examples of these are Matlab and Gauss.

Here the improved values of x_i are utilized as soon as they are obtained.

Linear Systems of Equations
Iterative Methods

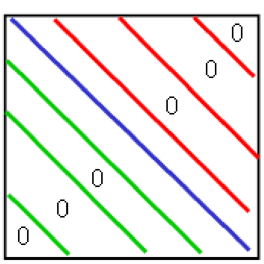
Sparse, Full-bandwidth Systems



Rewrite Equations

$$\overline{\overline{\mathbf{A}}}\overline{\mathbf{x}} = \overline{\mathbf{b}} \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij} x_j = b_i$$

$$a_{ii} \neq 0 \Rightarrow x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}} \; , \; i = 1, \ldots n$$



Iterative, Recursive Methods

Jacobi's Method

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}} \; , \; i = 1, \ldots n$$

Gauss-Seidels's Method

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}} \; , \; i = 1, \ldots n$$

$$\left.\begin{aligned} 3x_1 + x_2 + x_3 &= 8 \\ x_1 + 4x_2 + 2x_3 &= 15 \\ 2x_1 + x_2 + 5x_3 &= 19 \end{aligned}\right\} \quad . \tag{2.3.5}$$

For these equations, the solution under the Gauss-iteration scheme represented by equations (2.3.2) takes the form

$$\left.\begin{aligned} x_1^{(k+1)} &= \left[8 - x_2^{(k)} - x_3^{(k)}\right]\Big/3 \\ x_2^{(k+1)} &= \left[15 - x_1^{(k)} - 2x_3^{(k)}\right]\Big/4 \\ x_3^{(k+1)} &= \left[19 - 2x_1^{(k)} - x_2^{(k)}\right]\Big/5 \end{aligned}\right\} \quad . \tag{2.3.6}$$

However, if we were to solve equations (2.3.5) by means of the Gauss-Seidel method the iterative equations for the solution would be

$$\left.\begin{aligned} x_1^{(k+1)} &= \left[8 - x_2^{(k)} - x_3^{(k)}\right]\Big/3 \\ x_2^{(k+1)} &= \left[15 - x_1^{(k+1)} - 2x_3^{(k)}\right]\Big/4 \\ x_3^{(k+1)} &= \left[19 - 2x_1^{(k+1)} - x_2^{(k+1)}\right]\Big/5 \end{aligned}\right\} \quad . \tag{2.3.7}$$

For these equations, the solution under the Gauss-iteration scheme represented by equations (2.3.2) takes the form (2.3.6)

However, if we were to solve equations (2.3.5) by means of the Gauss-Seidel method the iterative equations for the solution would be (2.3.7)

If we take the initial guess to be

$$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 1 \ , \tag{2.3.8}$$

then repetitive use of equations (2.3.6) and (2.3.7) yield the results given in Table 2.1.

If we take the initial guess to be

$$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 1 \,, \tag{2.3.8}$$

then repetitive use of equations (2.3.6) and (2.3.7) yield the results given in Table 2.1.

Table 2.1

Convergence of Gauss and Gauss-Seidel Iteration Schemes

K	0		1		2		3		4		5		10	
	G	GS	G	GS	G	GS	G	GS	G	GS	G	GS	G	GS
x_1	1.00	1.00	2.00	2.00	0.60	0.93	1.92	0.91	0.71	0.98	1.28	1.00	0.93	1.00
x_2	1.00	1.00	3.00	2.75	1.65	2.29	2.64	2.03	1.66	1.99	2.32	2.00	1.92	2.00
x_3	1.00	1.00	3.20	2.45	1.92	2.97	3.23	3.03	2.51	3.01	3.18	3.00	2.95	3.00