ITERATIVE METHODS FOR SOLUTION OF LINEAR EQUATIONS

Marco Guerra Hinostroza[1]

[1]*Universidade Federal de Viçosa, Viçosa – <marco.hinostroza@ufv.br>*

1. THE FIXED-POINT ITERATION

The fixed point iteration method in numerical analysis is used to find an approximate solution to algebraic equations.

Problem

- Solve Ax = b approx.; A $\in R^{nxn}$.
- n large.
- A is sparse, small of non-zeros.

Suppose we have an equation $f(x) = 0$, for which we have to find the solution. The equation can be expressed as $x = g(x)$. Choose $g(x)$ such that $|g'(x)| < 1$ at $x = x_0$ where x_0 is some initial guess called fixed point iterative scheme. Then the iterative method is applied by successive approximations given by $x_n = g(x_n - 1)$, that is, $x_1 = g(x_o)$, $x_2 = g(x_1)$ and so on.

Algorithm of Fixed Point Iteration Method

- Choose the initial value xo for the iterative method. One way to choose x_0 is to find the values $x = a$ and $x = b$ for which $f(a) < 0$ and $f(b) > 0$. By narrowing down the selection of a and b, take x_0 as the average of a and b.
- Express the given equation, in the form $x = g(x)$ such that $|g'(x)| < 1$ at $x = x_0$. If there more than one possibility of $g(x)$, choose the $g(x)$ which has the minimum value of $g'(x)$ at $x = x_0$.
- By applying the successive approximations $x_n = g(x_n - 1)$, if f is a continuous function, we get a sequence of x_n which converges to a point, which is the approximate solution of the given equation.

2. JACOBI METHOD

Jacobi Iteration is defined as the simplest iterative method used to solve linear systems of equations. It involves computing the approximation of each variable in terms of the previous approximation and the other variables in the system. Although it is the slowest method, it serves as a foundation for understanding other iterative methods like Gauss-Seidel.

$$a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n = b_2$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n = b_n$$

Step 1: In this method, we must solve the equations to obtain the values x_1, x_2,... x_n.

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)$$

$$x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_2 - a_{23}x_3 - \cdots - a_{2n}x_n)$$

$$x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_2 - a_{n2}x_3 - \cdots - a_{n,n-1}x_{n-1})$$

Step 2: Now, we have to make an initial guess of the solution as follows:

$$x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, ..., x_n^{(0)})$$

Step 3: Substitute the values obtained in the previous step in the equations of Step 1 to obtain the first approximation as:

$$\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_n^{(1)}\right)$$

Step 4: Proceed in the same way with new approximations, and the iteration continues until a defined stopping criterion is reached.

$$x^k = \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \dots, x_n^{(k)}\right); \quad k = 1, 2, 3, \dots$$

3. GAUSS-SEIDEL METHOD

The Guass-Seidel method is a modification of the Jacobi method. This modification usually results in a higher degree of accuracy with fewer iterations.

In Jacobi's method the value of the variables does not change until the next iteration, while in the Gauss-Seidel method the value of the variables changes as soon as the new value is evaluated. For example, in Jacobi's method the value of $x_i^{(k)}$ does not change until the $(k+1)$th iteration, but in the Gauss-Seidel method the value of $x_i^{(k)}$ changes at kth iteration.

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

Discussion:

- Converge fast.
- Sequential, not good for parallel computing.

4. STOP CRITERIA

Stop criteria (or stopping conditions) in iterative methods are essential to determine when the iterative process should halt. The goal is to stop when the solution is "close enough"to the true solution, without over-computing or prematurely ending the process. Properly chosen stop criteria ensure that the method provides an accurate solution within reasonable computational time.

Common stopping criteria

- Maximum number of iterations.
- Change in successive iterates.

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| < \epsilon$$

- Residual.

$$r^k = Ax^k - b$$
$$\|\mathbf{r}^{(k)}\| < \epsilon$$

Where ϵ is a small positive tolerance value, the tolerance ϵ should be set very small (e.g., 10^{-06} or smaller).

5. EXAMPLE

$$10x_1 + 2x_2 + x_3 = 7$$

$$x_1 + 5x_2 + x_3 = -8$$

$$2x_1 + 3x_2 + 10x_3 = 6$$

Step 1:

$$x_1 = \frac{1}{10}(7 - 2x_2 - x_3)$$

$$x_2 = \frac{1}{5}(-8 - x_1 - x_3)$$

$$x_3 = \frac{1}{10}(6 - 2x_1 - 3x_2)$$

Step 2:

$$x^{(0)} = (0, 0, 0)$$

Step 3:

Iteration k=1

$$x_1 = \frac{1}{10}(7 - 2(0) - 0)$$

$$x_1 = 0.7$$

$$x_2 = \frac{1}{5}(-8 - 0.7 - 0)$$

$$x_2 = -1.74$$

$$x_3 = \frac{1}{10}(6 - 2(0.7) - 3(-1.74))$$

$$x_3 = 0.982$$

$$x^{(1)} = (0.7, -1.74, 0.982)$$

Iteration k=2

$$x_1 = \frac{1}{10}(7 - 2(0.7) - (-1.74))$$

$$x_1 = 0.734$$

$$x_2 = \frac{1}{5}(-8 - 0.734 - 0.982)$$

$$x_2 = -1.943$$

$$x_3 = \frac{1}{10}(6 - 2(0.734) - 3(-1.943))$$

$$x_3 = 1.036$$

$$x^{(2)} = (0.734, -1.943, 1.036)$$

Code 1. Example code in R.

```r
library(fastmatrix)
X <- matrix(c(10,2,1,
              1,5,1,
              2,3,10),
            nrow = 3,
            ncol = 3,
            byrow = TRUE)

Y <- matrix(c(7,-8,6),
            nrow = 3,
            ncol = 1)

a <- t(X)%*%X
b <- as.vector(t(X)%*%Y)

start <- rep(0, length(b))

# Tolerance = 0.05
ajuste <- seidel(a, b, start,
                 maxiter = 200,
                 tol = 0.05)
# Output
# 0.9793988 -1.9691503  0.9950706
# iterations 6

# Tolerance = 1e-7
ajuste <- seidel(a, b, start,
                 maxiter = 200,
                 tol = 1e-7)
# Output
# 0.9999999 -1.9999999  1.0000000
# iterations 18

# Change of initial values
start <- c(1,1,1)
ajuste <- seidel(a, b, start,
                 maxiter = 200,
                 tol = 1e-7)
# Output
# 1 -2  1
# iterations 16
```

Discussion:
Reducing the tolerance may require more iterations, but the solution will be more optimal. Change the initial values may change the number of iterations until the stop criterion is reached, better initial values generally reduce the number of iterations but does not mean it is a better solution.