Successive over relaxation method

In numerical linear algebra, the method of successive over-relaxation (SOR) is a variant of the Gauss–Seidel method for solving a linear system of equations, resulting in faster convergence. A similar method can be used for any slowly converging iterative process.

Procedure

1. Choose an **n X (n+1)** matrix

Consider the augmented matrix **[A: b]**; where A is an **n X n** matrix and **b** is an **n X 1** matrix.

Let's assume the provided equations are

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

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

$a_{31}x_1 + a_{32}x_2 + . . . + a_{3n}x_n = b_3$

. . .

$A_{n1}x_1 + a_{n2}x_2 + . . . + a_{nn}x_n = b_n$

In matrix form for the aforementioned square system of *n* linear equations with unknown column vector **x**

Ax = b

$$A = \begin{bmatrix} a11 & a12 & \dots & a1n \\ a21 & a22 & \dots & a2n \\ \dots & \dots & \dots & \dots \\ an1 & an2 & \dots & ann \end{bmatrix};\ x = \begin{bmatrix} x1 \\ x2 \\ \dots \\ xn \end{bmatrix};\ b = \begin{bmatrix} b1 \\ b2 \\ \dots \\ bn \end{bmatrix};$$

Otherwise- Show Pop up – please select the number of columns that is one more than the number of rows

2. Verify that the magnitude of the diagonal item in each row of the matrix A is greater than or equal to the sum of the magnitudes of all other (non-diagonal) values in that row so that

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i$$

Otherwise- Show Pop up – entered matrix is not diagonally dominant

(Instruct the user to rearrange the rows in order to satisfy the aforementioned criteria or you can use coding to rearrange the rows.)

Ensure that all of the diagonal elements are non-zero as well.

$$\boldsymbol{a_{ii} \neq 0}$$

Otherwise- Show Pop up – all of the diagonal elements must be non-zero

3. Initialize the solution vector **x** with an initial guess.

4. For each iteration, repeat the following steps for each equation in the system:
    a. Calculate the new values for vector **x** by using the following formula:

    x_new[i][1] = (1 - w) * **x_old[i][1]** + (w/A[i][i]) * (b[i][1] - sum(A[i][j] * **x_old[j][1]** for j in range(n) if j != i))

Where w is the relaxation factor, n is the number of unknowns, x_old[i] is the current value of x[i], and x_new[i] is the updated value of x[i]. It is analogous to if x[i] is the value we obtain after k^{th} iteration process then x_new[i] is going to be the value of updated x[i] after $(k + 1)^{th}$ iteration.

Expression x[i] is analogous to x_1, x_2, or x_n if i = 1, 2, or n and so on as mentioned in column vector **x** in step 1.

Ask the user to choose a relaxation factor value between 1 and 2, such as 1.25, 1.5, etc.

5. Check for convergence. The method has converged when the difference between the old and new solutions is below a certain tolerance or threshold
6. Update the solution vector x with the new values from the previous step.

7. Repeat steps 2 to 6 until convergence or a maximum number of iterations is reached.
8. If convergence is reached, the solution vector x will contain the solution to the system of linear equations.

Example:

Consider the following system of linear equations:

3x + y = 9

x + 2y = 8

We can represent this system of equations as a matrix equation:

A * **x** = **b**

where **A** = $\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$; **b** = $\begin{bmatrix} 9 \\ 8 \end{bmatrix}$;

The initial guess for the solution vector **x** is [0, 0].

Choose a relaxation factor w. A value between 1 and 2 is commonly used. For this example, let's use w = 1.5

Repeat the following steps until convergence:

1st iteration

x_new = [(1 - w) * x[1][1] + (w/A[1][1]) * (b[1][1] - (A[1][2] * x[2][1])),

(1 - w) * x[2][1] + (w/A[2][2]) * (b[2][1] - (A[2][1] * x[1][1]))]

(Above expression is analogous to **x_new** = $\begin{bmatrix} x1 \\ x2 \end{bmatrix}$*;)*

= [(1 - 1.5) * 0 + (1.5/3) * (9 - (1 * 0)), (1 - 1.5) * 0 + (1.5/2) * (8 - (1 * 0))]

= $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$

So, now, x[1][1] = 3 and x[2][1] = 4

2^{nd} iteration

x_new = [(1 - w) * x[1][1] + (w/A[1][1]) * (b[1][1] - (A[1][2] * x[2][1])),

(1 - w) * x[2][1] + (w/A[2][2]) * (b[2][1] - (A[2][1] * x[1][1]))]

= [(1 - 1.5) * 3 + (1.5/3) * (9 - (1 * 4)),   (1 - 1.5) * 4 + (1.5/2) * (8 - (1 * 3))]

$= \begin{bmatrix} 1 \\ 1.75 \end{bmatrix}$

Continue the iteration process until the difference between the old and new solutions of vector **x** is below a certain tolerance or threshold. If convergence is reached, the solution vector **x** (of final iteration) will contain the solution to the system of linear equations. From a programming perspective, you can continue using the 10e-8 threshold value.