



# Simultaneous Linear Equations

---



Topic: Gauss-Seidel Method

Major: Computer Engineering



# Gauss-Seidel Method

---

An iterative method.

Basic Procedure:

- Algebraically solve each linear equation for  $x_i$
- Assume an initial guess solution array
- Solve for each  $x_i$  and repeat
- Use absolute relative approximate error after each iteration to check if error is within a pre-specified tolerance.



# Gauss-Seidel Method

---

## Why?

The Gauss-Seidel Method allows the user to control round-off error.

Elimination methods such as Gaussian Elimination and LU Decomposition are prone to round-off error.

Also: If the physics of the problem are understood, a close initial guess can be made, decreasing the number of iterations needed.



# Gauss-Seidel Method

---

## Algorithm

A set of  $n$  equations and  $n$  unknowns:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n = b_2$$

$$\vdots \quad \quad \quad \vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n = b_n$$

If: the diagonal elements are non-zero

Rewrite each equation solving for the corresponding unknown

ex:

First equation, solve for  $x_1$

Second equation, solve for  $x_2$

# Gauss-Seidel Method

## Algorithm

Rewriting each equation

$$x_1 = \frac{c_1 - a_{12}x_2 - a_{13}x_3 \dots - a_{1n}x_n}{a_{11}}$$

← From Equation 1

$$x_2 = \frac{c_2 - a_{21}x_1 - a_{23}x_3 \dots - a_{2n}x_n}{a_{22}}$$

← From equation 2

⋮      ⋮      ⋮

$$x_{n-1} = \frac{c_{n-1} - a_{n-1,1}x_1 - a_{n-1,2}x_2 \dots - a_{n-1,n-2}x_{n-2} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

← From equation n-1

$$x_n = \frac{c_n - a_{n1}x_1 - a_{n2}x_2 - \dots - a_{n,n-1}x_{n-1}}{a_{nn}}$$

← From equation n



# Gauss-Seidel Method

## Algorithm

General Form of each equation

$$x_1 = \frac{c_1 - \sum_{\substack{j=1 \\ j \neq 1}}^n a_{1j} x_j}{a_{11}}$$

$$x_2 = \frac{c_2 - \sum_{\substack{j=1 \\ j \neq 2}}^n a_{2j} x_j}{a_{22}}$$

$$x_{n-1} = \frac{c_{n-1} - \sum_{\substack{j=1 \\ j \neq n-1}}^n a_{n-1,j} x_j}{a_{n-1,n-1}}$$

$$x_n = \frac{c_n - \sum_{\substack{j=1 \\ j \neq n}}^n a_{nj} x_j}{a_{nn}}$$



# Gauss-Seidel Method

---

## Algorithm

General Form for any row 'i'

$$x_i = \frac{c_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j}{a_{ii}}, i = 1, 2, \dots, n.$$

How or where can this equation be used?



# Gauss-Seidel Method

---

Solve for the unknowns

Assume an initial guess for  $[X]$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

Use rewritten equations to solve for each value of  $x_i$ .

Important: Remember to use the most recent value of  $x_i$ . Which means to apply values calculated to the calculations remaining in the **current** iteration.



# Gauss-Seidel Method

---

Calculate the Absolute Relative Approximate Error

$$|\epsilon_a|_i = \left| \frac{X_i^{\text{new}} - X_i^{\text{old}}}{X_i^{\text{new}}} \right| \times 100$$

So when has the answer been found?

The iterations are stopped when the absolute relative approximate error is less than a pre-specified tolerance for all unknowns.



# Example: Surface Shape Detection

---

To infer the surface shape of an object from images taken of a surface from three different directions, one needs to solve the following set of equations

$$\begin{bmatrix} 0.2425 & 0 & -0.9701 \\ 0 & 0.2425 & -0.9701 \\ -0.2357 & -0.2357 & -0.9428 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 247 \\ 248 \\ 239 \end{bmatrix}$$

The right hand side are the light intensities from the middle of the images, while the coefficient matrix is dependent on the light source directions with respect to the camera. The unknowns are the incident intensities that will determine the shape of the object.

Find the values of  $x_1$ ,  $x_2$ , and  $x_3$  use the Gauss-Seidel method.



# Example: Surface Shape Detection

The system of equations is:

$$\begin{bmatrix} 0.2425 & 0 & -0.9701 \\ 0 & 0.2425 & -0.9701 \\ -0.2357 & -0.2357 & -0.9428 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 247 \\ 248 \\ 239 \end{bmatrix}$$

Initial Guess: Assume an initial guess of

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}$$



# Example: Surface Shape Detection

---

Rewriting each equation

$$\begin{bmatrix} 0.2425 & 0 & -0.9701 \\ 0 & 0.2425 & -0.9701 \\ -0.2357 & -0.2357 & -0.9428 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 247 \\ 248 \\ 239 \end{bmatrix}$$

$$x_1 = \frac{247 - 0x_2 - (-0.9701)x_3}{0.2425}$$

$$x_2 = \frac{248 - 0x_1 - (-0.9701)x_3}{0.2425}$$

$$x_3 = \frac{239 - (-0.2357)x_1 - (-0.2357)x_2}{-0.9428}$$



# Example: Surface Shape Detection

---

Substituting initial guesses into the equations

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}$$

$$x_1 = \frac{247 - 0 \times 10 - (-0.9701) \times 10}{0.2425} = 1058.561$$

$$x_2 = \frac{248 - 0 \times 1058.561 - (-0.9701) \times 10}{0.2425} = 1062.685$$

$$x_3 = \frac{239 - (-0.2357) \times 1058.561 - (-0.2357) \times 1062.685}{-0.9428} = -783.8116$$



# Example: Surface Shape Detection

Finding the absolute relative approximate error

$$|\epsilon_a|_i = \left| \frac{X_i^{\text{new}} - X_i^{\text{old}}}{X_i^{\text{new}}} \right| \times 100$$

$$|\epsilon_a|_1 = \left| \frac{1058.561 - 10}{1058.561} \right| \times 100 = 99.0553\%$$

$$|\epsilon_a|_2 = \left| \frac{1062.685 - 10}{1062.685} \right| \times 100 = 99.0590\%$$

$$|\epsilon_a|_3 = \left| \frac{(-783.8116) - 10}{-783.8116} \right| \times 100 = 101.2758\%$$

At the end of the first iteration

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1058.561 \\ 1062.685 \\ -783.8116 \end{bmatrix}$$

The maximum absolute relative approximate error is 101.2758%

# Example: Surface Shape Detection

Iteration #2

Using 
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1058.561 \\ 1062.685 \\ -783.8116 \end{bmatrix}$$

$$x_1 = \frac{247 - 0 \times 1062.685 - (-0.9701) \times (-783.8116)}{0.2425} = -2117.013$$

$$x_2 = \frac{248 - 0 \times (-2117.013) - (-0.9701) \times (-783.8116)}{0.2425} = -2112.889$$

$$x_3 = \frac{239 - (-0.2357) \times (-2117.013) - (-0.2357) \times (-2112.889)}{-0.9428} = 803.9752$$



# Example: Surface Shape Detection

---

Finding the absolute relative approximate error for the second iteration

$$|\epsilon_a|_1 = \left| \frac{(-2117.013) - 1058.561}{-2117.013} \right| \times 100 = 150.0026\%$$

$$|\epsilon_a|_2 = \left| \frac{(-2112.889) - 1062.685}{-2112.889} \right| \times 100 = 150.2953\%$$

$$|\epsilon_a|_3 = \left| \frac{803.9752 - (-783.8116)}{803.9752} \right| \times 100 = 197.492\%$$



# Example: Surface Shape Detection

---

At the end of the second iteration

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2117.013 \\ -2112.889 \\ 803.9752 \end{bmatrix}$$

The maximum absolute  
relative approximate error is  
197.492%



# Example: Surface Shape Detection

Repeating more iterations, the following values are obtained

	$c_1$	$ \epsilon_a _1 \%$	$c_2$	$ \epsilon_a _2 \%$	$c_3$	$ \epsilon_a _3 \%$
1	1058.561	99.0556	1062.685	99.0590	-783.8116	101.2758
2	-2117.013	150.0023	-2112.889	150.2953	803.9752	197.492
3	4234.013	149.9910	4238.9128	149.8451	-2371.926	133.8955
4	-8470.124	149.9968	-8466.001	150.0698	3980.531	159.5882
5	16942.32	149.9939	16946.45	149.9574	-8725.692	145.6185
6	-33887.81	149.9953	-33883.69	150.0136	16689.37	152.2829

Notice: The absolute relative approximate errors are not decreasing



# Example: Surface Shape Detection

---

What went wrong?

Even though done correctly, the answer is not converging to the correct answer

This example illustrates a pitfall of the Gauss-Seidel method: not all systems of equations will converge.

Is there a fix?

One class of system of equations always converges: One with a *diagonally dominant* coefficient matrix.

Diagonally dominant:  $[A]$  in  $[A][X] = [C]$  is diagonally dominant if:

$$\left| a_{ii} \right| \geq \sum_{\substack{j=1 \\ j \neq i}}^n \left| a_{ij} \right| \quad \text{for all 'i'} \quad \text{and} \quad \left| a_{ii} \right| > \sum_{\substack{j=1 \\ j \neq i}}^n \left| a_{ij} \right| \quad \text{for at least one 'i'}$$



# Gauss-Seidel Method: Pitfall

---

Diagonally Dominant: In other words....

For every row: the element on the diagonal needs to be equal than or greater than the sum of the other elements of the coefficient matrix

For at least one row: The element on the diagonal needs to be greater than the sum of the elements.

What can be done? If the coefficient matrix is not originally diagonally dominant, the rows can be rearranged to make it diagonally dominant.



# Example: Surface Shape Detection

---

Examination of the coefficient matrix reveals that it is not diagonally dominant and cannot be rearranged to become diagonally dominant

$$\begin{bmatrix} 0.2425 & 0 & -0.9701 \\ 0 & 0.2425 & -0.9701 \\ -0.2357 & -0.2357 & -0.9428 \end{bmatrix}$$

This particular problem is an example of a system of linear equations that cannot be solved using the Gauss-Seidel method.

Other methods that would work:

1. Gaussian elimination

2. LU Decomposition



# Gauss-Seidel Method

---

## Summary

- Advantages of the Gauss-Seidel Method
- Algorithm for the Gauss-Seidel Method
- Pitfalls of the Gauss-Seidel Method



# Gauss-Seidel Method

---

Questions?