Gauss-Seidel Method

Jamie Trahan, Autar Kaw, Kevin Martin
University of South Florida
United States of America
kaw@eng.usf.edu

Introduction

This worksheet demonstrates the use of *Mathematica* to illustrate Gauss-Seidel method, an iterative technique used in solving a system of simultaneous linear equations.

Gauss-Seidel method is used to solve a set of simultaneous linear equations, [A][X]=[RHS], where $[A]_{nxn}$ is the square coefficient matrix, $[X]_{nx1}$ is the solution vector, and $[\text{RHS}]_{nx1}$ is the right hand side array. The equations can be rewritten as

$$x_i = \frac{(\text{rhs}_i - \sum_{j=1}^{n} a_{i,j} * x_j, i \neq j)}{a_{i,i}} \qquad \text{Equation (1.1)}$$

In certain cases, such as when a system of equations is large, iterative methods of solving equations such as Gauss-Seidel method are more advantageous. Elimination methods, such as Gaussian Elimination, are prone to round-off errors for a large set of equations whereas iterative methods, such as Gauss-Seidel method, allow the user to control round-off error. Also if the physics of the problem are well known, initial guesses needed in iterative methods can be made more judiciously for faster convergence.

The steps to apply Gauss-Seidel method are:
1) Make an initial guess for the solution vector [X]. This can be based on the physics of the problem.
2) All proper values are plugged into Equation (1.1). The new x_1 value that is calculated will replace the previous guess, x_1, in the solution vector. [X] will then be used to calculate x_2. This will be done for each x_i from x_1 to x_n until a new solution vector is complete. At this point, the first iteration is done.
3) The absolute relative approximate error is calculated by comparing each new guess x_i with the previous guess. The maximum of these errors is the absolute relative approximate error at the end of the iteration.
4) The new solution vector becomes the old solution vector and Steps 2-3 are repeated until either the maximum number of iterations have been conducted or the pre-specified tolerance has been met.

Complete details of how Equation (1.1) is derived as well as the pitfalls of the method can be found here.

Section 1: Input

The following are the input parameters to begin the simulation. The user can change those values that are highlighted and the worksheet will calculate an approximate solution to the system of equations.

- Number of equations, *n*

```
n = 4
```

```
4
```

- The *n*x*n* coefficient matrix [A]. Note that if the coefficient matrix is diagonally dominant**,** convergence is ensured. Otherwise, the solution may or may not converge.

```
A = Table[{{10, 3, 4, 5}, {2, 24, 2, 4}, {2, 2, 34, 3}, {2, 2, 2, 12}}]; A // MatrixForm
```

$$\begin{pmatrix} 10 & 3 & 4 & 5 \\ 2 & 24 & 2 & 4 \\ 2 & 2 & 34 & 3 \\ 2 & 2 & 2 & 12 \end{pmatrix}$$

- *n*x1 right hand side array

```
RHS = Table[{22, 32, 41, 18}]; RHS // MatrixForm
```

$$\begin{pmatrix} 22 \\ 32 \\ 41 \\ 18 \end{pmatrix}$$

- *n*x1 initial guess of the solution vector

```
X = Table[{1, 23, 4, 50}]; X // MatrixForm
```

$$\begin{pmatrix} 1 \\ 23 \\ 4 \\ 50 \end{pmatrix}$$

- Maximum number of iterations, *maxit*

```
maxit = 5
```

```
5
```

Section 2: Gauss-Seidel Iterations

Below, *maxit* iterations are conducted and values of the previous approximations, present approximations, absolute relative approximate error, and maximum absolute relative approximate error are calculated at the end of each iteration.

```
absea = Array[0, n];

Print["  "]
For[k = 1, k ≤ maxit, k++,
 Print["Iteration Number"];
 Print[k];
 Print["Previous iteration values of the solution vector"];
 Xold = X;
 Print[Xold];
 For[i = 1, i ≤ n, i++, summ = 0;
     For[j = 1, j ≤ n, j++,
        If[i ≠ j, summ = summ + A[[i, j]] * X[[j]]]];

     X[[i]] = N[(RHS[[i]] - summ)/A[[i, i]]];
 Maxabsea = 0.0;
 For[i = 1, i ≤ n, i++,
     absea[[i]] = Abs[(X[[i]] - Xold[[i]])/X[[i]]] * 100.0;
     If[absea[[i]] > Maxabsea, Maxabsea = absea[[i]]]];
 Print["New iterative values of the solution vector"];
 Print[X];
 Print["Absolute percentage relative approximate error"];
 Print[absea];
 Print["Maximum absolute percentage relative approximate error"];
 Print[Maxabsea];
 Print["  "]; Print["_____________________________________"]; Print["  "];]
```

```
General::spell :
 Possible spelling error: new symbol name "Xold" is similar to existing symbols {Fold, Hold}. More…

Iteration Number

1

Previous iteration values of the solution vector

{1, 23, 4, 50}

New iterative values of the solution vector

{-31.3, -4.725, -1.08676, 7.68529}

Absolute relative approximate percentage error

{103.195, 586.772, 468.065, 550.593}
```

Maximum absolute relative approximate percentage error

586.772

Iteration Number

2

Previous iteration values of the solution vector

{−31.3, −4.725, −1.08676, 7.68529}

New iterative values of the solution vector

{0.209559, 0.125551, 0.508056, 1.35947}

Absolute relative approximate percentage error

{15036.1, 3863.4, 313.907, 465.314}

Maximum absolute relative approximate percentage error

15036.1

Iteration Number

3

Previous iteration values of the solution vector

{0.209559, 0.125551, 0.508056, 1.35947}

New iterative values of the solution vector

{1.27938, 0.957802, 0.95433, 0.968082}

Absolute relative approximate percentage error

{83.6202, 86.8917, 46.7631, 40.4295}

Maximum absolute relative approximate percentage error

86.8917

Iteration Number

```
4

Previous iteration values of the solution vector

{1.27938, 0.957802, 0.95433, 0.968082}

New iterative values of the solution vector

{1.04689, 1.00522, 0.999751, 0.991357}

Absolute relative approximate percentage error

{22.2077, 4.71702, 4.54324, 2.34783}

Maximum absolute relative approximate percentage error

22.2077
```

```
Iteration Number

5

Previous iteration values of the solution vector

{1.04689, 1.00522, 0.999751, 0.991357}

New iterative values of the solution vector

{1.00286, 1.00122, 1.00052, 0.999233}

Absolute relative approximate percentage error

{4.39057, 0.399019, 0.0770952, 0.788184}

Maximum absolute relative approximate percentage error

4.39057
```

Section 3: Exact Answer

The last iteration above generates an approximate value with the given number of maximum iterations. The exact answer can be found by using *Mathematica's* built-in tools. The exact answer, last iterative solution vector, absolute relative true error (abset), maximum absolute relative true error (maxabset), and maximum absolute relative approximate error are given below.

```
B = LinearSolve[A, RHS];
abset = Array[0, n];
Maxabset = 0.0;
For[i = 1, i ≤ n, i++,
     abset[[i]] = Abs[(B[[i]] - X[[i]]) / B[[i]]] * 100.0;
       If[Maxabset ≤ abset[[i]], Maxabset = Abs[abset[[i]]]];
Print["Exact Answer"]
Print[B]
Print["  "]; Print["_____________________________________"]; Print["  "];
Print["Last Iterative Value"]
Print[X]
Print["  "]; Print["_____________________________________"]; Print["  "];
Print["Absolute percentage relative true error"]
Print[abset]
Print["  "]; Print["_____________________________________"]; Print["  "];
Print["Maximum absolute percentage relative true error"]
Print[Maxabset]
Print["  "]; Print["_____________________________________"]; Print["  "];
Print["Maximum absolute percentage relative approximate error"]
Print[Maxabsea]
```

```
General::spell1 :
 Possible spelling error: new symbol name "abset" is similar to existing symbol "absea". More…

General::spell1 :
 Possible spelling error: new symbol name "Maxabset" is similar to existing symbol "Maxabsea". More…
```

```
Exact Answer

{1, 1, 1, 1}

_____________________________________


Last Iterative Value

{1.00286, 1.00122, 1.00052, 0.999233}

_____________________________________


Absolute relative true error

{0.285531, 0.122322, 0.0522674, 0.0766869}

_____________________________________


Maximum absolute relative true error
```

```
0.285531
```

```
Maximum absolute relative approximate error

4.39057
```

Conclusion

Mathematica helped us apply our knowledge of Gauss-Seidel method to solve a system of *n* simultaneous linear equations.

Question 1: Change the coefficient matrix to one that is not diagonally dominant and see if Gauss-Seidel method converges.

Question 2: See if you can get a set of equations with a coefficient matrix that is not diagonally dominant to converge by Gauss-Seidel method.

References

[1]Autar Kaw, *Holistic Numerical Methods Institute, http://numericalmethods.eng.usf.edu/mws,* See
How does Gauss-Seidel method work?