

The Gauss-Seidel algorithm for solving resistive networks

In this paper we shall consider a numerical technique for solving networks

If we have a network that contains only generator and resistor the nodal or cut-set analysis will yield a set of linear simultaneous equations with constant coefficients

$$\begin{cases} I_1 = g_{11}V_1 + g_{12}V_2 + \dots g_{1n}V_n \\ I_2 = g_{21}V_1 + g_{22}V_2 + \dots g_{2n}V_n \\ \dots \\ I_n = g_{n1}V_1 + g_{n2}V_2 + \dots g_{nn}V_n \end{cases} \Leftrightarrow \begin{bmatrix} g_{11} & g_{12} & \dots & g_{1n} \\ g_{21} & g_{22} & \dots & g_{2n} \\ \dots & \dots & \dots & \dots \\ g_{n1} & g_{n2} & \dots & g_{nn} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ \dots \\ V_n \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \dots \\ I_n \end{bmatrix}$$

where g_{ij} are constant coefficients (conductance), V_j represent unknown voltage in a nodal and I_j represent the know quantities (current generator)

These simultaneous equations can be numerically solved by the iterative Gauss-Seidel algorithm. This algorithm can be implemented in a very straightforward way.

Starting from an initial guess of the vector solution \mathbf{V} (example $V_j = 0$ for $j = 1, 2, \dots, n$) we solve V_1 from the 1st equation and we update the vector solution with this new value.

Then we solve the V_2 from the 2nd equation, V_3 from the 3rd one, and so on.

Iterating, the vector \mathbf{V} converges to the solution of the linear system.

$$V_i^{(k)} = \frac{1}{g_{ii}} \left(\sum_{j=1}^{i-1} g_{ij} \cdot V_j^{(k)} + \sum_{j=i+1}^n g_{ij} \cdot V_j^{(k-1)} \right) \quad \begin{array}{l} \text{Where } \mathbf{V}^{(0)} = 0 \\ i = 1, 2, \dots, n \\ k = 1, 2, 3, \dots \end{array}$$

This method has several advantages

1. It is very easy to code
2. It is very robust
3. the error can be reduce till the smallest error machine
4. the iterations number is independent from the network dimension

First to discuss the convergence we have to say how to build the coefficients matrix

Indicating the conductance g_{ik} between the node i and k , the general diagonal g_{ii} element will given by the formula

$$g_{ii} = \sum_{\substack{k=0 \\ k \neq i}}^n g_{ik}$$

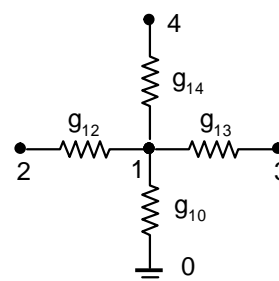
where the g_{i0} is the conductance between the node i and ground (node 0).

Example. For the network shown to the right the diagonal element g_{11} will be the sum of all conductances starting from the node 1. That is:

$$g_{11} = g_{10} + g_{12} + g_{13} + g_{14}$$

Note that if two nodes have no brunch, their mutual conductance will be zero. In the example:

$$g_{13} = g_{34} = g_{24} = 0$$

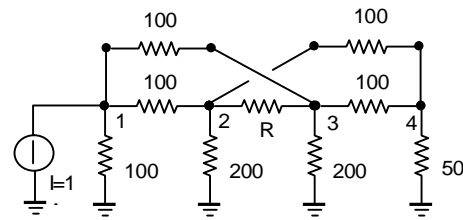


The conductance matrix of real networks has

$$g_{ii} \geq \sum_{\substack{k=1 \\ k \neq i}}^n g_{ik} \Rightarrow \sum_{\substack{k=0 \\ k \neq i}}^n g_{ik} \geq \sum_{\substack{k=1 \\ k \neq i}}^n g_{ik} \Rightarrow g_{i0} + \sum_{\substack{k=1 \\ k \neq i}}^n g_{ik} \geq \sum_{\substack{k=1 \\ k \neq i}}^n g_{ik}$$

Simplifying, we get the relation $g_{i0} \geq 0$ that is always true for real networks. At least, one node must have a finite conductance to the ground, so, at least, one relation is satisfied with the “>” symbols. This demonstrates that the matrix is diagonal dominant and thus the Gauss-Seidel algorithm is convergent. Let’s see how it works.

Assume to have the given network to analyse.
 There are 4 nodes and the unknown are the voltages V_1, V_2, V_3, V_4
 The resistors values are in ohm.
 The current generator flows into the node 1 a current of 1 A.
 The parameter $R = 100$ ohm



The linear system is $[G]V = I$, where

Conductance matrix G

0.03	-0.01	-0.01	0
-0.01	0.035	-0.01	-0.01
-0.01	-0.01	0.035	-0.01
0	-0.01	-0.01	0.04

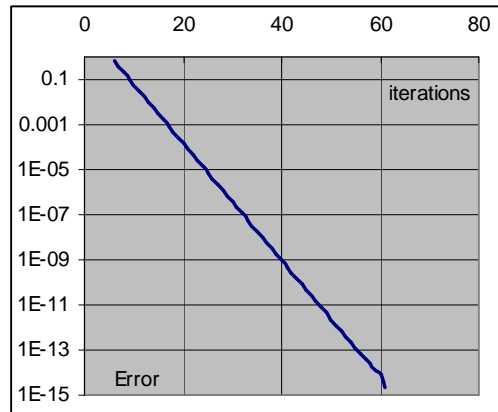
Current (I)

1
0
0
0

k	V1	V2	V3	V4
0	0	0	0	0
1	33.333	9.5238	12.245	5.4422
2	40.59	16.65	17.909	8.6399
3	44.853	20.401	21.113	10.378
4	47.171	22.475	22.864	11.335
5	48.446	23.613	23.827	11.86
6	49.147	24.238	24.356	12.148
7	49.531	24.581	24.646	12.307
8	49.743	24.77	24.806	12.394
9	49.859	24.874	24.893	12.442
10	49.922	24.931	24.941	12.468
11	49.957	24.962	24.968	12.482
12	49.977	24.979	24.982	12.49
13	49.987	24.989	24.99	12.495
14	49.993	24.994	24.995	12.497
15	49.996	24.997	24.997	12.498
16	49.998	24.998	24.998	12.499
17	49.999	24.999	24.999	12.5
18	49.999	24.999	25	12.5
19	50	25	25	12.5
20	50	25	25	12.5

Starting with the vector (0, 0, 0, 0), the algorithm converges quickly to the solution (50, 25, 25, 12.5)

After only 20 iterations the error is less than 0.1% and becomes less than 1E-14 after only 60 iteration



In the above example the convergence was very fast. But is this always true for every network? Unfortunately not. Let's repeat the calculus with the parameter $R = 1$ ohm, The solution vector does not change, but the convergence speed is much less than the previous one.

Conductance matrix G

0.03	-0.01	-0.01	0
-0.01	1.025	-1	-0.01
-0.01	-1	1.025	-0.01
0	-0.01	-0.01	0.04

Corrent (I)

1
0
0
0

k	V1	V2	V3	V4
0	0	0	0	0
1	33.333	0.3252	0.6425	0.2419
2	33.656	0.9575	1.2649	0.5556
3	34.074	1.5719	1.8714	0.8608
4	34.481	2.1705	2.4624	1.1582
5	34.878	2.7539	3.0383	1.4481
6	35.264	3.3224	3.5995	1.7305
7	35.641	3.8763	4.1464	2.0057
8	36.008	4.4161	4.6792	2.2738
9	36.365	4.9421	5.1985	2.5351
10	36.714	5.4546	5.7045	2.7898
11	37.053	5.9541	6.1976	3.0379
12	37.384	6.4408	6.678	3.2797
13	37.706	6.915	7.1462	3.5153
14	38.02	7.3772	7.6024	3.7449
15	38.327	7.8275	8.047	3.9686
16	38.625	8.2663	8.4802	4.1866
17	38.916	8.6939	8.9024	4.3991
18	39.199	9.1106	9.3137	4.6061
19	39.475	9.5166	9.7145	4.8078
20	39.744	9.9123	10.105	5.0043
...
300	49.9927	24.9893	24.9894	12.4947

Starting with the vector (0, 0, 0, 0), the algorithm converges very slowly to the solution (50, 25, 25, 12.5)

After 20 iterations the result is still very distance to the exact solution.

It needs more than 300 iterations to have an error less then 1%

Hill conditioned matrix

Why so many difference speed in the same algorithm? The reason is the matrix that in this example is still dominant but less than the first one. We note, in fact, that same element out of the diagonal is close in module to the diagonal element of the same row. That is:

$$|a_{23}| / |a_{22}| \cong 1 \text{ and } |a_{32}| / |a_{33}| \cong 1$$

On the contrary, for rapid convergence should be

$$|a_{23}| / |a_{22}| \ll 1 \text{ and } |a_{32}| / |a_{33}| \ll 1$$

This situation of poor convergence (hill-conditioned matrix) can be easy detected by direct inspection of the network itself. When the ratio between the highest and the lowest resistor is high then the associate matrix is hill conditioned. More high is the ratio; more slowly is the convergence feature.

In electric network is common encounter ratio of 1000 or more, and it would seem that this algorithm would be useless. Fortunately there is an efficient and cheap method for accelerating the convergence speed, called *delta square extrapolation* (Aitken's extrapolation)

Defining: $\Delta_i = x_i - x_{i-1}$

The extrapolation formula is:

$$x' = x_n - \frac{\Delta_n^2}{\Delta_n - \Delta_{n-1}} = x_n - \frac{(x_n - x_{n-1})^2}{x_n - 2x_{n-1} + x_{n-2}}$$

Where x' is the best approximation of the last three values x_n, x_{n-1}, x_{n-2}

In the last example, taking the last 18, 19, 20 iterations of V_1 we have

k	V1
18	39.198749596983
19	39.474758153645
20	39.743713756203

$$V_1' = V_1^{(20)} - \frac{(V_1^{(20)} - V_1^{(19)})^2}{V_1^{(20)} - 2 \cdot V_1^{(19)} + V_1^{(18)}}$$

extrap. =>	49.999999999979
------------	-----------------

The acceleration is superb! Using the only last three values of 20 iterations, with the extrapolation formula we have reached an error less than $1E-10$. This method can be applied also to V_2 , V_3 and V_4 , obtaining the same excellent acceleration.

The extrapolation formula works very fine for hill-conditional network. It is convenient to study it a bit deeper.

Here we will obtain a slight different form (but equivalent) for Aitken's extrapolation

We assume that the iteration formula has the following linear form (the simplest)

$$V_{k+1} = a + b \cdot V_k \quad (1)$$

Thus, starting from V_0 we have

$$V_1 = a + b \cdot V_0$$

$$V_2 = a + b \cdot V_1 = a + b \cdot (a + b \cdot V_0) = a + ab + b^2 \cdot V_0$$

$$V_3 = a + b \cdot V_2 = a + b \cdot (a + ab + b^2 \cdot V_0) = a + ab + ab^2 + b^3 \cdot V_0$$

....

$$V_n = a + b \cdot V_{n-1} = a + ab + ab^2 \dots + ab^{n-1} + b^n \cdot V_0 = a(1 + b + b^2 \dots + b^{n-1}) + b^n \cdot V_0$$

Rearranging, we get

$$V_n = a \cdot \sum_{k=0}^{n-1} b^k + b^n \cdot V_0 = a \cdot \left(\frac{1 - b^n}{1 - b} \right) + b^n \cdot V_0$$

In order that the iterations converge to a finite value it must be $|\beta| < 1$, thus:

$$\lim_{n \rightarrow \infty} b^n = 0$$

And then:

$$\lim_{n \rightarrow \infty} V_n = \lim_{n \rightarrow \infty} a \cdot \left(\frac{1 - b^n}{1 - b} \right) + b^n \cdot V_0 = \frac{a}{1 - b}$$

The solution is known if we know the coefficients α and β . They can be obtained from 3 sequential iterate value V_{n-2} , V_{n-1} , V_n by solving the following linear system

$$\begin{cases} V_{n-1} = a + b \cdot V_{n-2} \\ V_n = a + b \cdot V_{n-1} \end{cases} \Rightarrow b = \frac{V_n - V_{n-1}}{V_{n-1} - V_{n-2}}, \quad a = V_n - b \cdot V_{n-1}$$

Substituting, the solution V will be

$$V = \frac{a}{1-b} = \frac{V_n - b \cdot V_{n-1}}{1-b} \quad \text{where} \quad b = \frac{V_n - V_{n-1}}{V_{n-1} - V_{n-2}} \quad (2)$$

Eliminating β by substitution, we have

$$V = \frac{V_n \cdot V_{n-2} - V_{n-1}^2}{V_n - 2 \cdot V_{n-1} + V_{n-2}} = V_n - \frac{(V_n - V_{n-1})^2}{V_n - 2 \cdot V_{n-1} + V_{n-2}}$$

Thus the formula (2) is a different form for of the *delta square extrapolation*.

Obviously, because the initial assumption (1) is not exact, the value “ V ” will be an approximation of the exact solution. Repeating the formula every three (or more) iterations we get the solution with the highest approximation possible.

For hill-conditioned systems the parameter $\beta \cong 1$ or $|1 - \beta| < \epsilon$. (Usually ϵ is about $0.1 \div 0.01$) This is a cheap and practical test to detect when to improve the convergence speed by the extrapolation formula.

In the 1st example of fast convergence we have

$$\beta = (V^{(10)} - V^{(9)}) / (V^{(9)} - V^{(8)}) \cong 0.543 \Rightarrow 1 - \beta \cong 0.456$$

While in the 2nd example of slow convergence we have

$$\beta = (V^{(10)} - V^{(9)}) / (V^{(9)} - V^{(8)}) \cong 0.977 \Rightarrow 1 - \beta \cong 0.022$$

Note. The parameter β can be computed taking any three sequential iterates: in the example we have used the values (8, 9, 10); we would obtained a similar result taking the values (5, 6, 7) or (18, 19, 20)

Using the Gauss-Seidel algorithm with the extrapolation Aitken’s formula we can realized efficient methods for solving resistive network. For example we can adopt the following process flow.

