Thomas Algorithm (for Tridiagonal)

$$\begin{bmatrix} d_1 & u_1 & 0 & \bullet & 0 & 0 \\ l_2 & d_2 & u_2 & \bullet & 0 & 0 \\ 0 & l_3 & d_3 & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & l_{n-1} & d_{n-1} & u_{n-1} \\ 0 & 0 & 0 & 0 & l_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \bullet \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \bullet \\ b_{n-1} \\ b_n \end{bmatrix}$$

- No need to store $n^2 + n$ elements!
- Store only $4n$ elements in the form of four vectors l, d, u and b
- i^{th} equation is: $l_i x_{i-1} + d_i x_i + u_i x_{i+1} = b_i$
- Notice: $l_1 = u_n = 0$

Thomas Algorithm

- Initialize two new vectors α and β as $\alpha_1 = d_1$ and $\beta_1 = b_1$
- Take the first two equation and eliminate x_1 :

$$\begin{aligned} \alpha_1 x_1 + u_1 x_2 \qquad\qquad &= \beta_1 \\ l_2 x_1 + d_2 x_2 + u_2 x_3 &= b_2 \end{aligned}$$

- Resulting equation is: $\alpha_2 x_2 + u_2 x_3 = \beta_2$

where,

$$\alpha_2 = d_2 - \left(\frac{l_2}{\alpha_1}\right) u_1 \qquad \beta_2 = b_2 - \left(\frac{l_2}{\alpha_1}\right) \beta_1$$

- Similarly, we can eliminate x_2, x_3 ……

Thomas Algorithm

- At the i^{th} step: $\alpha_{i-1}x_{i-1} + u_{i-1}x_i = \beta_{i-1}$
  $l_i x_{i-1} + d_i x_i + u_i x_{i+1} = b_i$

- Eliminate $x_{i\text{-}1}$ to obtain: $\alpha_i x_i + u_i x_{i+1} = \beta_i$

where, $\alpha_i = d_i - \left(\dfrac{l_i}{\alpha_{i-1}}\right)u_{i-1}$ $\quad \beta_i = b_i - \left(\dfrac{l_i}{\alpha_{i-1}}\right)\beta_{i-1}$

- Last two equations are: $\alpha_{n-1}x_{n-1} + u_{n-1}x_n = \beta_{n-1}$
  $l_n x_{n-1} + d_n x_n = b_n$

- Eliminate $x_{n\text{-}1}$ to obtain: $\alpha_n x_n = \beta_n$

$$\alpha_n = d_n - \left(\frac{l_n}{\alpha_{n-1}}\right)u_{n-1} \qquad \beta_n = b_n - \left(\frac{l_n}{\alpha_{n-1}}\right)\beta_{n-1}$$

Thomas Algorithm

- Given: four vectors l, d, u and b
- Generate: two vectors α and β as

$$\alpha_1 = d_1 \text{ and } \beta_1 = b_1$$

$$\alpha_i = d_i - \left(\frac{l_i}{\alpha_{i-1}}\right) u_{i-1} \qquad \beta_i = b_i - \left(\frac{l_i}{\alpha_{i-1}}\right) \beta_{i-1}$$

$$i = 2, 3, \ldots\ldots n$$

- Solution: $x_n = \dfrac{\beta_n}{\alpha_n}$ $\quad x_i = \dfrac{\beta_i - u_i x_{i+1}}{\alpha_i}$

$$i = n\text{-}1 \ldots\ldots 3, 2, 1$$

- FP operations: $8(n\text{-}1) + 3(n\text{-}1) + 1 = 11n - 10$

Thomas Algorithm: Example

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{Bmatrix}$$

$$\alpha_1 = d_1; \alpha_i = d_i - \frac{l_i}{\alpha_{i-1}} u_{i-1}; \beta_1 = b_1; \beta_i = b_i - \frac{l_i}{\alpha_{i-1}} \beta_{i-1} \qquad x_n = \frac{\beta_n}{\alpha_n}; x_i = \frac{\beta_i - u_i x_{i+1}}{\alpha_i}$$

$$\alpha_1 = 2; \alpha_2 = d_2 - \frac{l_2}{\alpha_1} u_1 = 2 - \frac{-1}{2}(-1) = \frac{3}{2}; \alpha_3 = 2 - \frac{-1}{3/2}(-1) = \frac{4}{3}; \alpha_4 = 1 - \frac{-1}{4/3}(-1) = \frac{1}{4}$$

$$\beta_1 = 0; \beta_2 = b_2 - \frac{l_2}{\alpha_1} \beta_1 = 0 - \frac{-1}{2} 0 = 0; \beta_3 = 1 - \frac{-1}{3/2} 0 = 1; \beta_4 = 0 - \frac{-1}{4/3} 1 = \frac{3}{4}$$

$$x_4 = \frac{\beta_4}{\alpha_4} = 3; x_3 = \frac{\beta_3 - u_3 x_4}{\alpha_3} = \frac{1-(-1)3}{4/3} = 3; x_2 = \frac{0-(-1)3}{3/2} = 2; x_1 = \frac{0-(-1)2}{2} = 1$$

Iterative Methods

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \cdots\cdots\cdots + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \cdots\cdots\cdots + a_{2n}x_n = b_2 \\
&a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{3n}x_n = b_3 \\
&\bullet \quad \bullet \quad \bullet \qquad \bullet \quad \bullet \\
&a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{in}x_n = b_i \\
&\bullet \quad \bullet \quad \bullet \qquad \bullet \quad \bullet \\
&a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet + a_{nn}x_n = b_n
\end{aligned}
$$

- Assume (initialize) a solution vector $\boldsymbol{x}$
- Compute a new solution vector $\boldsymbol{x_{new}}$
- Iterate until $\| \boldsymbol{x} - \boldsymbol{x_{new}} \|_\infty \leq \varepsilon$
- We will learn two methods: *Jacobi* and *Gauss Seidel*

Jacobi and Gauss Seidel

- *Jacobi*: for the iteration index k ($k = 0$ for the initial guess)

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n$$

- *Gauss Seidel*: for the iteration index k ($k = 0$ for the initial guess)

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1,\ 2,\ \cdots\ n$$

Stopping Criteria

- Generate the error vector (***e***) at each iteration as

$$e_i^{(k+1)} = \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right|, \quad i = 1,\ 2,\ \cdots\ n$$

- Stop when: $\| \boldsymbol{e} \|_\infty \le \varepsilon$

Let us see an example?

Iterative Methods (Example)

Solve the following system of equations using Jacobi and Gauss Seidel methods and using initial guess as zero for all the variables with error less than 0.01%. Compare the number iterations required for solution using two methods:

$$x_1 + 2x_2 - x_4 = 1$$
$$x_2 + 2x_3 = 1.5$$
$$-x_3 + 2x_4 = 1.5$$
$$x_1 + 2x_3 - x_4 = 2$$

Jacobi

$$x_1^{(k+1)} = \frac{1 - 2x_2^{(k)} + x_4^{(k)}}{1} \qquad x_2^{(k+1)} = \frac{1.5 - 2x_3^{(k)}}{1}$$

$$x_3^{(k+1)} = \frac{1.5 - 2x_4^{(k)}}{-1} \qquad x_4^{(k+1)} = \frac{2 - x_1^{(k)} - 2x_3^{(k)}}{-1}$$

Gauss Seidel

$$x_1^{(k+1)} = \frac{1 - 2x_2^{(k)} + x_4^{(k)}}{1} \qquad x_2^{(k+1)} = \frac{1.5 - 2x_3^{(k)}}{1}$$

$$x_3^{(k+1)} = \frac{1.5 - 2x_4^{(k)}}{-1} \qquad x_4^{(k+1)} = \frac{2 - x_1^{(k+1)} - 2x_3^{(k+1)}}{-1}$$

Iterative methods (Example)

If you iterate, both the methods will diverge

Jacobi					
Iter	x1	x2	x3	x4	
0	0	0	0	0	
1	1	1.5	-1.5	-2	
2	-4	4.5	-5.5	-4	90.9
3	-12	12.5	-9.5	-17	76.5
4	-41	20.5	-35.5	-33	70.7
5	-73	72.5	-67.5	-114	71.1
6	-258	136.5	-229.5	-210	71.7
7	-482	460.5	-421.5	-719	70.8
8	-1639	844.5	-1439.5	-1327	70.6
9	-3015	2880.5	-2655.5	-4520	70.6
10	-10280	5312.5	-9041.5	-8328	70.7
11	-18952	18084.5	-16657.5	-28365	70.6
12	-64533	33316.5	-56731.5	-52269	70.6

Gauss Seidel					
Iter	x1	x2	x3	x4	
0	0	0	0	0	
1	1	1.5	-1.5	-4	
2	-6	4.5	-9.5	-27	85.2
3	-35	20.5	-55.5	-148	81.8
4	-188	112.5	-297.5	-785	81.1
5	-1009	596.5	-1571.5	-4154	81.1
6	-5346	3144.5	-8309.5	-21967	81.1
7	-28255	16620.5	-43935.5	-116128	81.1
8	-149368	87872.5	-232258	-613885	81.1
9	-789629	464516.5	-1227772	-3245174	81.1
10	-4174206	2455545	-6490350	-1.7E+07	81.1
11	-2.2E+07	12980701	-3.4E+07	-9.1E+07	81.1
12	-1.2E+08	68619633	-1.8E+08	-4.8E+08	81.1

Is the problem *ill conditioned*? The answer is NO!

Iterative methods (Example)

$$x_1 + 2x_2 - x_4 = 1$$
$$x_2 + 2x_3 = 1.5$$
$$-x_3 + 2x_4 = 1.5$$
$$x_1 + 2x_3 - x_4 = 2$$

Original Problem

A=	1	2	0	-1	***b***=	1
	0	1	2	0		1.5
	0	0	-1	2		1.5
	1	0	2	-1		2

Pivoting: Columns 2 to 1, 3 to 2, 4 to 3 and 1 to 4.

This is equivalent to change of variables:
x_1 (new) = x_2 (original)
x_2 (new) = x_3 (original)
x_3 (new) = x_4 (original)
x_4 (new) = x_1 (original)

After Pivoting

A=	2	0	-1	1	***b***=	1
	1	2	0	0		1.5
	0	-1	2	0		1.5
	0	2	-1	1		2

Iterative Methods (Example)

New Iteration Equations after pivoting (variable identifiers in the subscript are for the new renamed variables):

***A*=**	2	0	-1	1	***b*=**	1
	1	2	0	0		1.5
	0	-1	2	0		1.5
	0	2	-1	1		2

Jacobi

$$x_1^{(k+1)} = \frac{1 + x_3^{(k)} - x_4^{(k)}}{2} \qquad x_2^{(k+1)} = \frac{1.5 - x_1^{(k)}}{2}$$

$$x_3^{(k+1)} = \frac{1.5 + x_2^{(k)}}{2} \qquad x_4^{(k+1)} = \frac{2 - 2x_2^{(k)} + x_3^{(k)}}{1}$$

Gauss Seidel

$$x_1^{(k+1)} = \frac{1 + x_3^{(k)} - x_4^{(k)}}{2} \qquad x_2^{(k+1)} = \frac{1.5 - x_1^{(k+1)}}{2}$$

$$x_3^{(k+1)} = \frac{1.5 + x_2^{(k+1)}}{2} \qquad x_4^{(k+1)} = \frac{2 - 2x_2^{(k+1)} + x_3^{(k+1)}}{1}$$

Solution: Jacobi

Iter	x1	x2	x3	x4	||e||
0	0.000	0.000	0.000	0.000	
1	0.500	0.750	0.750	2.000	
2	-0.125	0.500	1.125	1.250	60.000
3	0.438	0.813	1.000	2.125	41.176
4	-0.063	0.531	1.156	1.375	54.545
5	0.391	0.781	1.016	2.094	34.328
6	-0.039	0.555	1.141	1.453	44.086
7	0.344	0.770	1.027	2.031	28.462
8	-0.002	0.578	1.135	1.488	36.483
9	0.323	0.751	1.039	1.979	24.778
10	0.030	0.588	1.125	1.537	28.717
11	0.294	0.735	1.044	1.949	21.123
12	0.048	0.603	1.117	1.574	23.771
13	0.271	0.726	1.051	1.912	17.637
14	0.070	0.614	1.113	1.599	19.537
15	0.257	0.715	1.057	1.885	15.143
16	0.086	0.622	1.108	1.627	15.827
17	0.240	0.707	1.061	1.864	12.734
18	0.098	0.630	1.103	1.647	13.200
19	0.228	0.701	1.065	1.844	10.664
20	0.111	0.636	1.100	1.663	10.858
21	0.219	0.695	1.068	1.829	9.054
22	0.120	0.641	1.097	1.679	8.940
23	0.209	0.690	1.070	1.816	7.568
24	0.127	0.645	1.095	1.690	7.458
25	0.203	0.686	1.073	1.804	6.344
26	0.134	0.649	1.093	1.700	6.157
27	0.197	0.683	1.074	1.796	5.344
28	0.139	0.652	1.091	1.708	5.110
29	0.192	0.680	1.076	1.788	4.458
30	0.144	0.654	1.090	1.715	4.260
31	0.188	0.678	1.077	1.782	3.736
32	0.148	0.656	1.089	1.721	3.529
33	0.184	0.676	1.078	1.777	3.131
34	0.151	0.658	1.088	1.726	2.940
35	0.181	0.675	1.079	1.772	2.612
36	0.153	0.659	1.087	1.730	2.449
37	0.179	0.673	1.080	1.768	2.186
38	0.156	0.661	1.087	1.733	2.035
39	0.177	0.672	1.080	1.765	1.827
40	0.157	0.662	1.086	1.736	1.697
41	0.175	0.671	1.081	1.763	1.525
42	0.159	0.662	1.086	1.738	1.413
43	0.174	0.671	1.081	1.761	1.275
44	0.160	0.663	1.085	1.740	1.177
45	0.173	0.670	1.082	1.759	1.064
46	0.161	0.664	1.085	1.742	0.982
47	0.172	0.669	1.082	1.757	0.888
48	0.162	0.664	1.085	1.743	0.818
49	0.171	0.669	1.082	1.756	0.742
50	0.163	0.665	1.084	1.744	0.682
51	0.170	0.669	1.082	1.755	0.619
52	0.164	0.665	1.084	1.745	0.569
53	0.169	0.668	1.082	1.754	0.516
54	0.164	0.665	1.084	1.746	0.474
55	0.169	0.668	1.083	1.754	0.431
56	0.165	0.665	1.084	1.747	0.395
57	0.169	0.668	1.083	1.753	0.360
58	0.165	0.666	1.084	1.747	0.330
59	0.168	0.668	1.083	1.753	0.300
60	0.165	0.666	1.084	1.748	0.275
61	0.168	0.667	1.083	1.752	0.250
62	0.165	0.666	1.084	1.748	0.229
63	0.168	0.667	1.083	1.752	0.209
64	0.166	0.666	1.084	1.748	0.191
65	0.168	0.667	1.083	1.751	0.174
66	0.166	0.666	1.084	1.749	0.159
67	0.167	0.667	1.083	1.751	0.145
68	0.166	0.666	1.084	1.749	0.133
69	0.167	0.667	1.083	1.751	0.121
70	0.166	0.666	1.084	1.749	0.111
71	0.167	0.667	1.083	1.751	0.101
72	0.166	0.666	1.083	1.749	0.093
73	0.167	0.667	1.083	1.751	0.084
74	0.166	0.666	1.083	1.749	0.077
75	0.167	0.667	1.083	1.751	0.070
76	0.166	0.666	1.083	1.749	0.064
77	0.167	0.667	1.083	1.750	0.059
78	0.166	0.667	1.083	1.750	0.054
79	0.167	0.667	1.083	1.750	0.049
80	0.166	0.667	1.083	1.750	0.045
81	0.167	0.667	1.083	1.750	0.041
82	0.166	0.667	1.083	1.750	0.037
83	0.167	0.667	1.083	1.750	0.034
84	0.166	0.667	1.083	1.750	0.031
85	0.167	0.667	1.083	1.750	0.028
86	0.167	0.667	1.083	1.750	0.026
87	0.167	0.667	1.083	1.750	0.024
88	0.167	0.667	1.083	1.750	0.022
89	0.167	0.667	1.083	1.750	0.020
90	0.167	0.667	1.083	1.750	0.018
91	0.167	0.667	1.083	1.750	0.017
92	0.167	0.667	1.083	1.750	0.015
93	0.167	0.667	1.083	1.750	0.014
94	0.167	0.667	1.083	1.750	0.013
95	0.167	0.667	1.083	1.750	0.011
96	0.167	0.667	1.083	1.750	0.010
97	0.167	0.667	1.083	1.750	0.010

Number of iteration required to achieve a relative error of < 0.01% = 97

Solution: Gauss Seidel

Iter	x1	x2	x3	x4	||e||
0	0.000	0.000	0.000	0.000	
1	0.500	0.500	1.000	2.000	
2	0.000	0.750	1.125	1.625	30.769
3	0.250	0.625	1.063	1.813	13.793
4	0.125	0.688	1.094	1.719	7.273
5	0.188	0.656	1.078	1.766	3.540
6	0.156	0.672	1.086	1.742	1.794
7	0.172	0.664	1.082	1.754	0.891
8	0.164	0.668	1.084	1.748	0.447
9	0.168	0.666	1.083	1.751	0.223
10	0.166	0.667	1.083	1.750	0.112
11	0.167	0.667	1.083	1.750	0.056
12	0.167	0.667	1.083	1.750	0.028
13	0.167	0.667	1.083	1.750	0.014
14	0.167	0.667	1.083	1.750	0.007

Number of iteration required to achieve a relative error of < 0.01% = 14

So, what makes the methods ***diverge***? When do we need ***pivoting*** or ***scaling*** or ***equilibration*** for the ***iterative methods***? Let's analyze for the ***convergence criteria***!

Iterative Methods

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & \cdot & \cdot & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ b_m \end{bmatrix}$$

How the iteration schemes look in the matrix form ?

Iterative Methods

$$\underset{\boldsymbol{A}}{\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ a_{31} & a_{32} & \cdot & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix}} = \underset{\boldsymbol{L}}{\begin{bmatrix} 0 & 0\cdots & \cdot & \cdot & \cdots\cdots 0 \\ a_{21} & 0 & 0 & \cdot & \cdots\cdot 0 \\ a_{31} & a_{32} & 0 & 0 & 0 \\ & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & a_{n3} & \cdot & 0 \end{bmatrix}} +$$

$$\underset{\boldsymbol{D}}{\begin{bmatrix} a_{11} & 0\cdots & 0 & \cdot & 0 \\ 0 & a_{22} & 0 & 0 & \cdot 0 \\ 0 & 0 & a_{33} & 0 & \cdot 0 \\ & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdots a_{nn} \end{bmatrix}} + \underset{\boldsymbol{U}}{\begin{bmatrix} 0 & a_{12} & a_{13} & \cdot & a_{1n} \\ 0 & 0 & a_{23} & \cdot & a_{2n} \\ 0 & 0 & 0 & \cdot & a_{3n} \\ & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & 0 \end{bmatrix}}$$

Iterative Methods

- $\boldsymbol{A = L + D + U}$
- $\boldsymbol{Ax = b}$ translates to $\boldsymbol{(L + D + U)x = b}$
- **Jacobi:** for an iteration counter k

$$Dx^{(k+1)} = -(U + L)x^{(k)} + b$$
$$x^{(k+1)} = -D^{-1}(U + L)x^{(k)} + D^{-1}b$$

- **Gauss Seidel:** for an iteration counter k

$$(L + D)x^{(k+1)} = -Ux^{(k)} + b$$
$$x^{(k+1)} = -(L + D)^{-1}Ux^{(k)} + (L + D)^{-1}b$$

Iterative Methods: Convergence

- All iterative methods: $x^{(k+1)} = Sx^{(k)} + c$
- *Jacobi*: $S = -D^{-1}(U + L) \qquad c = D^{-1}b$
- *Gauss Seidel*: $S = -(L + D)^{-1}U \qquad c = (L + D)^{-1}b$
- For true solution vector (x): $x = Sx + c$
- True error: $e^{(k)} = x - x^{(k)}$
- $e^{(k+1)} = Se^{(k)}$ or $e^{(k)} = S^k e^{(0)}$
- Methods will converge if: $\lim_{k \to \infty} e^{(k)} = 0$

$$\lim_{k \to \infty} S^k = 0$$

Iterative Methods: Convergence

- For the solution to exist, the matrix should have full rank (= n)
- The iteration matrix S will have n eigenvalues $\{\lambda_j\}_{j=1}^{n}$ and n independent eigenvectors $\{v_j\}_{j=1}^{n}$ that will form the basis for a n-dimensional vector space
- Initial error vector: $e^{(0)} = \sum_{j=1}^{n} C_j v_j$
- From the definition of eigenvalues: $e^{(k)} = \sum_{j=1}^{n} C_j \lambda_j^k v_j$
- Necessary condition: $\rho(S) < 1$
- Sufficient condition: $\|S\| < 1$ because $\rho(A) \leq \|A\|$

Iterative Methods: Convergence

- For the solution to exist, the matrix should have full rank (= n)
- The iteration matrix S will have n eigenvalues $\{\lambda_j\}_{j=1}^{n}$ and n independent eigenvectors $\{v_j\}_{j=1}^{n}$ that will form the basis for a n-dimensional vector space
- Initial error vector: $e^{(0)} = \sum_{j=1}^{n} C_j v_j$
- From the definition of eigenvalues: $e^{(k)} = \sum_{j=1}^{n} C_j \lambda_j^k v_j$
- Necessary condition: $\rho(S) < 1$
- Sufficient condition: $\|S\| < 1$ because $\rho(A) \leq \|A\|$

Iterative Methods: Convergence

Using the **definition of *S*** and using ***row-sum norm*** for matrix S, we obtain the following as the **sufficient condition for convergence** for both Jacobi and Gauss Seidel:

$$\left|a_{ii}\right| > \sum_{j=1,\, j\neq i}^{n} \left|a_{ij}\right|, \quad i = 1,2,\cdots n$$

If the original matrix is diagonally dominant, it will always converge!