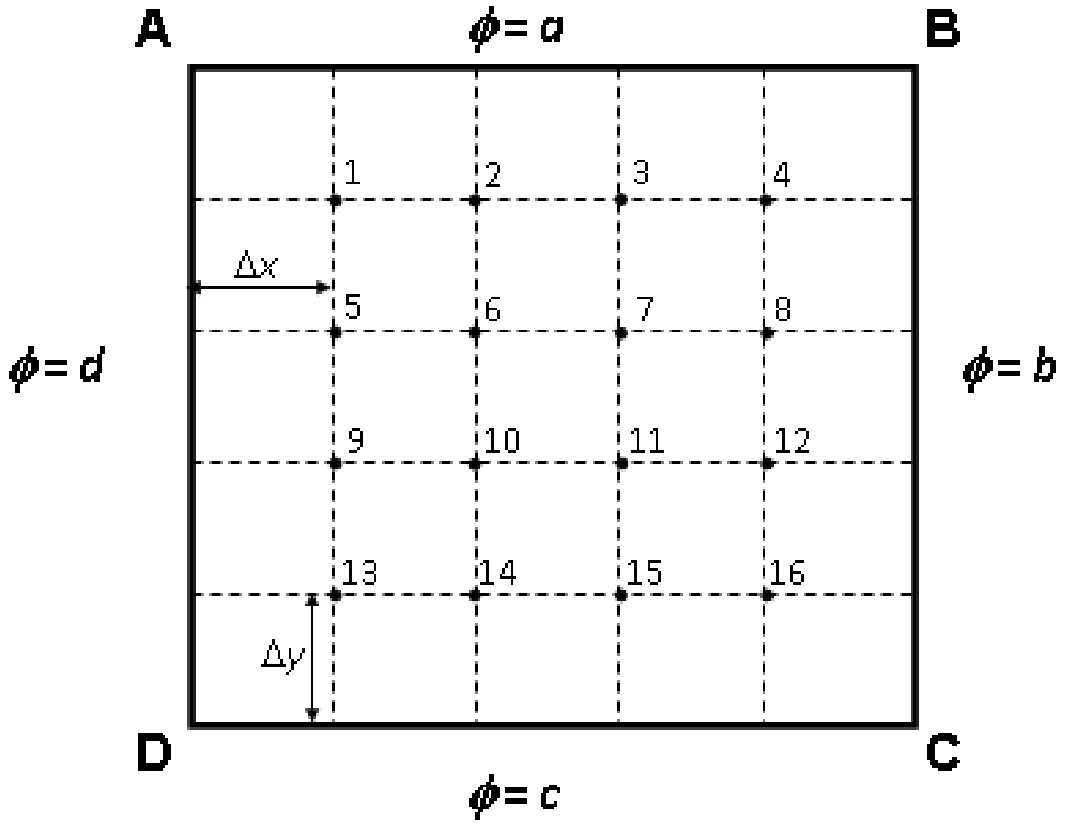
# ESO 208A: Computational Methods in Engineering

## Iterative Methods

*Saumyen Guha*

Department of Civil Engineering
IIT Kanpur

# Sparse Matrix: Origin



Laplace Equation:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

Boundary Conditions:

$$\phi(x,0) = c \qquad \phi(x, L_y) = a$$

$$\phi(0, y) = d \qquad \phi(L_x, y) = b$$

Also applicable for
Network Analysis

Size of the matrix 16×16
Total number of elements = 256
Number of non-zero elements = 56

## Sparse Matrix

$$\begin{bmatrix}
0 & \times & 0 & \times & 0 & 0 & \times & 0 & \cdots & 0 \\
0 & \times & 0 & 0 & \times & \times & 0 & 0 & \cdots & \times \\
\times & 0 & 0 & \times & 0 & \times & 0 & \times & \cdots & 0 \\
0 & \times & 0 & 0 & \times & 0 & 0 & 0 & \cdots & 0 \\
\times & 0 & \times & 0 & 0 & 0 & \times & 0 & \cdots & 0 \\
0 & \times & 0 & 0 & \times & 0 & 0 & 0 & \cdots & \times \\
0 & 0 & \times & 0 & 0 & \times & 0 & \times & \cdots & 0 \\
\times & 0 & 0 & \times & 0 & \times & 0 & \times & \cdots & \times \\
\times & 0 & \times & 0 & \times & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \times & 0 & 0 & \times & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \times & 0 & 0 & \times & 0 & 0 & 0 & \cdots & \times
\end{bmatrix}$$

Total number of elements = $n^2$

Number of non-zero elements ~ $O(n)$

*Direct methods algorithms such as Gauss elimination, Gauss Jordon, LU decomposition are inefficient for Banded and Sparse Matrices!*

# Iterative Methods

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

- 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

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

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

*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:

$$\begin{aligned} x_1 + 2x_2 - x_4 &= 1 \\ x_2 + 2x_3 &= 1.5 \\ -x_3 + 2x_4 &= 1.5 \\ x_1 + 2x_3 - x_4 &= 2 \end{aligned}$$

*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.

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 |

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)

# 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***!

## Questions?

- What are the condition of convergence for the iterative methods?

- Rate of convergence? Can we make them converge faster?

## Iterative Methods

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

*How the iteration schemes look in the matrix form ?*

# Iterative Methods

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

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

## Iterative Methods

- ***A* = *L* + *D* + *U***
- ***Ax* = *b*** translates to **(*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 $\rho(A) \leq \|A\|$

## Jacobi Convergence

$$\boldsymbol{S} = -\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})$$

$$s_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}} & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases}$$

If we use row-sum norm:

$$\|S\| = \max_{1 \le i \le n} \sum_{j=1}^{n} |s_{ij}| = \max_{1 \le i \le n} \sum_{j=1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right|$$

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

# 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:

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

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

## Rate of Convergence

For large *k*:

$$\frac{\|e^{(k+1)}\|}{\|e^{(k)}\|} \cong \rho(S)$$

or

$$\frac{\|e^{(k)}\|}{\|e^{(0)}\|} \cong \rho(S)^k$$

***Why?***

## Rate of Convergence

Number of iteration ($k$) required to decrease the initial error by a factor of $10^{-m}$ is then given by:

$$\frac{\left\|e^{(k)}\right\|}{\left\|e^{(0)}\right\|} \cong \rho(S)^k = 10^{-m}$$

or

$$k \geq -\frac{m}{\log_{10}\rho(S)} = \frac{m}{(-\log_{10}\rho(S))} = \frac{m}{R}$$

*R* is the *asymptotic rate of convergence* of the **iterative methods.**

# Improving Convergence

Recall Gauss Seidel:

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

Re-Write As:

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

$$x_i^{(k+1)} = x_i^{(k)} + d_i^{(k)}, \quad i = 1,\ 2,\ \cdots\ n$$

# Improving Convergence

Denoting: $\rho(S) = |\lambda_{\max}|$

$$e^{(k+1)} \cong |\lambda_{\max}| e^{(k)} \quad \text{or} \quad e^{(k+1)} - e^{(k)} \cong |\lambda_{\max}| (e^{(k)} - e^{(k-1)})$$

For any iterative method: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}^{(k)}$

$$d^{(k)} \cong \lambda_{\max} d^{(k-1)}$$

## Successive Over/Under Relaxation

$$x_i^{(k+1)} = x_i^{(k)} + \omega d_i^{(k)}, \quad i = 1,\ 2,\ \cdots n, \quad \omega > 0$$

$0 < \omega < 1$ : Under relaxation
$\omega = 1$ : Gauss Seidel
$1 < \omega < 2$ : Over Relaxation

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

## Solution of System Nonlinear Equations

## System of Non-linear Equations

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$$

$\boldsymbol{f}$ is now a vector of functions: $\boldsymbol{f} = \{f_1, f_2, \ldots f_n\}^T$

$\boldsymbol{x}$ is a vector of independent variables: $\boldsymbol{x} = \{x_1, x_2, \ldots x_n\}^T$

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ . \\ . \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

✓ Open methods: Fixed point, Newton-Raphson, Secant

# Open Methods: Fixed Point

- ✓ Rewrite the system as follows:

$$
\begin{aligned}
x_1 &= \phi_1(x_1, x_2, \ldots, x_n) \\
x_2 &= \phi_2(x_1, x_2, \ldots, x_n) \\
&. \\
&. \\
x_n &= \phi_n(x_1, x_2, \ldots, x_n)
\end{aligned}
$$

$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ is rewritten as $\boldsymbol{x} = \boldsymbol{\Phi}(\boldsymbol{x})$

- ✓ Initialize: assume $\boldsymbol{x}^{(0)}$
- ✓ Iteration Step $k$: $\boldsymbol{x}^{(k+1)} = \boldsymbol{\Phi}(\boldsymbol{x}^{(k)})$, initialize $\boldsymbol{x}^{(0)}$
- ✓ Stopping Criteria: $\dfrac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)}\|} \leq \varepsilon$

# Open Methods: Fixed Point

Condition for convergence:

For single variable: $|g'(\xi)| < 1$

For multiple variable, the derivative becomes the Jacobian matrix $\mathbb{J}$ whose elements are $J_{ij} = \frac{\partial \phi_i}{\partial x_j}$.

Example 2-variables: $\mathbb{J} = \begin{bmatrix} \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial x_2} \\ \frac{\partial \phi_2}{\partial x_1} & \frac{\partial \phi_2}{\partial x_2} \end{bmatrix}$

- ✓ Sufficient Condition: $\|\mathbb{J}\| < 1$
- ✓ Necessary Condition: Spectral Radius, $\rho(\mathbb{J}) < 1$

# Open Methods: Newton-Raphson

Example 2-variable: $f_1(x, y) = 0$ and $f_2(x, y) = 0$

2-d Taylor's series:

$$0 = f_1\left(x^{(k+1)}, y^{(k+1)}\right) = f_1\left(x^{(k)}, y^{(k)}\right) + \left(x^{(k+1)} - x^{(k)}\right) \left.\frac{\partial f_1}{\partial x}\right|_{\left(x^{(k)}, y^{(k)}\right)} + \left(y^{(k+1)} - y^{(k)}\right) \left.\frac{\partial f_1}{\partial y}\right|_{\left(x^{(k)}, y^{(k)}\right)} + HOT$$

$$0 = f_2\left(x^{(k+1)}, y^{(k+1)}\right) = f_2\left(x^{(k)}, y^{(k)}\right) + \left(x^{(k+1)} - x^{(k)}\right) \left.\frac{\partial f_2}{\partial x}\right|_{\left(x^{(k)}, y^{(k)}\right)} + \left(y^{(k+1)} - y^{(k)}\right) \left.\frac{\partial f_2}{\partial y}\right|_{\left(x^{(k)}, y^{(k)}\right)} + HOT$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}_{\left(x^{(k)}, y^{(k)}\right)} \begin{bmatrix} \left(x^{(k+1)} - x^{(k)}\right) \\ \left(y^{(k+1)} - y^{(k)}\right) \end{bmatrix} = \begin{bmatrix} -f_1\left(x^{(k)}, y^{(k)}\right) \\ -f_2\left(x^{(k)}, y^{(k)}\right) \end{bmatrix}$$

# Open Methods: Newton-Raphson

- ✓ Initialize: assume $\boldsymbol{x}^{(0)}$
- ✓ Recall single variable:

$$0 = f(x_{k+1}) = f(x_k) + (x_{k+1} - x_k)f'(x_k) + HOT$$

- ✓ Multiple Variables:

$$0 = \boldsymbol{f}\left(\boldsymbol{x}^{(k+1)}\right) = \boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right) + \left(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right)\mathbb{J}(x_k) + HOT$$

- ✓ Iteration Step *k*:

$$\Delta\boldsymbol{x}\mathbb{J}\left(\boldsymbol{x}^{(k)}\right) = -\boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right); \quad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}$$

- ✓ Stopping Criteria: $\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k)}\right\|} \le \varepsilon$

# Open Methods: Newton-Raphson

## Example 2-variable:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}_{\left(x_1^{(k)},x_2^{(k)}\right)} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -f_1\left(x_1^{(k)}, x_2^{(k)}\right) \\ -f_2\left(x_1^{(k)}, x_2^{(k)}\right) \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} - \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} = \begin{bmatrix} \left(x_1^{(k+1)} - x_1^{(k)}\right) \\ \left(x_2^{(k+1)} - x_2^{(k)}\right) \end{bmatrix}$$

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

## Open Methods: Secant

- ✓ Jacobian of the Newton-Raphson method is evaluated numerically using difference approximation.
- ✓ Numerical methods for estimation of derivative of a function will be covered in detail later.
- ✓ Rest of the method is same.

# ESO 208A: Computational Methods in Engineering

## Eigenvalues

*Saumyen Guha*

Department of Civil Engineering
IIT Kanpur

## Estimation of Eigenvalues

- ✓ Largest eigenvalue: Power Method
- ✓ Smallest eigenvalue: Inverse Power Method
- ✓ All Eigenvalues:
  - ✓ Inverse power method with shift
  - ✓ Faddeev-Leverrier method
  - ✓ QR Decomposition

## The Power Method

Works if:

There is a unique eigenvalue of maximum magnitude!

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots\ldots\ldots \geq |\lambda_n|$$

There are *n* independent eigenvectors corresponding to *n* eigenvalues

## The Power Method

Initialize an arbitrary non-zero vector ($z^{(0)}$) of length $n$ (same as that of the size of the square matrix)

Iteration Scheme: $z^{(k+1)} = Az^{(k)} = A^{k+1}z^{(0)}$

The vector ($z^{(k)}$) converges to the eigenvector corresponding to the eigenvalue of the maximum magnitude!

Why? How?

# Proof of Convergence: Power Method

Denote $x_i$ as the eigenvector corresponding to the eigenvalue $\lambda_i$

$$z^{(0)} = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$$

$\alpha_i$'s are the constants

Since, $Ax_j = \lambda_j x_j$

$$z^{(k)} = A^k z^{(0)} = A^k \sum_{i=1}^{n} \alpha_i x_i = \sum_{i=1}^{n} \lambda_i^k \alpha_i x_i = \lambda_1^k \left( \alpha_1 x_1 + \sum_{j=2}^{n} \left( \frac{\lambda_j}{\lambda_1} \right)^k \alpha_j x_j \right)$$

For Large $k$: $z^{(k)} \approx \lambda_1^k \alpha_1 x_1$

# The Power Method

$\lambda_1^k$ may become very large as the iteration progresses!

It’s a good idea to normalize $z^{(k)}$ at every iteration using a vector norm (e.g., $L_2$ norm)

$$y^{(k)} = \frac{z^{(k)}}{\left\| z^{(k)} \right\|_2} \qquad\qquad z^{(k+1)} = Ay^{(k)}$$

Once the iterations converged (max. norm of the error less than tolerance), use *Raleigh Quotient* to compute the eigenvalue:

$$\lambda_i = \frac{x_i^T A x_i}{x_i^T x_i}$$

# Power Method Algorithm

Initialize an arbitrary non-zero vector $z^{(0)}$

Iterate: $y^{(k)} = \dfrac{z^{(k)}}{\left\| z^{(k)} \right\|_2}$ $\quad z^{(k+1)} = Ay^{(k)}$

Stop when: $\left\| y^{(k+1)} - y^{(k)} \right\|_\infty \le \epsilon$

Compute the largest eigenvalue as:

$$\lambda_1^{(k)} = \frac{y^{T(k)} A y^{(k)}}{y^{T(k)} y^{(k)}} = y^{T(k)} z^{(k+1)} \text{ since } y^{T(k)} y^{(k)} = 1$$

## Power Method: Example

$$A = \begin{matrix} 3 & 4 & 1 \\ 3 & 5 & 1 \\ 2 & 2 & 1 \end{matrix}$$

| **y0** | **z = Ay** | **y = z/\|\|z\|\|** | **z = Ay** |
|---|---|---|---|
| 1 | 3 | 0.639602149 | 4.903616476 |
| 0 | 3 | 0.639602149 | 5.543218625 |
| 0 | 2 | 0.426401433 | 2.984810029 |
| | | | |
| **Lambda** | 3 | **Lambda** | 7.954545455 |
| **Error (%)** | | **Error (%)** | 62.28571429 |

# Power Method: Example

| **y = z/\|\|z\|\|** | **z = Ay** | **y = z/\|\|z\|\|** | **z = Ay** |
|---|---|---|---|
| 0.614481438 | 4.996001256 | 0.613586545 | 5.003847489 |
| 0.694631191 | 5.690632447 | 0.698898043 | 5.702745532 |
| 0.37403218 | 2.992257437 | 0.367495684 | 2.992464859 |
| | | | |
| **Lambda** | 8.142041399 | **Lambda** | 8.155649103 |
| **Error (%)** | 2.302812468 | **Error (%)** | 0.166850045 |

| **y = z/\|\|z\|\|** | **z = Ay** | **y = z/\|\|z\|\|** | **z = Ay** |
|---|---|---|---|
| 0.613543606 | 5.00450421 | 0.613540845 | 5.004557825 |
| 0.699238549 | 5.703742759 | 0.699265903 | 5.703823728 |
| 0.366919193 | 2.992483504 | 0.366871678 | 2.992485174 |
| | | | |
| **Lambda** | 8.156758006 | **Lambda** | 8.156848145 |
| **Error (%)** | 0.013594899 | **Error (%)** | 0.001105072 |

# The Inverse Power Method

- ✓ Apply power method on matrix $\boldsymbol{A}^{-1}$ to obtain the largest eigenvalue. Inverse of this eigenvalue is the smallest eigenvalue of $\boldsymbol{A}$
- ✓ Proposition: Inverse of the largest eigenvalue of $\boldsymbol{A}^{-1}$ is the smallest eigenvalue of $\boldsymbol{A}$
  - ✓ For any eigenvalue $\lambda_i$ and corresponding eigenvector $\boldsymbol{x_i}$ of matrix $\boldsymbol{A}$: $\boldsymbol{Ax_i} = \lambda_i \boldsymbol{x_i}$
  - ✓ Since $\lambda_i$ is a scalar, $(1/\lambda_i)\boldsymbol{x_i} = \boldsymbol{A}^{-1}\boldsymbol{x_i}$
  - ✓ If $\lambda_i$ is an eigenvalue of matrix $\boldsymbol{A}$, $(1/\lambda_i)$ is an eigenvalue of $\boldsymbol{A}^{-1}$ with the same corresponding eigenvector $\boldsymbol{x_i}$
  - ✓ Inverse of the smallest eigenvalue of matrix $\boldsymbol{A}$, is the largest eigenvalue of $\boldsymbol{A}^{-1}$

# The Inverse Power Method

$\lambda_1^k$ may become very large as the iteration progresses!

It's a good idea to normalize $z^{(k)}$ at every iteration using a vector norm (e.g., $L_2$ norm)

$$y^{(k)} = \frac{z^{(k)}}{\left\|z^{(k)}\right\|_2} \qquad z^{(k+1)} = A^{-1}y^{(k)}$$

Compute the largest eigenvalue as:

$$\lambda_1^{(k)} = \frac{y^{T(k)}Ay^{(k)}}{y^{T(k)}y^{(k)}} = y^{T(k)}z^{(k+1)}$$

**What is the smallest eigenvalue of *A* ?**

# Inverse Power Method with Shift

- ✓ Apply power method on matrix $(\boldsymbol{A} - \alpha \boldsymbol{I})^{-1}$ to obtain the eigenvalue that is closest to constant $\alpha$.
- ✓ Proposition: Inverse of the largest eigenvalue of $\boldsymbol{A}^{-1}$ is the smallest eigenvalue of $\boldsymbol{A}$
  - ✓ For any eigenvalue $\lambda_i$ and corresponding eigenvector $\boldsymbol{x_i}$ of matrix $\boldsymbol{A}$: $\boldsymbol{A x_i} = \lambda_i \boldsymbol{x_i}$
  - ✓ For a scalar constant $\alpha$, $\boldsymbol{A x_i} - \alpha \boldsymbol{x_i} = \lambda_i \boldsymbol{x_i} - \alpha \boldsymbol{x_i}$
  - ✓ $(\boldsymbol{A} - \alpha \boldsymbol{I})\boldsymbol{x_i} = (\lambda_i - \alpha)\boldsymbol{x_i}$, $(\lambda_i - \alpha)$ is an eigenvalue of the matrix $(\boldsymbol{A} - \alpha \boldsymbol{I})$.
  - ✓ Smallest eigenvalue of matrix $(\boldsymbol{A} - \alpha \boldsymbol{I})$ corresponds to that eigenvalue of matrix $\boldsymbol{A}$ that is closest in magnitude to the scalar constant $\alpha$
- ✓ This gives an algorithm to estimate an eigenvalue of a matrix that is closest to a given constant!

# The Inverse Power Method with Shift

$\lambda_1^k$ may become very large as the iteration progresses!

It’s a good idea to normalize $z^{(k)}$ at every iteration using a vector norm (e.g., $L_2$ norm)

$$y^{(k)} = \frac{z^{(k)}}{\left\|z^{(k)}\right\|_2} \qquad z^{(k+1)} = (A - \alpha I)^{-1} y^{(k)}$$

Compute the largest eigenvalue as:

$$\lambda_1^{(k)} = \frac{y^{T(k)} A y^{(k)}}{y^{T(k)} y^{(k)}} = y^{T(k)} z^{(k+1)}$$

**What is the eigenvalue closest to $\alpha$?**