10.34: Numerical Methods Applied to Chemical Engineering

Lecture 6:
Singular value decomposition
Iterative solutions of linear equations

Recap

- Eigenvalues
- Eigenvectors
- Eigendecomposition

Recap

- Find the eigenvalues and eigenfunctions of: $\frac{d^2}{dx^2}$

$$\frac{d^2}{dx^2}y = \lambda y, \quad y(0) = 0, y(L) = 0$$

Recap

- Find the eigenvalues and eigenfunctions of: $\frac{d^2}{dx^2}$

$$\frac{d^2}{dx^2}y = \lambda y, \quad y(0) = 0, y(L) = 0$$

$$y = C_1 e^{\sqrt{\lambda}x} + C_2 e^{-\sqrt{\lambda}x}$$

$$y = C_1' \cos(\sqrt{-\lambda}x) + C_2' \sin(\sqrt{-\lambda}x)$$

$$y(0) = 0 \Rightarrow C_1' = 0$$

$$y(L) = 0 \Rightarrow \sqrt{-\lambda} = \frac{2\pi n}{L}, n \in \mathbb{Z}$$

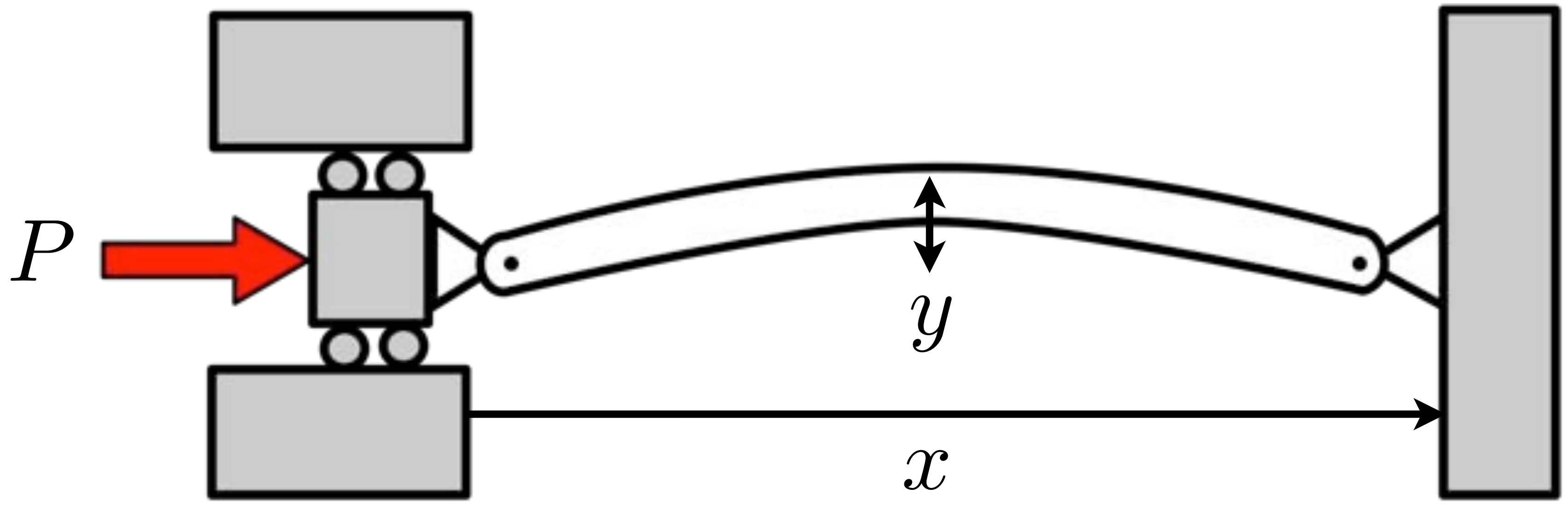
$$\lambda_n = -\left(\frac{2\pi n}{L}\right)^2 \qquad y_n = C \sin\left(\frac{2\pi n}{L}x\right)$$

Recap

- Energy balance for an elastic column:

$$EI\frac{d^2y}{dx^2} + Py = 0$$

  - Beyond what value of the pressure, P, will an elastic column buckle?



Singular Value Decomposition

- Is there an "eigendecomposition" for non-square matrices? Yes!
  - For: $\mathbf{A} \in \mathbb{R}^{N\times M}$
    - $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger$
    - with: $\mathbf{U} \in \mathbb{C}^{N\times N} \quad \mathbf{\Sigma} \in \mathbb{R}^{N\times M} \quad \mathbf{V} \in \mathbb{C}^{M\times M}$
    - and $\mathbf{V}^\dagger = \bar{\mathbf{V}}^T$
  - $\mathbf{\Sigma}$ has only diagonal elements which are positive:

$$\mathbf{\Sigma} = \begin{pmatrix} \Sigma_{11} & 0 & 0 \\ 0 & \Sigma_{22} & 0 \\ 0 & 0 & \ddots \end{pmatrix}$$

  - $\mathbf{U}$ and $\mathbf{V}$ are called the left and right singular vectors.

Singular Value Decomposition

- Properties of the singular value decomposition:
  - $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices
    - $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}, \quad \mathbf{V}\mathbf{V}^\dagger = \mathbf{I}$
  - $\mathbf{A}^\dagger\mathbf{A} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger)^\dagger\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{\Sigma}\mathbf{V}^\dagger$
    - $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^\dagger\mathbf{A}$
    - Σ_{ii}^2 are the eigenvalues of $\mathbf{A}^\dagger\mathbf{A}$
  - $\mathbf{A}\mathbf{A}^\dagger = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger)^\dagger = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^\dagger\mathbf{U}^\dagger$
    - $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^\dagger$
    - Σ_{ii}^2 are the eigenvalues of $\mathbf{A}\mathbf{A}^\dagger$
  - Σ_{ii} are called the singular values of $\mathbf{A}$.

Singular Value Decomposition

- Properties of the singular value decomposition: $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\dagger$
  - Some columns of $\boldsymbol{\Sigma}$ are zero. The columns of $\mathbf{V}$ corresponding to these span $\mathcal{N}(\mathbf{A})$
  - Some columns of $\boldsymbol{\Sigma}$ are non-zero. The rows of $\mathbf{U}$ corresponding to these span $\mathcal{R}(\mathbf{A})$
  - Example:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

```
[U,S,V] = svd(A)
```

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Singular Value Decomposition

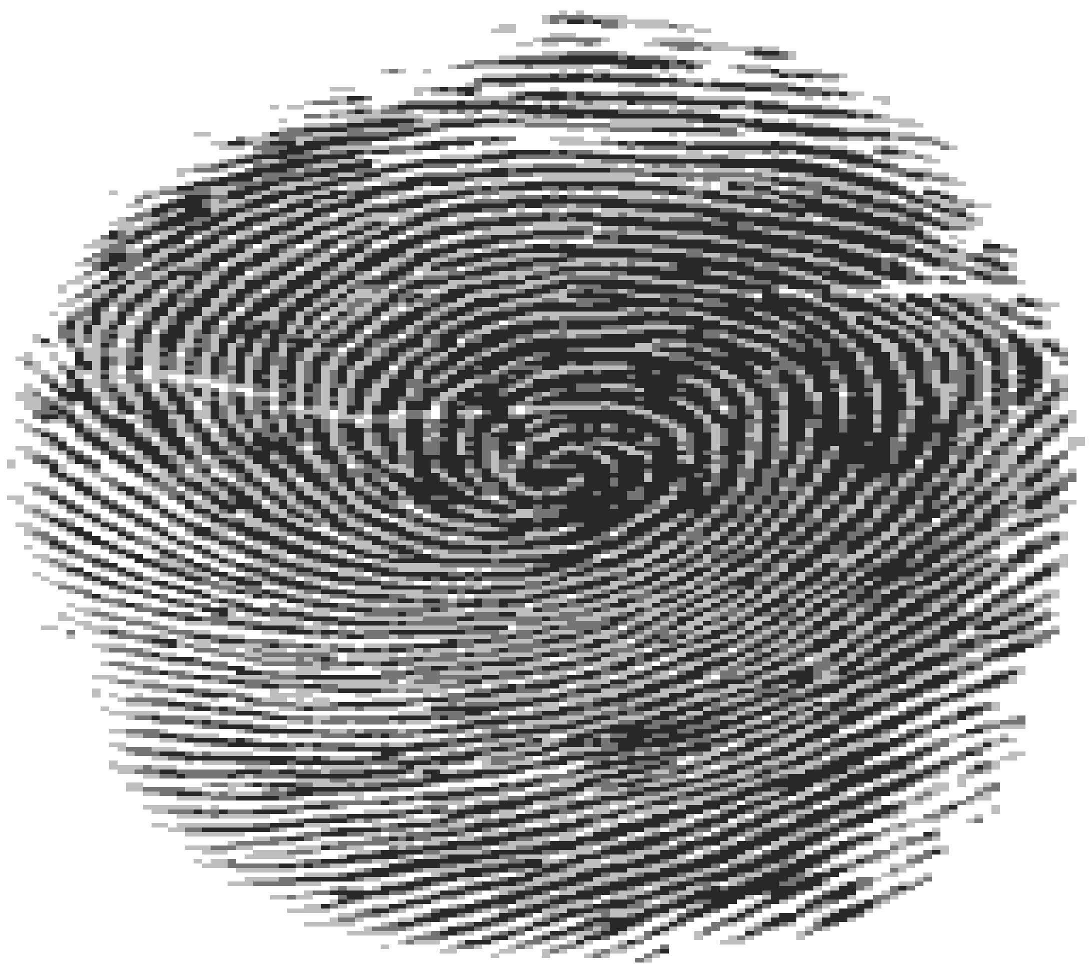
- Example:

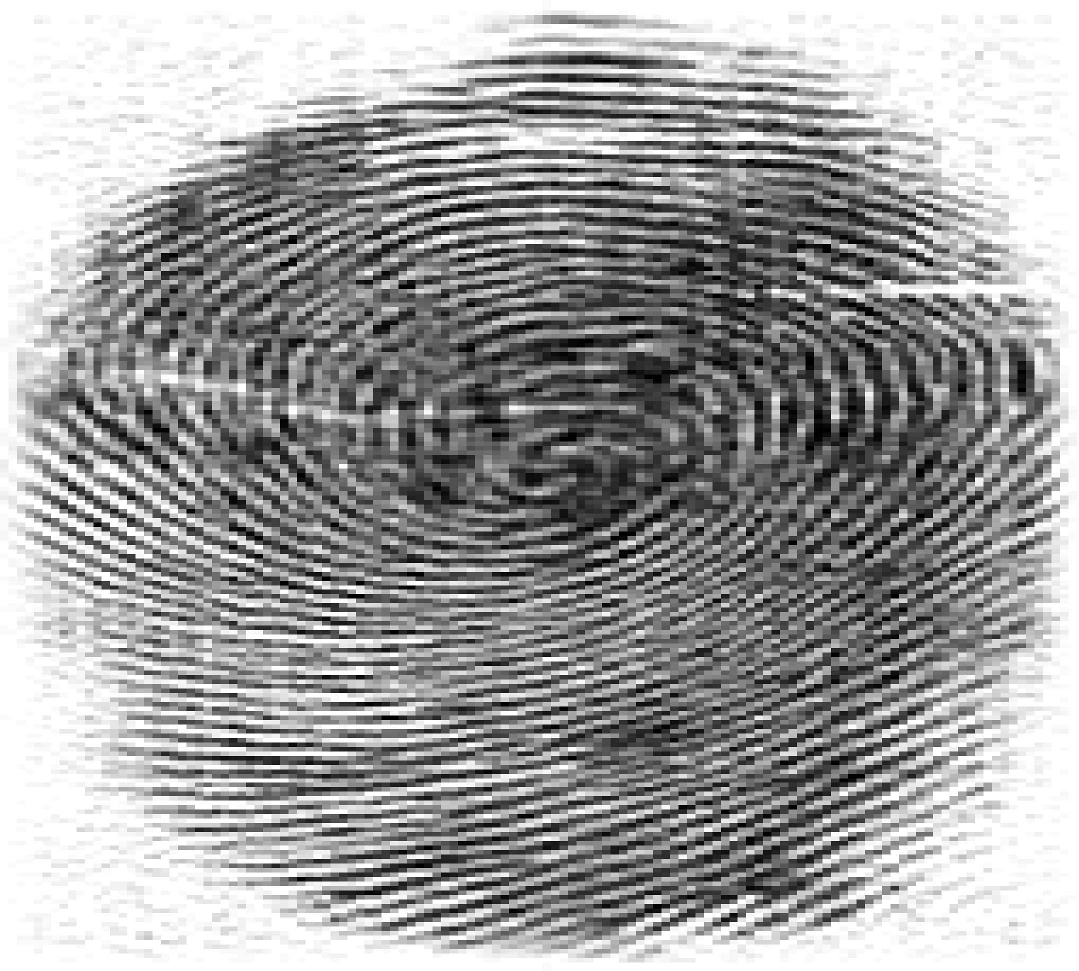
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Singular Value Decomposition

- How is singular value decomposition used?





- Example: data compression/matrix approximation
  - Left: original bitmap
  - Right: compressed bitmap retaining only 50 biggest singular values. All other set equal to zero.

Singular Value Decomposition

- How is singular value decomposition used?
  - Least squares solution to: $\mathbf{A}\mathbf{x} = \mathbf{b}$
    - with $\mathbf{A} \in \mathbb{R}^{N \times M}$ $\mathbf{x} \in \mathbb{R}^{M}$ $\mathbf{b} \in \mathbb{R}^{N}$
    - Least squares means find the vector $\mathbf{x}$ that minimizes: $\phi(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$
      - where $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{U}\left(\mathbf{\Sigma}\mathbf{V}^\dagger\mathbf{x} - \mathbf{U}^\dagger\mathbf{b}\right)$
    - Let $\mathbf{y} = \mathbf{V}^\dagger\mathbf{x}$ and $\mathbf{p} = \mathbf{U}^\dagger\mathbf{b}$
      - then $\phi(\mathbf{x}) = \|\mathbf{U}(\mathbf{\Sigma}\mathbf{y} - \mathbf{p})\|_2^2 = \|(\mathbf{\Sigma}\mathbf{y} - \mathbf{p})\|_2^2$
    - Let r be the number of non-zero singular values (also the rank of $\mathbf{A}$):
      - then $\phi(\mathbf{x}) = \sum_{i=1}^{r} |\Sigma_{ii} y_i - p_i|^2 + \sum_{i=r+1}^{N} |p_i|^2$

Singular Value Decomposition

- How is singular value decomposition used?
  - Least squares solution to: $\mathbf{Ax} = \mathbf{b}$
    - with $\mathbf{A} \in \mathbb{R}^{N \times M}$ $\mathbf{x} \in \mathbb{R}^M$ $\mathbf{b} \in \mathbb{R}^N$
    - and $\mathbf{y} = \mathbf{V}^\dagger \mathbf{x}$ $\mathbf{p} = \mathbf{U}^\dagger \mathbf{b}$
    - Minimizes:

$$\phi(\mathbf{x}) = \sum_{i=1}^{r} |\Sigma_{ii} y_i - p_i|^2 + \sum_{i=r+1}^{N} |p_i|^2$$

- Therefore, $y_i = \frac{p_i}{\Sigma_{ii}}$ for $1 \leq i \leq r$
- What about y_i for $r+1 \leq i \leq M$?
  - Least squares system is underdetermined
  - Just set: $y_i = 0$ for the rest and find $\mathbf{x} = \mathbf{Vy}$

Iterative Solutions to Lin. Eqns.

- Gaussian elimination or eigenvalue decomposition require $O(N^3)$ operations to complete.
- For many problems of practical interest (solutions to PDEs in particular) N can be so large that these calculations are infeasible.
- An alternative approach seeking approximate solutions to linear equations is more commonly employed.
- These algorithms are based on iterative refinement of an initial guess.
  - For: $\mathbf{A}\mathbf{x} = \mathbf{b}$
  - An iterative map might look like: $\mathbf{x}_{i+1} = \mathbf{C}\mathbf{x}_i + \mathbf{c}$
  - The map is converged when: $\mathbf{x}_{i+1} = \mathbf{x}_i$
  - The converged $\mathbf{x}_i$ is a solution if:

$$\mathbf{x}_i = (\mathbf{I} - \mathbf{C})^{-1}\mathbf{c} = \mathbf{A}^{-1}\mathbf{b}$$

Iterative Solutions to Lin. Eqns.

- Example: solve iteratively

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- split: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

- rename: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{i+1} = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} \mathbf{x}_i + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

- iterate: $\mathbf{x}_{i+1} = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} \mathbf{x}_i + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Jacobi Iteration

- For: $\mathbf{A}\mathbf{x} = \mathbf{b}$
  - Split $\mathbf{A}$ into $\mathbf{D} + \mathbf{R}$
    - $\mathbf{D}$ is the diagonal elements of $\mathbf{A}$
    - $\mathbf{R}$ is the off-diagonal elements of $\mathbf{A}$
- Rewrite the equations as an iterative map:
  - $\mathbf{D}\mathbf{x}_{i+1} = -\mathbf{R}\mathbf{x}_i + \mathbf{b}$
  - or $\mathbf{x}_{i+1} = \mathbf{D}^{-1}\left(-\mathbf{R}\mathbf{x}_i + \mathbf{b}\right)$
- If the iterations converge, then $(\mathbf{D} + \mathbf{R})\mathbf{x}_i = \mathbf{b}$
  - We have found the solution (if map converges)!
- Jacobi iteration transforms a hard problem, $\mathbf{A}^{-1}\mathbf{b}$, into a succession of easy problems, $\mathbf{D}^{-1}\mathbf{c}$

Jacobi Iteration

- For: $\mathbf{A}\mathbf{x} = \mathbf{b}$
  - Split $\mathbf{A}$ into $\mathbf{D} + \mathbf{R}$
    - $\mathbf{D}$ is the diagonal elements of $\mathbf{A}$
    - $\mathbf{R}$ is the off-diagonal elements of $\mathbf{A}$
- Rewrite the equations as an iterative map:
  - $\mathbf{x}_{i+1} = \mathbf{D}^{\quad 1}(-\mathbf{R}\mathbf{x}_i + \mathbf{b})$
- Does Jacobi converge to the right solution $\mathbf{x}$?
  - Substitute: $\mathbf{b} = \mathbf{A}\mathbf{x}$
  - Then: $\mathbf{x}_{i+1} - \mathbf{x} = -\mathbf{D}^{-1}\mathbf{R}(\mathbf{x}_i - \mathbf{x})$
  - Take the norm of both sides: $\dfrac{\|\mathbf{x}_{i+1} - \mathbf{x}\|_p}{\|\mathbf{x}_i - \mathbf{x}\|_p} \leq \|\mathbf{D}^{-1}\mathbf{R}\|_p$

Jacobi Iteration

- The ratio of absolute error in successive iterates is:

$$\frac{\|\mathbf{x}_{i+1} - \mathbf{x}\|_p}{\|\mathbf{x}_i - \mathbf{x}\|_p} \leq \|\mathbf{D}^{-1}\mathbf{R}\|_p$$

  - If this is less than one, the error gets smaller after each iteration. The iterative map converges!
  - When is $\|\mathbf{D}^{-1}\mathbf{R}\|_p < 1$?
    - Consider the ∞-norm of a matrix which gives the maximum row sum:

$$\|\mathbf{D}^{-1}\mathbf{R}\|_\infty = \max_i \sum_{j\neq i} |A_{ii}^{-1} A_{ij}|$$

    - $\|\mathbf{D}^{-1}\mathbf{R}\|_\infty < 1$ when $|A_{ii}| > \sum_{j\neq i} |A_{ij}|$
    - $\mathbf{A}$ is “diagonally dominant”

Gauss-Seidel Iteration

- For: $\mathbf{A}\mathbf{x} = \mathbf{b}$
  - Split $\mathbf{A}$ into $\mathbf{L} + \mathbf{U}$
    - $\mathbf{L}$ is the lower triangular elements of $\mathbf{A}$
    - $\mathbf{U}$ is the upper triangular elements (no diagonal)
- Rewrite the equations as an iterative map:
  - $\mathbf{L}\mathbf{x}_{i+1} = -\mathbf{U}\mathbf{x}_i + \mathbf{b}$
  - or $\mathbf{x}_{i+1} = \mathbf{L}^{-1}(-\mathbf{U}\mathbf{x}_i + \mathbf{b})$
- Again, successive calculations of $\mathbf{L}^{-1}\mathbf{c}$ are easier than $\mathbf{A}^{-1}\mathbf{b}$
- Does Gauss-Seidel converge? Yes if, $\|\mathbf{L}^{-1}\mathbf{U}\|_p < 1$
  - This happens for diagonally dominant and symmetric, positive definite matrices ($\lambda_i > 0$).

Iterative Solutions to Lin. Eqns.

- Example:

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{x}_{\text{exact}} = (3/4, 1/2, 1/4)$$

- Try Jacobi: $\mathbf{x}_0 = (1, 0, 0)$

$$\mathbf{x}_{i+1} = \mathbf{D}^{\quad 1}(-\mathbf{R}\mathbf{x}_i + \mathbf{b})$$

- Try Gauss-Seidel: $\mathbf{x}_0 = (1, 0, 0)$

$$\mathbf{x}_{i+1} = \mathbf{L}^{\quad 1}(-\mathbf{U}\mathbf{x}_i + \mathbf{b})$$

Iterative Solutions to Lin. Eqns.

- Example:

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{x}_{\text{exact}} = (3/4, 1/2, 1/4)$$

- Results

iteration	R.E. Jacobi	R.E. Gauss-Seidel
1	38%	40%
2	26%	20%
3	19%	10%
5	9.5%	2.5%
10	1.7%	0.08%

Successive Over Relaxation

- For equations that that do not converge under Jacobi/Gauss-Seidel or any other iterative scheme, there are ways to modify the procedure to force convergence.
  - Suppose we have an iterative map: $\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i)$
    - that gives the sought after solution when $\mathbf{x}_{i+1} = \mathbf{x}_i$
    - the function $\mathbf{f}(\mathbf{x})$ need not be linear in general
  - We modify the map so that:
    - $\mathbf{x}_{i+1} = (1-\omega)\mathbf{x}_i + \omega\mathbf{f}(\mathbf{x}_i)$
    - where the correct solution is still given when $\mathbf{x}_{i+1} = \mathbf{x}_i$
    - where ω is called the relaxation parameter.
  - This new iterative map can damp out any wild fluctuations from one iteration to the next by choosing values: $0 < \omega < 1$

Successive Over Relaxation

- When this damping is applied to Jacobi:
  - The original iterative map: $\mathbf{x}_{i+1} = \mathbf{D}^{\ \ 1}(-\mathbf{R}\mathbf{x}_i + \mathbf{b})$
  - Becomes: $\mathbf{x}_{i+1} = (1-\omega)\mathbf{x}_i + \omega\mathbf{D}^{-1}(-\mathbf{R}\mathbf{x}_i + \mathbf{b})$
    - Matrices that are not diagonally dominant might converge when ω is small enough
- When this dampling is applied to Gauss-Seidel:
  - The original iterative map: $\mathbf{x}_{i+1} = \mathbf{L}^{\ \ 1}(-\mathbf{U}\mathbf{x}_i + \mathbf{b})$
  - Becomes: $\mathbf{x}_{i+1} = (1-\omega)\mathbf{x}_i + \omega\mathbf{L}^{-1}(-\mathbf{U}\mathbf{x}_i + \mathbf{b})$
    - The relaxation parameter acts like an effective increase in the eigenvalues of the matrix. A small enough value can enable convergence.
- Successive over relaxation might be slow, however.