9/19/03

Last time

Matrices - Linear Algebra

Mutiplication $\vec{y} = A\vec{x}$ A is square matrix

$C = AB$ $C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj}$

transpose

determinant

if $\det(A) = 0$ A is singular and

A^{-1} does not exist

inverse $A^{-1}A = I$

Solving $A\vec{x} = \vec{b}$ $A\vec{v} = \lambda\vec{v}$

Defined a matrix norm $\|A\|_F = \left[\sum_i \sum_j |A_{ij}|^2\right]^{1/2}$

If $K(A) = \|A\| \, \|A^{-1}\|$ very large then we have unstable solutions for $\vec{x}$

Solved $L\vec{x} = \vec{b}$ lower triang matrix

$U\vec{x} = \vec{b}$ upper triang matrix

Method called forward substitution

$$x_i = \left(b_i - \sum_{j=0}^{i-1} L_{ij} x_j \right) / L_{ii} \qquad i = 0, 1, \ldots n-1$$

Very similar to back substitution $U\vec{x} = \vec{b}$

U is upper triangular

$$U_{00} x_0 + U_{01} x_1 + U_{02} x_2 = b_0$$
$$U_{11} x_1 + U_{12} x_2 = b_1$$
$$U_{22} x_2 = b_2$$

$$\Rightarrow \quad x_i = \left(b_i - \sum_{j=i+1}^{n-1} U_{ij} x_j \right) / U_{ii} \qquad i = n-1, n-2, \ldots 1, 0$$

Code triang.py

Solving $A\vec{x} = \vec{b}$ with Gaussian elimination

Ex

$$2x_0 + x_1 + x_2 = 8$$
$$x_0 + x_1 - 2x_2 = -2$$
$$5x_0 + 10x_1 + 5x_2 = 10$$

$$\Rightarrow \left(\begin{array}{ccc|c} 2 & 1 & 1 & 8 \\ 1 & 1 & -2 & -2 \\ 5 & 10 & 5 & 10 \end{array} \right)$$

Method: new row i = row i − coeff × $\underbrace{\text{row } j}_{\text{pivot}}$

new row 1 = row 1 − ½ × row ∅

$$\left(\begin{array}{ccc|c} 2 & 1 & 1 & 8 \\ 0 & 1/2 & -5/2 & -6 \\ 5 & 10 & 5 & 10 \end{array}\right)$$

pivot row = 0 new row = 2 coeff = $+5/2$

$$\left(\begin{array}{ccc|c} 2 & 1 & 1 & 8 \\ 0 & 1/2 & -5/2 & -6 \\ 0 & 7\tfrac{1}{2} & 2\tfrac{1}{2} & -10 \end{array}\right)$$

pivot row = 1 new row = 2 coeff $= + \dfrac{7\tfrac{1}{2}}{1/2} = +15$

$$\left(\begin{array}{ccc|c} 2 & 1 & 1 & 8 \\ 0 & 1/2 & -5/2 & -6 \\ 0 & 0 & 40 & 80 \end{array}\right)$$

$x_2 = 80/40$

$x_1 = \dfrac{-6 - (-2.5)x_2}{0.5}$

Easy to code

$j = 0, 1, 2, \ldots n-2$

$i = j+1, j+2, \ldots, n-1$

coeff $= A_{ij}/A_{jj}$

See gauelim.py

We can do something a little more useful by "storing" results of Gauss elimination process so we can use it again for different $\vec{b}$ values

LU decomposition

We can decompose (with some caveats) a square matrix by $A = LU$

$L \equiv$ lower triangular matrix

$U \equiv$ upper triangular matrix

We will make extra assumption that L is unit lower triangular (1's on main diagonal)

(the LU decomp is not necessarily unique –)

our form is called Doolittle

Suppose $A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 1 & -2 \\ 5 & 10 & 5 \end{pmatrix}$ Ignore $\vec{b}$ for now

We do gauss elim

new row 1 = row 1 − $L_{10} \times$ row ϕ $\quad L_{10} = 1/2$

new row 2 = row 2 − $L_{20} \times$ row ϕ $\quad L_{20} = 5/2$

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & 1/2 & -2\tfrac{1}{2} \\ 0 & 7\tfrac{1}{2} & 2\tfrac{1}{2} \end{pmatrix}$$

new row 2 = row 2 − $L_{21} \times$ row 1

$L_{21} = 15$ $\qquad L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 5/2 & 15 & 0 \end{pmatrix}$

$$U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1/2 & -5/2 \\ 0 & 0 & 40 \end{pmatrix} \qquad L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 5/2 & 15 & 1 \end{pmatrix}$$

$A = UL$ $\qquad$ U is result of Gauss elim.

L collects coeff used in elim.

(common to store L and U in single matrix)

Why does this help?

Trying to solve $A\vec{x} = \vec{b}$

We write $A = LU$ $\qquad$ $LU\vec{x} = \vec{b}$

Now write $U\vec{x} = \vec{y} \Rightarrow L\vec{y} = \vec{b}$

We can solve this for $\vec{y}$ by forward substitution

We can solve $U\vec{x} = \vec{y}$ for $\vec{x}$ by backward subst.

[code ludec.py]

Eigenproblems (abbreviated)

$$A\vec{v}_i = \lambda\vec{v}_i$$

assume $n \times n$ matrix A has n eigenvalues λ_i that are distinct (eigenvectors are linearly ~~dep~~ independent)

We can create a matrix Λ that is diagonal with eigenvalues λ_i by

$V^{-1} A V = \Lambda$ and V is eigenvector matrix whose columns are $\vec{v}_i$ $\quad V = (\vec{v}_0 \ \vec{v}_1 \ \dots \ \vec{v}_{n-1})$

"diagonalize A", eigendecomposition

multiply both sides by V on left

$$AV = V\Lambda$$

$$AV = \left[A\vec{v}_0 \ \ A\vec{v}_1 \ \dots \ A\vec{v}_{n-1} \right]$$

$$= \left[\lambda_0\vec{v}_0 \ \ \lambda_1\vec{v}_1 \ \dots \ \lambda_{n-1}\vec{v}_{n-1} \right]$$

$$= \left[\vec{v}_0 \ \ \vec{v}_1 \ \dots \ \vec{v}_{n-1} \right] \begin{bmatrix} \lambda_0 & & & 0 \\ & \lambda_1 & & \\ & & \ddots & \\ 0 & & & \lambda_{n-1} \end{bmatrix}$$

$A U = U \Lambda$ and $U^{-1} A U = \Lambda$ $\blacksquare$

Full solution is too advanced until we have taken linear algebra

We will use <u>Power Method</u> to find largest λ

$|\lambda_0| > |\lambda_1| > |\lambda_2| > \ldots |\lambda_{n-1}|$ since all λ_i distinct

Start with a guess $\vec{z}^{(0)}$ and increment

$$\vec{z}^{(k)} = A \vec{z}^{(k-1)} \qquad k = 1, 2, \ldots$$

We note $\vec{z}^{(k-1)} = A \vec{z}^{(k-2)} = A A \vec{z}^{(k-3)}$ so

$$\vec{z}^{(k)} = A^k \vec{z}^{(0)}$$

Let us assume $\vec{z}^{(0)}$ has a component along $\vec{v}^{(0)}$

We write $\vec{z}^{(0)} = \sum_{i=0}^{n-1} c_i \vec{V}_i$ linear combination of eigenvectors

$$\vec{z}^k = A^k \vec{z}^{(0)} = \sum_i c_i A^k \vec{V}_i = \sum_{i=0}^{n-1} c_i \lambda_i^k \vec{V}_i$$

$$\vec{z}^k = c_0 \lambda_0^k \vec{V}_0 + \lambda_0^k \sum_{i=1}^{n-1} c_i \underbrace{\left(\frac{\lambda_i}{\lambda_0}\right)^k}_{} \vec{V}_i \qquad k = 1, 2, \ldots$$

$$\left(\frac{\lambda_i}{\lambda_0}\right)^k \to 0 \text{ as } k \to \infty$$