Feb 29th

Thursday, February 29, 2024 1:57 PM

Example The ill-conditioned Hilbert matrices, defined by $h_{ij} = \frac{1}{(i+j-1)}$

For example

$$H_4 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

See if you can find L and U.

$$U = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ 0 & \frac{1}{12} & \frac{1}{12} & \frac{3}{40} \\ 0 & 0 & \frac{1}{120} & \frac{9}{700} \\ 0 & 0 & 0 & -\frac{1}{4200} \end{bmatrix}$$

Last row could look like a row of zeros if we only carry 2 decimal places.

---

(3) Given that $A = \begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 2 \\ -1 & -2 & 1 \end{pmatrix}$, $B = \begin{pmatrix} -1- & 0 & 4 \\ -2 & 1 & 2 \\ 0 & -2 & 1 \end{pmatrix}$, $C = \begin{pmatrix} -1- & 1 & 1 \\ -2 & 1 & 2 \\ 1 & -2 & 1 \end{pmatrix}$, $D =$

$E = \begin{pmatrix} -2- & 1 & 4 \\ -2 & 1 & 1 \\ 1 & -2 & 1 \end{pmatrix}$. Answer the following questions:

(a) Multiply $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} C & 0 \\ 0 & D \end{pmatrix}$ in an efficient manner

(b) Multiply $\begin{pmatrix} A^T & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} C^T & 0 \\ 0 & D \end{pmatrix}$ in an efficient manner

(c) Multiply $\begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & D \end{pmatrix}$ in an efficient manner

(d) Multiply $\begin{pmatrix} A & 0 \\ 0 & B^T \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix}$ in an efficient manner

(a) $\begin{pmatrix} AC & 0 \\ 0 & BD \end{pmatrix}$

$AC =$

$BD =$

(5) If we derive an algorithm that is $\mathcal{O}(n^3)$ that solves the system $Ax = b$. How would the length of time (approximately) to compute x be effected if instead of a 100 by 100 matrix $\bar{A}$ we had a 300 by 300 matrix A?

A 3 times as large as $\bar{A}$

$(3)^3 = 27$

(9) Given $A = \begin{pmatrix} -1 & 1 \\ 2 & 3 \end{pmatrix}$ and $\kappa_p(A)$ is the condition number using the p-norm. Find the following.

(a) $\kappa_1(A)$
(b) $\kappa_2(A)$
(c) $\kappa_\infty(A)$

$A = \begin{pmatrix} -\frac{3}{5} & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix}$

$A = \begin{pmatrix} -1 & 1 \\ 2 & 3 \end{pmatrix}$

$K_2(A) = \|A\|_2 \, \|A^{-1}\|_2$

$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}$

How do we translate into matrix norm?

$\sqrt{(-1)^2 + (1)^2} = \sqrt{2}$
$\sqrt{(2)^2 + (3)^2} = \sqrt{13}$

$\sqrt{2 + 13} = \sqrt{15}$

$d_1 = b_1$

$A^{-1} = \begin{pmatrix} -\frac{3}{5} & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix}$

$\sqrt{\frac{9+1}{25}} = \frac{\sqrt{10}}{5}$

$\frac{\sqrt{5}}{5}$

$\sqrt{\frac{10}{25} + \frac{5}{25}} = \frac{\sqrt{15}}{5}$

$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

$a a_1 + b c_1 = 1$
$a b_1 + b d_1 = 0$
$c a_1 + d c_1 = 0$
$c b_1 + d d_1 = 1$

$-a_1 + c_1 = 1$
$-b_1 + d_1 = 0$
$2a_1 + 3c_1 = 0$
$2b_1 + 3d_1 = 1$
$5b_1 = 1$
$b_1 = \frac{1}{5}$

$c_1 = 1 + a_1$
$2a_1 + 3(1 + a_1) = 0$
$5a_1 = -3$
$a_1 = -\frac{3}{5}$
$c_1 = \frac{2}{5}$

$\sqrt{\left(\sqrt{a_{11}^2 + a_{12}^2}\right)^2 + \left(\sqrt{a_{21}^2 + a_{22}^2}\right)^2}$

$= \sqrt{a_{11}^2 + a_{12}^2 + a_{21}^2 + a_{22}^2}$

$\|A\|_2 \, \|A^{-1}\|_2 = (\sqrt{15})\left(\frac{\sqrt{15}}{5}\right) = \frac{15}{5} = \boxed{3}$

$\|A\|_2 \, \|A^{-1}\|_2 = (\sqrt{15})(\frac{\cdot}{5}) = \ldots 5$

LU - Decomposition

* - relatively easy
* - challenging
* - average

1. Overall Code Structure
   - Use comments to show the location of different segments of the code.

2. Check for more efficient algorithm that use Gaussian based on the structure of matrix (sparse, banded)
   ↳ up to 2 people working together

3. Check to see if matrix meets criteria for a Cholesky decomposition

4. Program in Cholesky decomposition

5. Calculate condition number for the matrix A be able to use any p-norm. You may not use the built in norms/condition #s in NumPy. (2 to 3 people)

6. Determine pivots, keep track of row swap

7. Compute column i for matrix L

8. Compute row i for matrix U

9. Write code for back substitution

10. Write code for forward substitution

11. State L, P, U

12. State solution to the system and condition number

(12) State solution to the system and condition number (print out information,

Communication on Discord server

due March 19th

Let's have assignments decided by next Tuesday March 5th.

Error Analysis of Gaussian Elimination

- Roundoff error
- Backward stability

Why small pivots should be avoided

Example Consider the linear system

$$\begin{bmatrix} \boxed{0.002} & 1.231 & 2.471 \\ 1.196 & \boxed{3.165} & 2.543 \\ 1.475 & 4.271 & 2.142 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3.704 \\ 6.904 \\ 7.888 \end{bmatrix}$$

This system is well conditioned

$$K_2(A) \approx 30$$

Let's say we perform Gaussian Elimination without pivoting. Then

$$l_{21} = \frac{1.196}{0.002} = 598$$

$$l_{31} = \frac{1.475}{0.002} = 737.5$$

These values are multiplied by the first row and subtracted from their respective rows

$$3.165 - 598(1.231) = -732.9$$

$$3.165 - 598(1.231) = -732.9$$

The digits in 3.165 were lost due to roundoff

This type of information loss is called <u>swamping</u>

This occurs at other entries of matrix as well.

$$\begin{bmatrix} 0.002 & 1.231 & 2.471 \\ 598 & \boxed{-732.9} & -1475 \\ 737.5 & -903.6 & \boxed{-1820} \end{bmatrix}$$

$$l_{32} = \frac{-903.6}{-732.9} = \cancel{1.233}$$

$$-1820 - (1.233)(-1475) = -1$$

$$\begin{bmatrix} 0.002 & 1.231 & 2.471 \\ 598 & -732.9 & -1475 \\ 737.5 & 1.233 & -1 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 598 & 1 & 0 \\ 737.5 & 1.233 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 3.704 \\ 6.904 \\ 7.858 \end{bmatrix} \qquad Ly = b$$

$$y_1 = 3.704$$
$$y_2 = -2208$$
$$y_3 = -2.000$$

$$598 y_1 + y_2 = 6.904$$
$$y_2 = 6.904 - 598(3.704)$$

$$Ux = y$$

$$x_3 = \frac{y_3}{u_{33}} = \frac{-2.000}{-1.000} = +2$$

We can show $x = \begin{bmatrix} 4.000 \\ -1.012 \\ 2.000 \end{bmatrix}$

But the actual solution is $x = \begin{bmatrix} 1.000 \\ 1.000 \\ \ldots \end{bmatrix}$

But the actual solution is $x = \begin{bmatrix} 1.000 \\ 1.000 \\ 1.000 \end{bmatrix}$

This can found by storing only 4 digits per entry, but allowing for partial pivoting,

Backward Error Analysis of Gaussian Elimination

Gaussian elimination involves accumulation of sums

$$\sum_{j=1}^{n} w_j$$

there are many ways to add n numbers together.

We could work left to right or we could reorder the terms of the summation,

Because of floating point arithmetic the order in which we add could play a significant role.

Goal If a sum is accumulated in floating-point arithmetic, the computation is always backwards stable, regardless of the way the sum is accumulated,

Let u be the unit roundoff.
$O(u^2)$ order of u^2.

For example, $(1+\alpha_1)(1+\alpha_2) = (1+\beta)$ where $\beta = \alpha_1 + \alpha_2 + \alpha_1\alpha_2$

If $|\alpha_1| \approx u$ $|\alpha_2| \approx u$ then $\alpha_1\alpha_2 = O(u^2)$.

so $\beta = \alpha_1 + \alpha_2 + O(u^2)$

Proposition Suppose we compute the sum $\sum_{j=1}^{n} w_j$ using floating point arithmetic with unit roundoff u, [Proposition] Suppose we compute the sum $\sum_{j=1}^{n} w_j$ using floating point arithmetic with unit roundoff u.

Then $$fl\left(\sum_{j=1}^{n} w_j\right) = \sum_{j=1}^{n} w_j(1+\gamma_j)$$

where $|\gamma_j| \le (n-1)u + O(u^2)$, regardless of the order in which the terms are added.

γ_j can be termed relative backwards errors.

The inequality $|\gamma_j| \le (n-1)u + O(u^2)$ overestimates $|\gamma_j|$

$n-1$ = reflects the fact that there at most $n-1$ additions.

thus most γ_j are sums of many fewer than $n-1$ roundoffs.

Bounds like $|\gamma_j| \le (n-1)u + O(u^2)$ takes into account the worst possible cases.

γ_j is more likely to be closer to u than $(n-1)u$,

the factor $(n-1)$ is ignored in practice.