Numerical Analysis 25

Goal: Find a method for solving linear systems. — Without doing a matrix inverse!

$$\left.\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= b_n \end{aligned}\right\}$$

We will use a very basic method: Gaussian Elimination.

BUT to ask a computer to do this, we must be very precise in using the method... we need exact steps.

START — by solving a simple system by hand... really think about the steps we take.

Use this example to check your code...

$$\left.\begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 \quad EQ1 \\ 6x_1 - 2x_2 + 2x_3 &= 6 \quad EQ2 \\ -9x_1 + 7x_2 + x_3 &= -1 \quad EQ3 \end{aligned}\right\}$$

For Gaussian Elimination we Eliminate x_1, x_2 from EQ3 and x_1 from EQ2 so we can then do back substitution.

1. Eliminate x_1 from EQ2 and EQ3.

EQ3 + 3 times EQ1: $x_2 - 2x_3 = -1$
EQ2 − 2 time EQ1: $2x_2 + 4x_3 = 6$

$$\Rightarrow \quad \begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 \\ 2x_2 + 4x_3 &= 6 \\ x_2 - 2x_3 &= -1 \end{aligned}$$

2. Eliminate x_2 from EQ3

EQ3 − ½ EQ2: $-4x_3 = -4$

$$\Rightarrow \quad \begin{aligned} 3x_1 - 2x_2 - x_3 &= 0 \\ 2x_2 + 4x_3 &= 6 \\ -4x_3 &= -4 \end{aligned}$$

$x_3 = \frac{-1}{4}[-4]$, $x_2 = \frac{6 - 4x_3}{2}$, $x_1 = \frac{1}{3}[0 + 2x_2 + x_3]$

3. Solve for $x_3 = 1$, $x_2 = 1$, $x_1 = 1$

Because of the form this part is easy!

ON THE COMPUTER WE USE ARRAYS and ELEMENTARY ROW OPERATIONS.

$$A = \begin{bmatrix} 3 & -2 & -1 \\ 6 & -2 & 2 \\ -9 & 7 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 6 \\ -1 \end{bmatrix}$$

we will talk about how to do this.

$$\left[\begin{array}{ccc|c} 3 & -2 & 1 & 0 \\ 6 & -2 & 2 & 6 \\ -9 & 7 & 1 & -1 \end{array}\right] \xrightarrow[\text{operations}]{\text{same basic}} \begin{bmatrix} 3 & -2 & -1 & 0 \\ 0 & 2 & 4 & 6 \\ 0 & 0 & -4 & -4 \end{bmatrix}$$

So it looks like our goal is to reduce to upper triangular.

GENERALIZE THIS.

In General we reduce $A\vec{x} = b$

$$\Rightarrow [A \,|\, b] = \begin{bmatrix} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{bmatrix}$$

Elementary Row Operations

in $n-1$ steps to the form

$$\begin{bmatrix} \hat{a}_{11} & \cdots & \hat{a}_{1n} & \hat{b}_1 \\ 0 & \ddots & \vdots & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots \; 0 & \hat{a}_{nn} & \hat{b}_n \end{bmatrix} = \begin{bmatrix} \ddots & \hat{a} & \uparrow \\ \emptyset & \ddots & b \\ & & \downarrow \end{bmatrix} \Rightarrow U\vec{x} = g$$

Then using back substitution ... SOLVE FOR $x_1, x_2 \ldots x_n$

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & \\ \vdots & & & \\ 0 & \cdots & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}$$

so $x_n = \dfrac{g_n}{u_{nn}}$

start at the largest x_n

$$x_{n-1} = \frac{g_{n-1} - u_{n-1,n} x_n}{u_{n-1,n-1}}$$

$\vdots$

backward until you reach the first one x_1

and in general

all the elements in the k^{th} row $k \to n^{th}$ col.

$$x_k = \frac{g_k - \{u_{k,k+1} x_{k+1} + u_{k,k+2} x_{k+2} + \cdots + u_{kn} x_n\}}{u_{kk}}.$$

We know we need elementary row operations to get from $A\vec{x} = b$ to $U\vec{x} = g$

By hand we must inspect the eqns to know what to do ... how do we make a computer do this. ??

- Start by assuming that none of the elements in the "first" pivot position are zero ... later we will relax this.

STEP 1. Eliminate x_1 from eqns $2 \to n$.

multiplier: $m_{i1} = \dfrac{a_{i1}^{(1)}}{a_{11}^{(1)}}$ $\quad i = 2 \dots n$ $\quad$ here the superscript (1) denotes the first operation.

this is why we need $a_{11} \neq 0$

Subtract m_{i1} times row 1 from row i $\quad i = 2 \dots n$.

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)} \qquad j = 2 \dots n$$

$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}$$

$$i = 2 \dots n$$

start at $i=2$ because we don't operate on first row.

we start @ $j=2$ not $j=1$ because the first column will all be zero!

STEP 2. Eliminate x_2 from eqns $3 \to n$.

$m_{i,2} = \dfrac{a_{i,2}^{(2)}}{a_{22}^{(2)}}$ again $i = 3 \dots n$

⋮

STEP K Eliminate x_K from eqns $k+1 \to n$ $\quad$ we assume all previous operations were successful.

IN GENERAL

multiplier: $m_{iK} = \dfrac{a_{iK}^{(K)}}{a_{KK}^{(K)}}$ $\quad i = k+1, \dots, n$

pivot $\neq 0$

Subtract m_{iK} times row K from row i for $i = K+1 \dots n$

$$a_{ij}^{(K+1)} = a_{ij}^{(K)} - m_{iK} a_{Kj}^{(m)} \qquad j = K+1 \dots, n$$

$$b_i^{(K+1)} = b_i^{(K)} - m_{iK} b_K^{(K)}$$

STEP $n-1$ is our last step ... this leaves us with: $U\vec{x} = g$

Now how do we deal with zeros in our pivot element?

PARTIAL PIVOTING.

Say $a_{11} = 0$ then our ~~[illegible]~~ method breaks down!

We can deal with this by interchanging rows...

1. look at all the elements in the first column:

$$a_{11}^{(1)}, a_{12}^{(1)}, \ldots, a_{n1}^{(1)}$$

choose the largest one ... largest absolute value.
This is in row k.

2. interchange rows 1 and k in the matrix
3. Proceed with elimination.

We could do this at any point... say $a_{22}^{(2)} = 0$ then we search all elements in the second column below row 2: $a_{22}^{(2)} \; a_{32}^{(2)} \ldots a_{n2}^{(2)}$

and interchange with $\max |a_{2k}^{(2)}|$

IN PRACTICE: We do this at every stage, regardless of whether or not there is a zero.

This is actually GOOD for the error in our numerical method!

JUST KEEP TRACK OF THE PIVOTS!!

print pivots and A to check work.

If we partial pivot regardless of a zero the we gaurantee that

$$|m_{ik}| < 1$$

this is good because if m_{ik} was large we could end up with VERY large #'s

EX - The a_{nn} element is multiplied by ALL of the m_{ik}.

Only error in the method is from rounding avoiding large #'s helps avoid rounding error.

PSUEDO CODE (Together)

PSUEDO CODE

A =
b =
} enter these

Enter the Matrices.

check the size of A and b ... they should match. if not stop!

[n, m] = size(A) ⟶ A should be square! if not stop.

Check that They work.

piv = (1:n)′ initialize the pivot order [1, 2, 3, . . . , n] originally They are in order

start elimination steps.

→ for k = 1 : n-1   n-1 steps.

  [col_max index] = max(abs(A(k:n, k)))   index will be the row where the max is found

  index = index + (k-1) ⟶ do this because of how max(abs(A(2:3, 2))) works.

$\underbrace{A(2:3,2)}_{\text{only counts last two rows}}$ so we need to add k-1

  if index ~= k   [not equal to k Then pivot.]   if it is already in correct spot we leave it alone.

  3 things to switch:

  #1 Switch rows in A:
    temp A = A(k, k:n)
    A(k, k:n) = A(Index, k:n)
    A(index, k:n) = tempA.

You write these.

  #2 Switch rows in b: temp b = b(k)
                b(k) = b(index)
                b(index) = temp b

  ~~[illegible]~~ we also keep track of the pivots

  ~~[illegible]~~

  Check what we are getting in the end.

  #3 switch rows in pivot.

  end.

form multipliers . . . some store these in the soon to be zeros in The pivot colmn. others define a new matrix m.

$$\begin{bmatrix} A_1 & B_1 & C_1 & D_1 \\ M & A_2 & B_2 & C_2 \\ M & M & A_3 & B_3 \\ M & M & M & A_4 \end{bmatrix}$$

in the end the zeros places are filled w/ multipliers.

all rows beyond k   in col k

  A(k+1:n, k) = A(k+1:n, k)/A(k, k) → gives the multiplier for each remaining rows. STORES in "zero" spots. We don't actually set these to zero.

  for i = k+1 : n

  Take my row   minus   "Pivot row"   multiplier

    A(i, k+1:n) = A(i, k+1:n) - A(i, k) * A(k, k+1:n)

  end   does The row operation on A

  b(k+1:n) = b(k+1:n) - A(k+1:n, k) b(k)
    does The row operation on b.

end.

loop

Then solve the Uppertriangular System . . .

```
x = zeros(n,1)
x(n) = b(n)/A(n,n)

for i = n-1 : -1 : 1
    x(i) = ( b(i) - A(i, i+1:n) * x(i+1:n) ) ./ A(i,i)
end
```

X is your solution matrix!

NOTE. this is approximately $\frac{1}{3}n^3$ operations so for a LARGE system it is not great Computationally Expensive.

NEXT TIME Iterative Methods