

## Math 218D Problem Session

Week 3

### 1. Finding $PA = LU$

a) Find the  $PA = LU$  decomposition of  $A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 1 & 2 & 3 \end{pmatrix}$  using the 3-column method.

b) Find the  $A = LU$  decomposition of  $A = \begin{pmatrix} 1 & 1 & 1 \\ -10 & -20 & -30 \\ 5 & 15 & 10 \end{pmatrix}$ .

c) Find the  $PA = LU$  decomposition of  $A = \begin{pmatrix} 1 & 1 & 1 \\ -10 & -20 & -30 \\ 5 & 15 & 10 \end{pmatrix}$ , this time using *maximal partial pivoting*.

### 2. Using $PA = LU$

In this problem, we'll solve the matrix equation

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 1 & 2 & 3 \end{pmatrix} x = \begin{pmatrix} 1 \\ 5 \\ 2 \end{pmatrix}$$

using the  $PA = LU$  decomposition of problem 1a).

- Identify  $A$ ,  $b$ ,  $P$ ,  $L$ , and  $U$ .
- Compute  $Pb$ . What does  $P$  do to  $b$ ?
- Convert  $Lc = Pb$  into 3 linear equations, and solve for  $c = (c_1, c_2, c_3)$  using forward-substitution.
- Convert  $Ux = c$  into 3 linear equations, and solve for  $x$  using back-substitution.
- Check your answer, by multiplying  $A \cdot x$  and confirming that it equals  $b$ .

### 3. Parametric forms

Consider the augmented matrix  $(A | b) = \left( \begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 1 & -2 & -1 & 1 \\ 0 & -3 & -2 & 1 \end{array} \right)$ .

- Compute the RREF, and verify that the system of equations  $Ax = b$  has one free variable. Which variable is it?
- Find the parametric form of the solution:

$$\begin{aligned}x_1 &= (?) \\x_2 &= (?) \\x_3 &= (?)\end{aligned}$$

where all the (?) only involve scalars and the free variable. What are two different solutions to the system of equations?

- Find the parametric vector form:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = (\text{free variable}) \cdot \begin{pmatrix} (?) \\ (?) \\ (?) \end{pmatrix} + \begin{pmatrix} (?) \\ (?) \\ (?) \end{pmatrix}$$

where all the (?) are scalars.

- This is the parametric vector form of a line. Find a point that this line passes through. What direction is the line pointing? Check your answer with this [demo](#).
- Let's modify the  $b$ -vector:  $b = (0, 1, -1)$ . How many solutions does this new system of equations have? Check your answer geometrically by moving the  $b$  vector in the demo linked above.
- Now find the parametric vector form of the homogeneous equation  $Ax = 0$ . How is this related to your answer in **d**)? Check your answer geometrically by moving the  $b$  vector in the demo linked above.
- Describe all the  $b_1, b_2, b_3$  which make

$$\left( \begin{array}{ccc|c} 1 & 1 & 1 & b_1 \\ 1 & -2 & -1 & b_2 \\ 0 & -3 & -2 & b_3 \end{array} \right)$$

consistent. Your answer should involve a single linear equation in the variables  $b_1, b_2, b_3$ .

- What shape (point, line, plane, ...) do you get if you add together the vectors

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \\ -3 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -2 \end{pmatrix}$$

in every way possible? Does this shape contain the point  $(0, 1, 1)$ ? How about  $(0, 1, -1)$ ? or  $(0, 0, 0)$ ? How does this relate to **g**)? Check your answer with

this demo.

$$\mathbf{Hint:} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 0 & -3 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ -2 \\ -3 \end{pmatrix} + x_3 \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}.$$

#### 4. Maximal Partial Pivoting

Consider the linear system

$$\begin{aligned}x_2 &= 1 \\x_1 + x_2 &= 2.\end{aligned}$$

Clearly the solution is  $x_1 = 1$  and  $x_2 = 1$ . Let's modify the system just a little bit:

$$\begin{aligned}10^{-17}x_1 + x_2 &= 1 \\x_1 + x_2 &= 2.\end{aligned}$$

Presumably the solution  $(x_1, x_2)$  will be very close to  $(1, 1)$ .

a) Perform Gauss–Jordan elimination on the augmented matrix

$$\left( \begin{array}{cc|c} 10^{-17} & 1 & 1 \\ 1 & 1 & 2 \end{array} \right)$$

to solve the modified system. You should obtain

$$x_1 = \frac{1}{1 - 10^{-17}} \quad x_2 = 2 - \frac{1}{1 - 10^{-17}},$$

which are indeed very close to 1.

Now let's see if a computer can do the same. Load up [linalg.js](#), which can be found on the course homepage, and open a Javascript console in your browser (follow the instructions on that page). Create the augmented matrix as follows:

```
A = mat([1e-17, 1, 1], [1, 1, 2])
```

In `linalg.js`, matrices are just arrays of arrays, so you can inspect their elements as follows:

```
A[0][0] // 1e-17
```

Note that Javascript arrays are indexed from zero, the above command prints the  $(1,1)$  entry.

b) Now let's perform Gauss–Jordan elimination:

```
A.rowReplace(1,0,-1/A[0][0])
A.rowScale(1,1/A[1][1])
A.rowReplace(0,1,-A[0][1]/A[1][1])
A.rowScale(0,1/A[0][0])
```

The first command translates into  $R_2 \leftarrow R_2 - 1/10^{-17}R_1$ : the first argument to `rowReplace` is the row to replace (indexed from zero), the second is the row to add/subtract, and the third is the scaling factor.

c) Verify that the resulting matrix has the form

$$\left( \begin{array}{cc|c} 1 & 0 & (?) \\ 0 & 1 & (?) \end{array} \right).$$

d) What does the computer think  $x_1$  and  $x_2$  are? What went wrong?

- e) Javascript uses IEEE-754 64-bit floating point numbers. This means that they have about 16 decimal digits of precision. Try evaluating  $1+1e17$  in your console. What did you get?

The problem was that you produced enormous numbers by dividing by the tiny number  $10^{-17}$ . When you're doing math on a computer, *you never want to divide by tiny numbers*.

- f) Now try performing Gauss–Jordan elimination again, after selecting the maximal pivot in the first column:

```
A = mat([1e-17, 1, 1], [1, 1, 2])
```

```
A.rowSwap(0,1)
```

Did that work? What does the computer think  $x_1$  and  $x_2$  are now?

## 5. $PA = LU$ on a computer

The purpose of this problem is to convince you that computing a  $PA = LU$  decomposition really is faster for solving  $Ax = b$  for many values of  $b$ . Load up [linalg.js](#) in your browser, and open a Javascript console.

a) Let's create a  $1000 \times 1000$  invertible matrix:

```
A = Matrix.identity(1000).add(Matrix.constant(1,1000))
```

The resulting matrix is

$$\begin{pmatrix} 2 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 2 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 2 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 2 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 2 \end{pmatrix}.$$

b) Let's solve  $Ax = b$  using a  $PA = LU$  decomposition.

```
A.PLU() // Computes and caches a PA=LU decomposition  
b = Vector.constant(1000,1)  
for(i = 0; i < 1000; ++i) A.solve(b)
```

This solves  $Ax = (1, 1, \dots, 1)$  1000 times, using the  $PA = LU$  decomposition. On my computer, both steps take a few seconds.

c) Now let's solve  $Ax = b$  without using  $PA = LU$ .

```
for(i = 0; i < 1000; ++i) { A.invalidate(); A.solve(b); }
```

When you run `A.solve(b)`, the library actually computes and caches the  $PA = LU$  decomposition, since that's no more difficult than running Gauss–Jordan elimination anyway. The command `A.invalidate()` clears that cache to force the library to run elimination 1000 times.

The above command crashed my browser tab.