Privoting Strategies for Gaussian Elimination

1. Problems with Basic Gaussian Elimination

In the preceding lecture we discussed the basic algorithm for Gaussian elimination. Unfortunately, this algorithm does not always work. Consider the following simple example

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

If we were to use Gaussian elimination to produce an equivalent lower triangular system, then we would meet the following obstruction: there is no way to add a multiple of the first row to the second row in order to kill off the first entry in the second row.

The basic Gaussian elimination algorithm is also a bit problematical when a **pivot element** is small. For consider the following system

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad , \quad \varepsilon << 1$$

After the first stage of Gaussian elimination we have

$$\begin{pmatrix} \varepsilon & 1 \\ 0 & 1-\varepsilon^{-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2-\varepsilon^{-1} \end{pmatrix}$$

the solution of which is

$$\begin{aligned} x_2 &= \frac{2-\varepsilon^{-1}}{1-\varepsilon^{-1}} \\ x_1 &= (1-x_2)\varepsilon^{-1} \end{aligned}$$

On a computer, if ε is small enough both $2-\varepsilon^{-1}$ and $1-\varepsilon^{-1}$ will be computed to be the machine number corresponding to $-\varepsilon^{-1}$. Hence, $x_2 \equiv 1$ and so $x_1 \equiv 0$.

However, the exact solution is

$$\begin{aligned} x_1 &= \frac{1}{1-\varepsilon} \approx 1 \\ x_2 &= \frac{1-2\varepsilon}{1-\varepsilon} \approx 1 \end{aligned}$$

Thus, the computed solution for x_1 would be way off.

A final example will show that it is not so much the smallness of the coefficient a_{11} that's causing the trouble, but rather the smallest of its size relative to the other elements in its row. To see this, let's multiply the first row in the example above by ε^{-1} to obtain

$$\begin{pmatrix} 1 & \varepsilon^{-1} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \varepsilon^{-1} \\ 2 \end{pmatrix}$$

This is an equivalent system of equations, but now a_{11} is no longer particularly small. Gaussian elimination leads us to

$$\begin{pmatrix} 1 & \varepsilon^{-1} \\ 0 & 1-\varepsilon^{-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \varepsilon^{-1} \\ 2-\varepsilon^{-1} \end{pmatrix}$$

the solution of which would be

$$\begin{aligned} x_2 &= \frac{2-\varepsilon^{-1}}{1-\varepsilon^{-1}} \\ x_1 &= (1-x^2)\varepsilon^{-1} \end{aligned}$$

and we have the same problem as before $x_1 \approx 0 \neq 1$.

1.1. Pivoting Strategies. The difficulties in all the examples above can be avoided if the order of equations is changed. For example,

$$\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

is an equivalent system. but now Gaussian elimination produces

$$\begin{pmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{pmatrix} \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1-2\varepsilon \end{pmatrix}$$

or

$$\begin{aligned} x_1 &= \frac{1-2\varepsilon}{1-\varepsilon} \approx 1 \\ x_2 &= 2 - x_1 \approx 1 \end{aligned}$$

Let me now show you how this simple idea of interchanging rows can be developed into a working algorithm.

First some notation. By a *row permutation* we shall mean a reordering of the rows of a matrix. The possible row permutations of an $n \times n$ matrix can be prescribed by special n-dimensional vectors as follows. Let $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ be a n-dimensional vector such that for each integer i from 1 to n there is some integer $j \in \{1, \ldots, n\}$ such that $p_j = i$. (In other words, the components of $\mathbf{p}$ must coincide wtih some reordering of the integers from 1 to n.) The row permutataion of an $n \times n$ matrix $\mathbf{A}$ corresponding to such a permutation vector $\mathbf{p}$ will then be the matrix whose i^{th} row is identical to the $(p_i)^{th}$ row of $\mathbf{A}$.

EXAMPLE 12.1. Let

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix}$$

Then the row permutation of $\mathbf{A}$ corresponding to the permutation vector $\mathbf{p} = (3, 1, 2)$ will be

$$\mathbf{PA} = \begin{pmatrix} 3 & 6 & 9 \\ 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix}$$

REMARK 12.2. As the notation in the preceding example suggests. A row permutation of a matrix $\mathbf{A}$ can also be viewed as the matrix product of a specical matrix $\mathbf{P}$ and the matrix $\mathbf{A}$. Indeed, let $\mathbf{p} = (p_1, \ldots, p_n)$ be a permutation vector, and let $\mathbf{P}$ be defined by

$$(\mathbf{P})_{ij} = \begin{cases} 1 & \text{if } j = p_i \\ 0 & \text{if } j \neq p_i \end{cases}$$

then the row permutation of $\mathbf{A}$ corresponding to the permutation vector $\mathbf{p}$ is precisely the matrix product $\mathbf{PA}$.

Now that we have generalized and systematized the notion of interchanging the rows of a matrix, the next thing we need to do is figure out a way of identifying a row permutation that improves, if not makes feasible, our basic Gaussian elimination algorithm.

In the examples above we demonstrated problems that arise when a pivot element is too small relative to the other elements in its row. The basic idea for circumventing these difficulties was to switch a "bad" row with a "better" one below it. To implement this idea, we must decide how to measure the 'badness" and "goodness" of rows.

Since the crux of the problem has to do with the size of the leading element of a pivoting row compared to the size of the other non-trivial elements in the row, a natural way to choose a row to interchange with the first row (before the first stage of Gaussian elimination) would be to calculate for each i from k to n the numbers

$$\begin{aligned} s_i &= \max\{|a_{ik}|, |a_{i,k+1}|, \ldots, |a_{in}|\} \\ q_i &= \frac{|a_{ki}|}{s_i} \end{aligned}$$

We shall refer to these numbers, respectively, as the **scale** and **quality** of the ith row at the kth stage of Gaussian elimination. In our improved Gaussian algorithm we will seek to identify row with maximal q_i and then interchange that row with the first row before carrying out the next stage of Gaussian elimination. This technique is called **Gaussian Elimination with Scaled Row Pivoting**.

EXAMPLE 12.3. Consider the matrix equation

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 6 \\ 1 & 1 & 1 \\ 1 & 3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

We intend to keep track of the row exchanges by using a permutation vector $\mathbf{p}$. Initially, we'll have $\mathbf{p} = (1, 2, 3)$; since $\mathbf{PA} = \mathbf{A}$ (meaning, the permutated matrix has the first row of $\mathbf{A}$ as its first row, the second row of $\mathbf{A}$ as its second row, etc.).

The initial scales of the rows of $\mathbf{A}$ are evidently

$$s_1 = 6 \quad , \quad s_2 = 1 \quad , \quad s_3 = 3$$

and so the corresponding qualities are

$$q_1 = \frac{1}{6} \quad , \quad q_2 = 1 \quad , \quad q_3 = \frac{1}{3}$$

So the second row has the highest quality. Therefore, before carrying out the row operations we'll interchange the second row with the first.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 6 \\ 1 & 3 & 3 \end{bmatrix}$$

This operation corresponds to the permutation vector $\mathbf{p} = (2, 1, 3)$. Now we carry out the first stage of Gaussian elimination.

$$\Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ \underline{1} & 2 & 5 \\ \underline{1} & 2 & 2 \end{bmatrix}$$

(Note our use of the improved notation for carrying out simultaneous Gaussian elimination and LU factorization.) Now the qualities of the second and third rows are

$$\begin{aligned} q_2 &= \frac{2}{5} \\ q_3 &= 1 \end{aligned}$$

Because the quality of the third row is greater than that of the second, we interchange those to rows before carrying out the last stage of Gaussian elimination

$$\begin{bmatrix} 1 & 1 & 1 \\ \underline{1} & 2 & 5 \\ \underline{1} & 2 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ \underline{1} & 2 & 2 \\ \underline{1} & 2 & 5 \end{bmatrix} \quad , \quad \mathbf{p} = (2,1,3) \quad \Rightarrow \quad \mathbf{p} = (2,3,1)$$

$$2^{nd} \text{ stage of Gaussian Elimination} \Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ \underline{1} & 2 & 2 \\ \underline{1} & \underline{1} & 3 \end{bmatrix}$$

Pulling the matrix apart into its lower triangular and upper triangular factors yields

$$\mathbf{LU} = \begin{pmatrix} 1 & 0 & 0 \\ \underline{1} & 1 & 0 \\ \underline{1} & \underline{1} & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 3 & 6 \end{pmatrix}$$

Note that $\mathbf{LU}$ is not a factorization of the original matrix $\mathbf{A}$ but rather the permutation of $\mathbf{A}$ corresponding to the final permutation vector $\mathbf{p} = (2,3,1)$. That's okay though, because even though we don't have an LU factorization appropriate for the equation $\mathbf{Ax} = \mathbf{b}$, we do have one for the equivalent equation $\mathbf{PAx} = \mathbf{Pb}$. In other words, we can determine the solution of $\mathbf{Ax} = \mathbf{b}$ by interchanging the components of $\mathbf{b}$ according to the permutaion vector $\mathbf{p}$ and then use the LU factorization of $\mathbf{PA}$ to solve

$$\mathbf{LUx} = \mathbf{PAx} = \mathbf{Pb}$$

for $\mathbf{x}$.

1.2. General Algorithm for Gaussian Elimination with Scaled Row Pivoting. It's time now to generalize and systematize the procedure used in the preceding example. This procedure added two additional tasks to the basic Gaussian elimination algorithm:

1. Identification of the row that had the highest "quality", and selecting that row as the pivoting row.
2. Interchanging rows and keeping track how we've permutated rows.

Now the first task is fairly easy to implement. We simply calculate the scales s_i and qualitities q_i of the rows and select the row corresponding to the maximal value of q_i.

To accomplish the second task, we'll use permutation vectors to keep track of the row interchanges. To wit, the ith row after a row permutation prescribed by a vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ will be the $(p_i)^{th}$ row of the original matrix. For example, if

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$\mathbf{p} = (3,1,2)$, and then

$$\mathbf{PA} = \begin{pmatrix} 7 & 8 & 9 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

What's nice about this technique for describing row interchanges is that we can insulate the complications of row interchanges from the rest of the Gaussian algorithm by precribing the row indices by the corresponding

components of the permutation vector $\mathbf{p}$. For example, suppose we knew that the permuted matrix $\mathbf{PA}$ was upper triangular. Then we could solve

$$(\mathbf{PA})\mathbf{x} = \mathbf{Pb}$$

using

$$\begin{aligned} x_{n-i} &= \frac{1}{(\mathbf{PA})_{n-i,n-i}} \left((\mathbf{Pb})_{n-i} - \sum_{s=0}^{i-1} (\mathbf{PA})_{n-i,s}\, x_s \right) \\ &= \frac{1}{a_{p_{n-i},n-i}} \left(b_{p_{n-i}} - \sum_{s=0}^{i-1} a_{p_{n-i},s} x_s \right) \end{aligned}$$

Note here that $a_{p_{n-i},j}$ is the jth element in the $(p_{n-i})^{th}$ row of the original matrix $\mathbf{A}$.

Let me now write down the Maple code that carries out Gaussian elimination with scaled row pivoting. (Note that code below assumes that an n-dimensional matrix $\mathbf{A}$ and an n-dimensional column vector $\mathbf{b}$ have already been declared and intialized with numerical values).

```
s := array(1..n);    # vector to hold row scales
p := array(1..n);    # permutation vector
P := array(1..n,1..n); # matrix corresponding to p
PA := array(1..n,1..n); # matrix corresponding to final permuation of A
                        # at the final stage of Gaussian elimination
L := array(1..n,1..n);  # the lower triangular factor
U := array(1..n,1..n);  # the upper trianuglar factor of
LU := array(1..n,1..n); # the matrix product of L and U

#initialize the permutation vector p and the row scales
for i from 1 to n do
   p[i] := i;
   s[i] := 0;
   for j from 1 to n do
      if abs(A[i,j]) > s[i] then s[i] := abs(A[i,j]) fi;
    od;
   s[i];
od;

n1 := n-1;   # many sums are from 1 to n-1

for k from 1 to n1 do
   i := k;
   pk := p[k];
   maxq := abs(A[pk,k])/s[pk];   # maximum quality
#find row with highest quality
   for j from k+1 to n do
      pj := p[j];
      q := abs(A[pj,k])/s[pj];
      if q > maxq then
            maxq := q;
            i := j;
      fi;
   od;
# update p
   tmp := p[k];
```

```
   p[k] := p[i];
   p[i] := tmp;
   pk := p[k];
   k;
# carry out kth stage of Gaussian elimination
   for i from k+1 to n do
      pi := p[i];
       z := A[pi,k]/A[pk,k];
      A[pi,k] := z;
      for j from k+1 to n do
         A[pi,j] := A[pi,j] - z*A[pk,j];
      od;
   od;
od;
# end of Gaussian elimination

# form the permutation matrix corresponding to p
for i from 1 to n do
   for j from 1 to n do
      if p[i] = j then P[i,j] := 1 else P[i,j] := 0 fi;
   od;
od;

# now we'll explicitly swap the rows in end product A
for i from 1 to n do
   for j from 1 to n do
        PA[i,j] := add(P[i,k]*A[k,j],k=1..n);
   od;
od;

print(`end product of Gaussian elimination is`,A);
print(`PA is`, PA);

for i from 1 to n do
   for j from 1 to n do
      if i<j then
         U[i,j] := PA[i,j];
         L[i,j] := 0;
      elif i=j then
         U[i,j] := PA[i,j];
         L[i,j] := 1;
      else
         U[i,j] := 0;
         L[i,j] := PA[i,j];
      fi;
   od;
od;

print(`L is`, L);
print(`U is`, U);

for i from 1 to n do
   for j from 1 to n do
      LU[i,j] := add(L[i,k]*U[k,j],k=1..n);
```

```
   od;
od;

print(`LU is`, LU);
```

Note: I have again I introduced additional array variables (**PA**, **L**, **U**, and **LU**) for conceptual clarity.