IB Paper 7: Linear Algebra Handout 3

3.5 Making LU decomposition robust – Partial Pivoting

When we say an algorithm is robust, we mean that it should be able to obtain an accurate solution for the majority of cases without user intervention i.e. it should not be easily "fooled" or get "hung up" by particular combinations of data. This raises the question what might go wrong with the algorithm we have used so far. It helps to look at that algorithm in a bit more detail, exemplified initially using 3×3 matrices.

Step 1 Choose $\underline{\tilde{u}}_1^{\mathrm{T}} = \begin{bmatrix} a & b & c \end{bmatrix}$ and $\underline{l}_1 = \begin{bmatrix} 1 \\ d \\ e \end{bmatrix}$ so that

$$\mathbf{R}_1 = \mathbf{A} - \underline{l}_1 \underline{\tilde{u}}_1^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & r & s \\ 0 & t & u \end{bmatrix}$$

$$\Rightarrow \qquad \mathbf{R}_1 = \mathbf{A} - \begin{bmatrix} 1 \\ d \\ e \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \mathbf{A} - \begin{bmatrix} a & b & c \\ da & db & dc \\ ea & eb & ec \end{bmatrix}$$

It is clear that $\underline{\tilde{u}}_1$ must be the first row of $\mathbf{A}$, and that $d = \dfrac{a_{21}}{a}$ and $e = \dfrac{a_{31}}{a}$

Step 2 Choose $\underline{\tilde{u}}_2^{\mathrm{T}} = \begin{bmatrix} 0 & g & h \end{bmatrix}$ and $\underline{l}_2 = \begin{bmatrix} 0 \\ 1 \\ f \end{bmatrix}$ so that

$$\mathbf{R}_2 = \mathbf{R}_1 - \underline{l}_2 \underline{\tilde{u}}_2^{\mathrm{T}} = \mathbf{A} - \underline{l}_1 \underline{\tilde{u}}_1^{\mathrm{T}} - \underline{l}_2 \underline{\tilde{u}}_2^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & w \end{bmatrix}$$

$$\Rightarrow \quad \mathbf{R}_2 = \mathbf{R}_1 - \begin{bmatrix} 0 \\ 1 \\ f \end{bmatrix} \begin{bmatrix} 0 & g & h \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & r & s \\ 0 & t & u \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & g & h \\ 0 & fg & fh \end{bmatrix}$$

so that $\underline{\tilde{u}}_2$ is the second row of $\mathbf{R}_1$ and $f = \dfrac{t}{g}$ what happens when $g = 0$?

After each step of the process the remainder matrix has one more row and one more column of zeros, and the process continues until we run out of rows. This gives us the potential problem about what to do if g (or its equivalent element in a later stage of the process when $\mathbf{A}$ is bigger than 3×3) turns out to be 0 . If you look back at the end of the previous handout, you will see that the

thing we divide by at each step is called the *pivot* element for this step. This symptom is, therefore, often referred to as a zero candidate pivot element.

If we were doing this by hand, we would know what to do as the next example illustrates, we would choose a different pivot.

Example

Perform LU on $\begin{bmatrix} 2 & 1 & -2 \\ 6 & 3 & -3 \\ 4 & 3 & 0 \end{bmatrix}$

Step 1

$$\begin{bmatrix} 2 & 1 & -2 \\ 6 & 3 & -3 \\ 4 & 3 & 0 \end{bmatrix} = \begin{matrix} 2 \\ 3 \\ 2 \end{matrix} \begin{bmatrix} 2 & 1 & -2 \\ 6 & 3 & -6 \\ 4 & 2 & -4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 1 & 4 \end{bmatrix}$$

can't use this do zero the second column

so use this instead

Step 2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 1 & 4 \end{bmatrix} = \begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \begin{bmatrix} 0 & 1 & 4 \\ 0 & 0 & 0 \\ 0 & 1 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

Note that we have still zeroed one additional row and one column in this step, so the decomposition is still on track.

Step 3

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix} = \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

Pulling this together

$$\mathbf{A} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \begin{bmatrix} 2 & 1 & -2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 4 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & -2 \\ 0 & 1 & 4 \\ 0 & 0 & 3 \end{bmatrix} = \hat{\mathbf{L}}\mathbf{U}$$

Note that $\hat{\mathbf{L}}$ isn't quite lower triangular but, in the immortal words of the MATLAB help file, $\hat{\mathbf{L}}$ is "psychologically equivalent" to a lower triangular matrix.

We can tidy this up a bit if we now permute the rows of $\hat{\mathbf{L}}$ to get it into triangular form

$$\Rightarrow \quad \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

$L \qquad P^T \qquad \hat{L}$

(note that in this case $P^T = P^{-1} = P$)

Please forgive this notation. It is designed so that we end up with something snappy.

Now, a permutation matrix has rows and columns which are orthogonal unit vectors, so it is an orthogonal matrix which means that its inverse is its transpose. We have now an extended LU decomposition of the form

$$A = \hat{L}U = P^T L U$$

$\Rightarrow$

$$\boxed{PA = LU}$$

where **P** is a permutation matrix.

Computer algorithms will ensure that there are a minimum of difficulties of this type during the factorisation process by, at every stage, scanning the remaining non-zero elements in the next column in the remainder matrix to be zeroed and choosing the largest (in absolute value). This technique is called *partial pivoting* . (*Full pivoting* would involve scanning all of the remaining non-zero elements and choosing to use the largest one as the tool for zeroing whatever column it happened to be in).

It is a matter of personal preference whether one thinks of deriving a "psychologically equivalent" lower triangular matrix, as we did above, or whether one swaps rows of **A** whenever this type of problem occurs to use the original LU with a modified matrix **A** which has permuted rows. As long as you keep track of what is being permuted where, then it makes no difference.

MATLAB and/or **OCTAVE** do it this way by default.

[L , U, P] = lu(A)

Example

scan first column

Use partial pivoting on the matrix

$$\mathbf{A} = \begin{bmatrix} .6 & 2.04 & .2 \\ .3 & .62 & 1.06 \\ 3 & .2 & 0 \end{bmatrix}$$

The largest pivot element in the first column is 3.

Step 1

$$\begin{bmatrix} .6 & 2.04 & .2 \\ .3 & .62 & 1.06 \\ 3 & .2 & 0 \end{bmatrix} = \begin{matrix} 3 \\ .2 \\ .1 \\ 1 \end{matrix}\begin{bmatrix} 3 & .2 & 0 \\ .6 & .4 & 0 \\ .3 & .02 & 0 \\ 3 & .2 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 & .2 \\ 0 & .6 & 1.06 \\ 0 & 0 & 0 \end{bmatrix}$$

Step 2

$$\begin{matrix} 0 \\ 1 \\ .3 \\ 0 \end{matrix}\begin{bmatrix} 0 & 2 & .2 \\ 0 & 2 & .2 \\ 0 & .6 & .06 \\ 0 & 0 & 0 \end{bmatrix} + \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \end{matrix}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

For this step we have used 2 as the largest potential pivot. In matrix form

$$\mathbf{A} = \begin{bmatrix} .2 & 1 & 0 \\ .1 & .3 & 1 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 3 & .2 & 0 \\ 0 & 2 & .2 \\ 0 & 0 & 1 \end{bmatrix}$$

Permute the rows,

$$\Rightarrow \quad \begin{bmatrix} .2 & 1 & 0 \\ .1 & .3 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ .2 & 1 & 0 \\ .1 & .3 & 1 \end{bmatrix}$$

giving

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} .6 & 2.04 & .2 \\ .3 & .62 & 1.06 \\ 3 & .2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ .2 & 1 & 0 \\ .1 & .3 & 1 \end{bmatrix}\begin{bmatrix} 3 & .2 & 0 \\ 0 & 2 & .2 \\ 0 & 0 & 1 \end{bmatrix}$$

A further consequence of this way of choosing the pivot element is that all of the pivot elements have ended up large and all of the multipliers are less than unity.

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ .2 & 1 & 0 \\ .1 & .3 & 1 \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} 3 & .2 & 0 \\ 0 & 2 & .2 \\ 0 & 0 & 1 \end{bmatrix}$$

multipliers

pivots

The **L** and **U** that we have ended up with have their largest terms on the diagonal (and so will be well conditioned when it comes to forward and backward substitution). This is a good thing, as the following example shows.

3.6 Ill-conditioned problems

The examples presented in this section are somewhat extreme, so that we can see the principles on small (2×2 matrices). The general $m \times n$ case shows exactly the same features/problems but with less extreme data.

Suppose $\quad \mathbf{A}\underline{x} = \underline{b} \quad \begin{bmatrix} .0001 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2.0001 \\ 2 \end{bmatrix}$

Performing **non-intelligent basic LU** decomposition leads us to

small pivot

$$\begin{bmatrix} .0001 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix} \begin{bmatrix} .0001 & 2 \\ 0 & -19999 \end{bmatrix}$$

large multipliers

We re-write $\mathbf{A}\underline{x} = \underline{b}$ as $\mathbf{LU}\underline{x} = \underline{b}$ and solve $\mathbf{U}\underline{x} = \mathbf{c}$, where $\mathbf{L}\underline{c} = \underline{b}$

$$\mathbf{L}\underline{c} = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 2.0001 \\ 2 \end{bmatrix} = \underline{b}$$

Solving this by forward substitution $\quad c_1 = 2.0001$

$$c_2 = 2 - 10^4 c_1 = 2 - 20001 = -19999$$

Then $\mathbf{U}\underline{x} = \underline{c}$ is solved by backward substitution

$$\begin{bmatrix} .0001 & 2 \\ 0 & -19999 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2.0001 \\ -19999 \end{bmatrix}$$

This gives

$$y = 1$$
$$x = 10^4 (2.0001 - 2y) = 10^4 \times .0001 = 1$$ which is correct

Now this is all well and good if we are working to *exact* precision. Machines, however, will always be working to a limited number of significant figures. If, for example, y was in error by 1 part in 10^{-6}, $y = 1.000001$, then this last equation becomes

$$x = 10^4 (2.0001 - 2y) = 10^4 \times .000098 = .98 \qquad (*)$$

i.e. we don't even have three significant figure accuracy for x, which is not exactly robust.

The trouble comes from the fact that the pivot for x , .0001, is very small, and so any errors on the right-hand side of (*) are immediately multiplied by a large number (10^4). This equation is said to be *ill-conditioned*.

For the **extended LU method**, it is clear that the permutation matrix will simply swap the rows, so then we can proceed naively. This time we solve $\quad P\mathbf{A}\underline{x} = P\underline{b}$ (not $\mathbf{A}\underline{x} = \underline{b}$)

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} .0001 & 2 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 2.0001 \\ 2 \end{bmatrix} \quad \text{i.e.} \quad \begin{bmatrix} 1 & 1 \\ .0001 & 2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix}$$

LU decomposition gives $\begin{bmatrix} 1 & 1 \\ .0001 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1.9999 \end{bmatrix}$

$$\mathbf{L}\underline{c} = \mathbf{P}\underline{b} \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix}$$

small multiplier

Solving this by forward substitution

$$c_1 = 2$$
$$c_2 = 2.0001 - 10^{-4}c_1 = 2.0001 - 0.0002 = 1.9999$$

Then $\mathbf{U}\underline{x} = \underline{c}$ is solved by backward substitution

pivots not small

$$\begin{bmatrix} 1 & 1 \\ 0 & 1.9999 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1.9999 \end{bmatrix}$$

This gives $y = 1$, $x = 2 - y = 1$. Any error in y then gives a *comparable error* in x.

Final Note

LU in its partial-pivoting mode is immune to *introducing* ill-conditioning into a problem. But it should be noted that some *problems* are ill-conditioned (as opposed the *algorithm* used to solve them being ill-conditioned). All algorithms (including LU) will stuggle with ill-conditioned problems.

e.g. $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 4.0001 \end{bmatrix}$

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} 2 \\ 4.0001 \end{bmatrix} \Rightarrow x = 0,\ y = 1$$

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \Rightarrow x = 2,\ y = 0\ !$$

You can see that det **A** = .0001 which is four orders of magnitude less than a typical element. The matrix **A** is "nearly singular". When faced with a problem like this it usually means that you have set the problem up the wrong way.

Key Point from Lecture

LU decomposition with partial pivoting is **PA** = **LU** **You can now do Ex Paper 1 Q 6 & 7**