9 More on the Gaussian elimination. LU factorization

In the previous sections I discussed the basic idea of the classical Gaussian elimination and also why it should be always preferred over very slow Cramer's method. Here I cover several further topics related to this method and look at it from a slightly more theoretical point of view.

Recall that in the Gaussian elimination with back substitution we perform two steps: first, forward step, which turns $\boldsymbol{A}$ into an upper triangular matrix $\boldsymbol{U}$:

$$[\boldsymbol{A} \mid \boldsymbol{b}] \longrightarrow [\boldsymbol{U} \mid \boldsymbol{b}'],$$

(here $\boldsymbol{b}'$ is the modified vector $\boldsymbol{b}$, not the derivative), and the backward step, in which an upper triangular system is solved:

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}' \implies \boldsymbol{x} = \boldsymbol{U}^{-1}\boldsymbol{b}',$$

here I write $\boldsymbol{U}^{-1}$ to mean that the system is solved in consecutive way, starting with x_n, then x_{n-1}, etc. I never need to explicitly calculate the inverse of $\boldsymbol{U}$.

9.1 Determinant of $\boldsymbol{A}$ and the inverse of $\boldsymbol{A}$

In many problems I need to find $\det \boldsymbol{A}$ and the inverse $\boldsymbol{A}^{-1}$ (if it exists). Recall that for Cramer's method finding the determinant is an intermediate step in the solution, which was also the main reason for such unacceptable amount of computations. In reality, however, one finds the determinant through the process of Gaussian elimination. In particular, I remind that the whole process of reducing $\boldsymbol{A}$ to $\boldsymbol{U}$ consists of applying the same elementary operation: namely, take a row, multiply with some scalar, and add to another row. From a basic Linear Algebra course I know that this kind of elementary operation does not change the determinant of my matrix. That is, the determinants of $\boldsymbol{A}$ and $\boldsymbol{U}$ are the same. The determinant of $\boldsymbol{U}$, however, is simply the product of the diagonal terms:

$$\det \boldsymbol{A} = \det \boldsymbol{U} = \prod_{i=1}^{n} u_{ii},$$

you can prove this formula by using the standard expansion of determinant with respect to the first row of $\boldsymbol{U}$.

Another common numerical linear algebra problem is to find the inverse of $\boldsymbol{A}$ (recall that the inverse is defined as the matrix $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$ and $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$, and it exists and unique if and only if $\det \boldsymbol{A} \neq 0$). To see how $\boldsymbol{A}^{-1}$ can be found by the same algorithm, let me consider a sequence of problems with the same $\boldsymbol{A}$ but different $\boldsymbol{b}$:

$$\boldsymbol{b}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots, \quad \boldsymbol{b}_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

i.e., $\boldsymbol{b}_i$ is the vector with 1 at the i-th position and zeroes everywhere else, they are also called the standard unit vectors.

Math 488/688: Numerical Analysis
e-mail: artem.novozhilov@ndsu.edu. Fall 2021

Assume that I can solve n systems

$$\boldsymbol{A}\boldsymbol{x}_i = \boldsymbol{b}_i,$$

and find n solutions $\boldsymbol{x}_i$. I claim that the matrix composed of these vectors as columns, i.e., $[\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_n]$ is the inverse to $\boldsymbol{A}$. Indeed, consider such matrix $\boldsymbol{B} = [\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_n]$ and calculate

$$\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}[\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_n] = [\boldsymbol{b}_1 \mid \ldots \mid \boldsymbol{b}_n] = \boldsymbol{I},$$

because $\boldsymbol{b}_i$ are exactly the columns of the identity matrix. Therefore, $\boldsymbol{B} = \boldsymbol{A}^{-1}$ as required.

From the algorithmic point of view it should be clear that n systems should not be solved separately, which would require too many repeated calculations (think this out). It is much better to start with the unified forward step

$$[\boldsymbol{A} \mid \boldsymbol{I}] = [\boldsymbol{A} \mid \boldsymbol{b}_1 \mid \ldots \mid \boldsymbol{b}_n] \longrightarrow [\boldsymbol{U} \mid \boldsymbol{b}_1' \mid \ldots \mid \boldsymbol{b}_n'],$$

and after it solve n upper triangular systems to recover the columns of the inverse matrix. I invite the students to program this algorithm to make sure they understand how it works.

9.2 Gauss–Jordan elimination

In the previous section I discussed how to find the inverse matrix. It required solving n upper triangular systems as the last step. Can I improve on it? Yes. Instead of Gaussian elimination with back substitution I can consider the so-called *Gauss–Jordan elimination*, in which matrix $\boldsymbol{A}$ is turned into the identity matrix now using two types of elementary row operations: in addition to the one used in the Gaussian elimination, I will also allow multiplication of my rows by scalars. Using my notation, the Gauss–Jordan elimination is defined schematically as follows:

$$[\boldsymbol{A} \mid \boldsymbol{b}] \longrightarrow [\boldsymbol{I} \mid \boldsymbol{x}].$$

Note that in this case my procedure ends up with the vector $\boldsymbol{x}$, which must be my solution by construction.

In slightly more words, to perform Gauss–Jordan elimination, I start with dividing the first row of $[\boldsymbol{A} \mid \boldsymbol{b}]$ by a_{11}, such that the leading coefficient of the first row becomes 1, and then use the first row to eliminate all the nonzero entries in the first column (with indexes $2, \ldots, n$). After its done, I divide the second row by the current a_{22}' to have this entry to be equal to 1, and now eliminate all the nonzero entries in the second column in rows $1, 3, \ldots, n$, etc.

Example 9.1. `Add explicit example`

It seems that for solving just one system of equations Gauss–Jordan method requires somewhat more elementary operations compared to the Gauss elimination, since here we need to apply the elementary operations to all the elements of $\boldsymbol{A}$. This intuitive conclusion is true, I will leave the exact count to the students. So, is there any advantage? Yes, for finding the inverse matrix. In particular, the Gauss–Jordan elimination on $[\boldsymbol{b}_1 \mid \ldots \mid \boldsymbol{b}_n] = \boldsymbol{I}$ simultaneously schematically will look like

$$[\boldsymbol{A} \mid \boldsymbol{I}] \longrightarrow [\boldsymbol{I} \mid \boldsymbol{A}^{-1}],$$

i.e., the output of the algorithm is the inverse matrix (this method is usually discussed in introductory linear algebra classes), and there will be a (relatively small) advantage in elementary operation over the Gauss elimination (again, the details are left to the student).

9.3 Pivoting

Do the Gauss and Gauss–Jordan methods always work? Obviously no, even for nice $\boldsymbol{A}$ with $\det \boldsymbol{A} \neq 0$. Imagine, for instance, any system of linear algebraic equations with matrix $\boldsymbol{A}$ such that $a_{11} = 0$. None of the discussed methods will work because they both require division by a_{11} as one of the first steps. An obvious solution is to rearrange our equations in the way such that at the first step we would not need to divide by zero. There are multiple ways of doing this, the one which is recommended is called *pivoting* whereas we put in the position corresponding to a_{11} either the maximum in absolute value element from the first column (partial pivoting) or the maximum in absolute value element from the whole matrix $\boldsymbol{A}$ (full pivoting). The actual reasons for implementing this procedure will be made clear later.

Example 9.2. `Add explicit examples`

In general, the pivoting strategy is clear: We want to put on the main diagonal the largest in absolute value element from all the elements that will not destroy the previous computations. For instance, for partial pivoting for both Gauss and Gauss–Jordan methods assume that we worked through the first $i - 1$ columns; now it is time to work with row (and column) i. I choose now the maximal in absolute value element out of those *below* the main diagonal only, say, it is row $j > i$, and switch rows i and j before moving forward. For the full pivoting I am choosing only from the elements of the sub matrix with rows $i, \ldots, n$ and columns $i, \ldots, n$, and now, if necessary, switch both rows and columns.

There is no large theoretical advantage for doing full pivoting over the partial pivoting; it is, however, a little more computationally involved, since for the full pivoting one has to keep track for the labeling of the variables, since if one switches columns i and j for the system of linear equations it means that variables x_i and x_j switched their places. Such switching must be undone when you return your final answer. This does not happen for the partial pivoting since switching two rows does not change anything in the order of variables.

Finally, if one is performing Gauss or Gauss–Jordan algorithms with partial or full pivoting for the calculation of determinant, remember that switching any two rows or columns in a matrix implies determinant of this matrix changes its sign. In general, one has to keep track of the total number of row and/or column switches during the computation if the function must return a correct determinant.

In general, for a non singular matrix $\boldsymbol{A}$, if pivoting is implemented, both the Gauss and Gauss–Jordan methods always finish working in a finite number of steps and return an answer.

9.4 *LU* decomposition

Here I will return back to the forward step of Gaussian elimination. Once again, all I am doing in this step is to use one row, multiply by a certain constant, and add to another row. From a more theoretical point of view this corresponds to the multiplication of $\boldsymbol{A}$ by a certain matrix $\boldsymbol{P}_{ij}$ from the left. Specifically, I consider an $n \times n$ matrix

$$\boldsymbol{P}_{ij} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & \alpha & \ddots & \\ & & & 1 \end{bmatrix},$$

such that it is an almost identity matrix, with one additional element α in the position i, j, where $i > j$. I claim that if I take any $n \times n$ matrix $\boldsymbol{A}$ and consider the product $\boldsymbol{P}_{ij}\boldsymbol{A}$ then it will be the same as to take the j-th row of $\boldsymbol{A}$, multiply it by α and add to row i. I leave it as an exercise to check this claim. In other words the process $\boldsymbol{A} \longrightarrow \boldsymbol{U}$ can be written as

$$\boldsymbol{U} = \boldsymbol{P}^k \boldsymbol{P}^{k-1} \dots \boldsymbol{P}^2 \boldsymbol{P}^1 \boldsymbol{A},$$

where the indexes denote the order of applications of my elementary matrices and not powers.

I also claim that any $\boldsymbol{P}_{ij}$ is an invertible matrix. It is quite easy to realize that the inverse of $\boldsymbol{P}_{ij}$ is the matrix with $-\alpha$ at the i, j position, ones on the main diagonal, and zeros everywhere else. Indeed, to bring everything back to the original state I now need to take row j multiply it by $-\alpha$ and add to row i. Hence,

$$(\boldsymbol{P}^1)^{-1} \dots (\boldsymbol{P}^k)^{-1} \boldsymbol{U} = \boldsymbol{A}.$$

By construction all $(\boldsymbol{P}^j)^{-1}$ are unit lower triangular matrices. That is, they are lower triangular with ones on the main diagonal. Their product will be again a unit lower triangular matrix $\boldsymbol{L}$ (this statement is left for the student to prove), and finally I can formulate an important conclusion: The forward step of the Gaussian elimination is equivalent to the factorization of matrix $\boldsymbol{A}$ into the product of a unit lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}.$$

I can construct matrix $\boldsymbol{L}$ by keeping track of elementary operations performed on $\boldsymbol{A}$, but since now I now that this factorization is possible, I would like to determine $\boldsymbol{L}$ directly. As the following example shows, this can be done step by step, at least in some cases.

Example 9.3. For my explicit example I choose a 4×4 matrix. I must have

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

I will be working by the columns of matrix $\boldsymbol{A}$ from top to bottom (other approaches are also possible).

For the first column I have

$$\begin{aligned} u_{11} &= a_{11}, \\ l_{21}u_{11} &= a_{21} \implies l_{21} = \frac{a_{21}}{u_{11}}, \\ l_{31}u_{11} &= a_{31} \implies l_{31} = \frac{a_{31}}{u_{11}}, \\ l_{41}u_{11} &= a_{41} \implies l_{31} = \frac{a_{41}}{u_{11}}, \end{aligned}$$

note that I can determine all the unknowns uniquely here, using only the information from the previous steps and known matrix $\boldsymbol{A}$.

For the second column I have

$$\begin{aligned}
u_{12} &= a_{12},\\
l_{21}u_{12}+u_{22} &= a_{22} \implies u_{22}=a_{22}-l_{21}u_{12}\,,\\
l_{31}u_{12}+l_{32}u_{22} &= a_{32} \implies l_{32}=\frac{1}{u_{22}}(a_{32}-l_{31}u_{12})\,,\\
l_{41}u_{12}+l_{42}u_{22} &= a_{42} \implies l_{42}=\frac{1}{u_{22}}(a_{42}-l_{41}u_{12})\,.
\end{aligned}$$

One more column:

$$\begin{aligned}
u_{13} &= a_{13},\\
l_{21}u_{13}+u_{23} &= a_{23} \implies u_{23}=a_{23}-l_{21}u_{13}\,,\\
l_{31}u_{13}+l_{32}u_{23}+u_{33} &= a_{33} \implies u_{33}=a_{33}-l_{31}u_{13}-l_{32}u_{23}\,,\\
l_{41}u_{13}+l_{42}u_{23}+l_{43}u_{33} &= a_{43} \implies l_{43}=\frac{1}{u_{33}}(a_{43}-l_{41}u_{13}-l_{42}u_{23})\,.
\end{aligned}$$

I will leave it to the student to finish the calculation for the final fourth column, but my hope is that at this point already a pattern can be seen.

First, it is clearly possible (if all found $u_{ii}\neq 0$ at least) to determine the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$ sequentially column by column. Second, it should be noted that for each column we have two different formulas: those that do not require any division and those that actually require division. In the general form, for $n\times n$ matrix, these can be written as follows.

I denote j the column index, so for all $j=1,\dots,n$ I must have

$$\begin{aligned}
u_{ij} &= a_{ij}-\sum_{k=1}^{i-1} l_{ik}u_{kj}, \quad i=1,\dots,j,\\
l_{ij} &= \frac{1}{u_{jj}}\left(a_{ij}-\sum_{k=1}^{j-1} l_{ik}u_{kj}\right)=\frac{m_{ij}}{u_{jj}}\,, \quad i=j+1,\dots,n.
\end{aligned}$$

Here in the last formulas I use the convention that if the upper summation index is less than 1 then this sum is zero.

Another thing to note is that I actually do not have to store $\boldsymbol{A},\boldsymbol{L},\boldsymbol{U}$ separately. If one of the a_{ij} is used, it is not required again for any computation, which allows me to store all the information in the same matrix (you will need to make a copy of $\boldsymbol{A}$ if you need to use it in the future). In my notation, the $\boldsymbol{LU}$ factorization algorithm that I presented above, can be represented as schematically

$$\begin{bmatrix} a_{11} & \dots & a_{1n}\\ \vdots & \ddots & \\ \vdots & & \\ a_{n1} & \dots & a_{nn}\end{bmatrix} \to \begin{bmatrix} u_{11} & a_{12} & \dots & a_{1n}\\ l_{21} & a_{22} & & \\ \vdots & \vdots & & \\ l_{n1} & a_{n2} & \dots & a_{nn}\end{bmatrix} \to \dots \to \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n}\\ l_{21} & u_{22} & & \\ \vdots & \vdots & & \\ l_{n1} & l_{n2} & \dots & v_{nn}\end{bmatrix}.$$

How to do the pivoting for the $\boldsymbol{LU}$ factorization? Actually, only partial pivoting can be efficiently implemented. Note in the formulas above I need to divide by u_{jj}. Pivoting usually means that I divide by the largest in absolute value entry that will not destroy my previous computations. For $\boldsymbol{LU}$ factorization it works as follows: First I compute u_{jj} and all $m_{ij}, i = j+1, \ldots, n$ (see above the definition of m_{ij}). Next I choose the largest in absolute value element out of them and put it on the main diagonal switching two rows. Note that when I write "switching two rows" I assume that all my previous calculations are stored in the same matrix as described above, such that the elements of $\boldsymbol{L}$, which were already computed, and elements of $\boldsymbol{A}$, which were not used yet, are also exchanging the rows.

After the last step I finally divide all the entries below the main diagonal in the column j by my "new" u_{jj}. You should consider an explicit example to completely see how this procedure works.

Example 9.4. `Add an explicit example`

In short, I claim that if $\boldsymbol{A}$ is non-singular, then it is always possible to present it as

$$\boldsymbol{PA} = \boldsymbol{LU},$$

where $\boldsymbol{P}$ is the permutation matrix, which keeps track of the rows I switched because of pivoting.

To summarize the algorithm: I accept $\boldsymbol{A}$ and return $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}$. Due to the above I must have

$$\boldsymbol{LUx} = \boldsymbol{PAx} = \boldsymbol{Pb} = \boldsymbol{b}'.$$

First I solve

$$\boldsymbol{Ly} = \boldsymbol{b}'$$

for $\boldsymbol{y}$, note that it is lower triangular system, and hence can be solved very quickly.

After I solve

$$\boldsymbol{Ux} = \boldsymbol{y},$$

which is upper triangular, we already know how to solve it.

This may look like an unnecessary complication, but careful counting shows that the number of elementary operations is approximately the same as for the Gaussian elimination. So what is an advantage of this method?

Here the important fact is that if you found $\boldsymbol{LU}$ factorization once, you can solve systems with the same $\boldsymbol{A}$ but different $\boldsymbol{b}$ very fast, which is often important to do (note that for the Gaussian elimination you would need to repeat the whole process of turning $\boldsymbol{A}$ into $\boldsymbol{U}$ again for each new right hand side). Also, matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ carry a lot of important information, which can be used.

To finish this section, out of three related methods I considered so far, $\boldsymbol{LU}$ factorization is the one, which can be used even in professional software to solve systems of linear algebraic equations, and which usually yields a reliable result for sufficiently small ($n < 50$) and "nice" (which we will discuss later) matrices.