Due date: Maple: Monday, March 21, at the beginning of the lab
C++/Java: Monday, March 21, at the beginning of the lab.

1 What this Lab is About

Our plans for today are

- To implement an efficient algorithm for performing the \( L \) \( U \) factorization of a matrix \( A \);
- To apply our knowledge and tools to the resolution of a simple practical problem.

2 Pivoting

2.1 When to pivot

So far, we have carefully avoided the pivoting issue. What do we do when the term \( u_{jj} \) is too small? We swap rows, as usual? But this bring up two questions:

1. Since swapping rows is a very fast operation, why not do it all the time, and always choose the largest pivot (in absolute value)?

2. At what point in the algorithm can we compare the pivots and swap rows?

The answer to Question 1 is: Indeed, why not? This is exactly what we will do. The second question is somewhat trickier, at first. We will address it in the next subsections.

2.2 Choice of a pivot

When we look carefully in the algorithm without pivoting at the expression for \( u_{jj} \) and that of the terms \( l_{ij} \) we see that it is identical, except for the \( u_{jj} \) scaling factor. In other words, we can do the following

1. For \( i = j \ldots n \), compute \( h_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} a_{kj} \). Here I use the notation \( h_{ij} \) to reflect the fact that before we have chosen a pivot, the terms calculated are not terms of the \( L \) or \( U \) matrices yet. Not only because we have not scaled them yet to give us \( l_{ij} \), but because they are all still candidates to be the bivot for that column, that is \( u_{jj} \).

2. Decide which of the above terms (say, the one at row \( i_{\text{max}} \) would constitute the best pivot.
3. If $i_{\text{max}} \neq j$, swap rows $i_{\text{max}}$ and $j$ (this defines the new pivot $u_{j,j}$).

4. For all rows $i = j + 1 \ldots n$ after the row swap, divide row elements $lu_{ij}$ computed in Step 1 by the new pivot $u_{j,j}$ to obtain $l_{ij}$.

We will discuss implementation issues regarding Step 3 (row swapping) in Subsection 2.3, but for the moment, let us consider more carefully Step 2. Obviously, we need to swap rows when the candidate pivot computed at Step 1, $lu_{j,j}$ is null or very small. To pick a new pivot, we have two possible choices: “first fit” or “best fit.” The first strategy consists in selecting the first element computed in Step 1 whose absolute value is larger than some predefined threshold. The second strategy consists in estimating a “quality” criterion for each pivot candidate and selecting the one with the highest score.

A naive criterion is simply the absolute value $|lu_{ij}|$: Select the candidate pivot with the largest absolute value. We saw in class why this is a bad strategy. What matters is not the absolute size of the pivot but its size relative to the rest of its row. This is why we will use the value $|lu_{ij}| \max_{k=1\ldots n} |a_{i,k}|$ for $i = j \ldots n$ as our criterion for selecting a pivot for column $j$: We select the candidate pivot that is the largest relative to the original row in matrix $A$.

This leads to the following algorithm:

```c
// for all rows of matrix A
for j from 1 to n do
    // for all columns of matrix A
    for j from 1 to n do
        // for all elements above the diagonal
        for i from 1 to j - 1 do
            compute $u_{i,j} = a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} u_{k,j}$
        // for all elements on or below the diagonal
        for i from j to n do
            compute $lu_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} u_{k,j}$
        determine $i_{\text{max}}$ such that $|lu_{i_{\text{max}},j}| = \max_{i=j\ldots n} |lu_{i,j}| / v_i$
        if $i_{\text{max}} \neq j$ swap rows $j$ and $i_{\text{max}}$ in $A$, $L$, $U$ (set $u_{j,j} = lu_{i_{\text{max}},j}$)
        compute $s = 1 / u_{j,j}$ (if $u_{j,j} = 0$, set $s = 0$)
        // for all elements below the diagonal
        for i from $j + 1$ to n do
            $l_{i,j} = s lu_{i,j}$
```

2.3 How to swap rows

Regarding the swapping operation, we must be careful about one thing: we are (or could be) performing the factorization of $A$ before we know the right hand term $b$. Therefore, we cannot simply swap rows. We have to keep track of the swaps we perform so that we can apply them to the right hand term when we try to solve the SLE.

The simplest way to do this is to maintain an array of row indexes. In other words, we replace in our algorithms any row index $i$ by $p_i$. Initially, $p = \{1, 2, \ldots, n\}$. Each time the algorithm calls for a row swap, we simply invert the corresponding elements of $p$. At the end of the factorization, $p$ represents the permutation that has been applied to $A$ and must now be applied to any right hand term $b$ of an SLE we want to solve. So now, instead of manipulating element $L_i U_j$, we will work with $L_{p_i j}$, or if you prefer to write this in “array” notation, $LU[p[i]][j]$.

Now, where you have to be very careful is that the way you apply this row permutation depends on the way you decided to store your matrices. If you overwrite your $A$ matrix with the terms of the $L$ and $U$ matrices, then you should always use $s[i]$ instead of $i$ as the first index whenever you access the matrix. You should of course understand that this permutation vector should be sent back as a result of the factorization since you will need it in your substitution modules.

Maple 1. Modify your LU factorization procedure to include row swapping. Your algorithm is now complete.

3 Use the LU Factorization

3.1 Solve a general SLE

This is a pretty straightforward application of the factorization, merely an integration of all your procedures into a general “solve SLE” Maple procedure that takes as parameters a square $n \times n$ matrix $A$ and an $n \times 1$ column vector $b$, and solve the SLE $A \mathbf{x} = b$.

Obviously, swapping rows of your $L$ and $U$ matrices has effects on their product, and on the right-side term of your SLE if you want the new SLE to be equivalent to the original one.

Maple 2. Modify your backsubstitution and forward substitution algorithms to take into account the permutations applied during the factorization process.

Maple 3. Implement a Maple procedure that takes as parameters a matrix $A$ and a column vector $b$ and returns the solution to the SLE $A \mathbf{x} = b$.

3.2 Compute the determinant of a matrix

This may come as a surprise to you, but the most efficient way to compute the determinant of a matrix is to compute its LU factorization first, rather than apply the recurrent expression you learned in your Linear Algebra course.

I will not explain how to get the determinant once you have the LU factorization: This one is almost embarrassingly easy. The only thing you have to be careful about is the sign of the determinant: each time you swap two lines, the sign of the determinant is inverted. In your C++/Java
implementation, the simplest way to handle that is to use a local variable that is initially set to 1 and changes sign each time two rows are swapped.

3.3 Solve an SLE with multiple right side terms

If you have to solve multiple SLEs with the same matrix but \( k \) different right side terms \( b_1, b_2, \ldots, b_k \), you could naturally factorize the matrix, and then apply the forward and backward substitution procedures for each of the right side terms.

Alternatively, you could build the matrix \( B = (b_1, b_2, \ldots, b_k) \) and apply a slightly modified version of your backsubstitution and forward substitution modules to solve the \( k \) SLEs in one pass.

Maple 4. Modify your your backsubstitution and forward substitution modules so that you can now accept a matrix as the right side term of an SLE.

3.4 Compute the inverse of a matrix

Having completed the previous task, it now becomes trivial to compute the inverse of a matrix. In fact it is so simple that this is the standard way to inverse a matrix when we have to do it (which is not very common).

Maple 5. Write a procedure that inverses a nonsingular square matrix.

3.5 Compute the condition number of a matrix [extra credit]

An easy 4 points of extra credit here. Of course, for this I want both condition numbers (for the \( N_1 \) and \( N_\infty \) norms) and the C++/Java implementation.

4 A test application

This little application will give you an opportunity to reuse your Function1D classes from Lab 02.

4.1 Passing a curve though data points

Imagine that you have a set of \( N \) data points of the form \( (x_i, y_i) \), for \( i = 1 \ldots n \). The number of points is at least 2, but could be arbitrarily large. The range of the \( x \) and \( y \) coordinates of these points is arbitrary. The only constraint they must respect is that no two points have the same \( x \) coordinate (i.e. stand one above the other). Figure 1 gives an example of such a data set.

What you would like to do or, rather, what I would like you to do, is find a function whose graph goes through all these points. Obviously, there is an infinity of such functions. Figure 2 shows two such functions.

If there is an infinity of solution, which one should we pick? first, there is no clear “best” solution, although some solutions are clearly better than other. For example, in Figure 2, the smooth graph is more pleasing to the eye (and more pleasing to any computational module that may need to evaluate the derivative of the function), but it is not very faithful to the data: It adds
peaks and valleys that were not in the initial data. Ideally, you would like a bit of both, but this not always an easy objective.

4.2 Interpolation by polynomials — Lagrange polynomials

We have already used in the past a polynomial approximation to a function: This is exactly what the jet of a function \( f \) at \( x_0 \) (the truncated Taylor expansion) is. The problem with this particular approximation is that it requires that the value taken by the \( n \) first derivatives of \( f \) at \( x_0 \) be known. Still, the basic idea remains that for most “well-behaved” functions \( f \) in the neighborhood of a point \( x_0 \) we can write

\[
    f(x) \approx a_0 + a_1 x + a_2 x^2 + \cdots + a_m x_m.
\]

The question is: Can we compute good values for the \( a_i \) without having to estimate the successive derivatives of \( f \) at \( x_0 \)? Before that, I would like you answer the following question:

**Report 1.** Does there (should there be) any relation between the number of data points and the degree of the polynomial?
Lagrange polynomials
The idea behind this particular form of polynomials (there are other forms, but this one will do as
an introduction) is to write our desired polynomial function $p_n(x)$ as

$$p_n(x) = \sum_{i=0}^{n} L_i(x) y_i$$

How do we find the $L_i(x)$? We want to select them so that they verify

$$L_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that the following functions satisfy the above constraints:

$$L_i(x) = \prod_{j=0 \atop j \neq i}^{n} \left( \frac{x - x_j}{x_i - x_j} \right).$$

**Maple 6.** Implement a Maple procedure that determines the Lagrange polynomial corresponding
to $n + 1$ data points.

### 4.3 Interpolation by polynomials as an SLE problem

Now that you have seen a method that relies only on the toolbox of calculus, we are going to solve
the polynomial interpolation problem using the tools we have developed for solving SLEs. By this,
I want to show you how closely calculus and linear algebra are related, and how the same problem
can be solved in two completely different ways. When you are confronted with real scientific
computing problem, keep in mind that there are always multiple ways of solving that problem.
Sometimes we will select a particular method because it is clearly superior. Other times, we will
select another method because it is “good enough” for our purposes and we happen to have already
developed software tools for it.

Let us now return to our interpolation problem. Consider a data point $(x_i, y_i), i = 1 \ldots n$ and a
polynomial function of degree $m$,

$$f : \mathbb{R} \longrightarrow \mathbb{R}$$

$$x \longmapsto a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m = \sum_{k=0}^{m} a_k x^k.$$ 

Let us write that our data point $(x_i, y_i)$ belongs to the graph of $f$:

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m = y_i$$

Now, you have most likely been brainwashed into seeing such an equation as one in which $x_i$
would be the unknown, but in this case, we know $x_i$ and $y_i$ and what we are trying to compute are
the $a_k$, which determine our function $f$. If you look again at this equation, you will see that our
data point gives us one linear equation in terms of our unknowns the $a_k$.

Let me insist on this point: the equation is not linear in terms of $x_i$ but it is definitely linear in
terms of the $a_k$. This is one important thing I expect you to remember from this lab: Being “linear”
means nothing, we should say is “linear in terms of” [a variable or group of variables].
4.4 The problem

Now we have established that each data point gives us one linear equation in terms of our unknowns, the \( a_j \). In other words, we have developed a computational model for our problem and formulated it in a way that we can now solve using classical techniques.

**Maple 7.** Implement a Maple procedure that takes as parameter a set of data points \((x_i, y_i)_{i=0...n}\) and returns the coefficients of the polynomial function of lowest degree whose graph contains all the data points.

4.5 Comment on the problem and the solutions

**Report 2.** What do you think of the two methods that you have implemented. In your opinion, is one of them superior to the other? Does the answer depend on the size of the data set? Beyond that, what do you think of the soundness of polynomial approximation for very large data sets?

5 Evaluation

5.1 What to hand in

Please read this part carefully. You will be penalized (see next subsection) if you don’t follow the rules and the TA ends up wasting time converting files just to be able to evaluate your work.

**A:** You should upload the following (the procedure to follow will be clarified by the end of this week):

1. Next week: the file of a well-commented Maple worksheet. Please make

2. Two weeks from now: A complete, cleaned-up CodeWarrior, gcc (mingw or cygwin), or Eclipse project folder.

   - Complete means that everything needed to compile and execute the project is there: source files, project file or make file, header files (.h files) and precompiled header source files (.pch files) if you program in C++. Of course, your test data files should be there as well, but the TA might use other data sets to test your code.

   - Cleaned-up means that files resulting from the compilation should be removed: .exe files, .sym files, .o object files (in C++), .class files (in Java), precompiled header binary files (if in C++). If you use CodeWarrior, the entire `<name of project>data` folder should be deleted as well (close the project before you throw away this folder).

3. Two weeks from now: Your report as a Word, HTML, or Acrobat (.pdf) document.

**B:** You should hand in printouts of your (complete) Maple worksheet and of your report on the day the corresponding part of the assignment is due.
5.2 Point distribution

The maximum number of points is 100, but extra points could be awarded for excellent aspects of the project or report. The point distribution for this assignment is as follows:

**Maple Modelling**
- Accomplishes what was demanded: 20 pts
- Comments and analysis: 10 pts

**C++/Java Code**
- Accomplishes what was demanded: 20 pts
- Good class design: 10 pts
- General quality & readability: 10 pts

**Report**
- Discussion and analysis of the results: 20 pts
- General quality of the writing and presentation: 10 pts

5.3 Various point penalties

Hopefully we won’t have to apply many of these:

- Project folder incomplete or not properly cleaned up: -5 pts
- Report file missing from the project folder: -5 pts
- Maple file missing: 0 for that part

**Late penalties**
- Printed copy of the report, 1 day late: -5 pts
- Project folder (uploaded to EnVision server), per day late: -5%

If you submit a project late, then it is your responsibility to notify the TA (with CC. to me) that the project is finally available for download on the EnVision server. If you fail to do so, then the “late penalty clock” will keep ticking until the TA gets around to checking your folder on the EnVision server and notices your project. Unless asked explicitly to do so, do not mail your project folder as an attachment.

As I explained in class, I will post sometimes partial solutions to the assignments, to make sure that nobody gets too far behind. Obviously, it will be impossible to get any point on a part after a solution for it has been posted (this does not affect the late penalty count).