NUR - Hand-in Exercise 1

Martina Cacciola (s4170814)

March 7, 2024

Abstract

In this document, I present the results for the Hand-in Exercise 1 for the course Numerical Recipes for Astrophysics.

1 Poisson distribution

The exercise is done in the script `poisson.py`. The necessary explanations are in the comments of the code. The output produced for the values required is in `poisson_output.txt`.

2 Vandermonde matrix

In this section, we add the comments to the plots produced by the script given by: `vandermonde.py` . For question (a), we are performing a polynomial interpolation on a set of data points using a 19th-degree polynomial, evaluated at ≈ 1000 equally-spaced points. The coefficients of the polynomial are determined by solving a system of linear equations using LU decomposition. We also plot the absolute difference between the given points y_i and our result $y(x)$, i.e. $|y(x) - y_i|$.

From Fig.1, it seems that the polynomial is fitting the data points quite well for most of the range. This is expected as a polynomial of degree (n) can always fit $(n + 1)$ data points exactly. However, towards the end (around $x = 100$), the polynomial shoots up drastically. This is a common issue with high-degree polynomial interpolation known as Runge's phenomenon. It is a form of overfitting where the polynomial oscillates significantly at the boundaries of the data set.

The choice of a 19th-degree polynomial for this data might not be the best. While it fits the given data points well, the extreme behavior at the boundaries suggests that it might not generalize well to other data. A lower-degree polynomial or a different type of function might provide a better fit without the extreme oscillations.

The bottom part of the plot shows the absolute difference between the given y_i values and the calculated $y(x)$ values. This represents the error in the polynomial fit at the data points. Since the polynomial fits the data points exactly, these errors are close to zero, going from order 10^{-14} to 10^{-2}. We can note how the error increases at the boundaries, consistently with the issue presented before.

For question (b), we implement Neville's algorithm (see Fig. 2).It works by recursively evaluating a set of polynomials and combining them to form the final interpolating polynomial. Regarding the interpolation, we are getting the same results as LU decomposition, so they have same efficiency in fitting the given data points. The errors are drastically small for Neville's (reaching a maximum of 10^{-14}) while LU decomposition have errors slightly higher. We could explain this addressing to the nature of the algorithm, which tends to be stable and accurate, especially when the data points are equally spaced. The accuracy of LU decomposition for interpolation can be influenced by the condition of the matrix involved. Ill-conditioned matrices may lead to numerical instability and higher errors. In our case, the condition number of the Vandermonde matrix depends on the arrangement and spacing of the data points. If the data points are well-spaced and not too close to each other, the condition number may be reasonable. However, as the data points become closely spaced or nearly collinear, the problem

of ill-conditioning can arise.

For question (c), we improve iteratively the solution found by LU decomposition. We do such procedure for 1 and 10 iterations (Fig. 3). The interpolation is the same for both cases, since we have overlapping fits. Regarding the trend for the error, the two procedures get similar results, with the LU with 10 iterations having slightly lower errors in some regions of the data.

For question (d), we obtain the execution times for LU decomposition, Neville's algorithm, and LU with iterative improvement.

- **LU Decomposition Time**: 0.012404 seconds. The algorithm is known for its efficiency in solving linear systems of equations, and it typically has a low time complexity. This outcome indicates that LU decomposition is well-suited for our data set.
- **Neville's Algorithm Time**: 9.874374 seconds. The algorithm involves several polynomial evaluations and recursive calculations: the complexity of these calculations can lead to longer execution times, especially as the number of data points increases.
- **Iterative Improvement Time**: 0.117441 seconds. It is refining a solution through iterative steps, in a relatively fast process. The efficiency of this process can depend on the convergence behavior of the iterative method and the initial guess.

From these results, the LU decomposition method appears to be the most efficient in terms of computational time. If we take the speed as a critical factor and the accuracy achieved by LU decomposition is sufficient for your application, then LU decomposition may be the preferred choice. In our case, the accuracy achieved by LU decomposition is sufficient for our application. However, we have to keep in mind that LU decomposition, while efficient, might be sensitive to ill-conditioned matrices, potentially affecting accuracy in certain cases. In general, accuracy depends on the interpolation requirements and the nature of your data. However, Neville's algorithm is generally considered accurate, especially for well-behaved, evenly spaced data. In our implementation, it is behaving considerably slower than the other, so it might not be the suitable option for this particular setting, even though it is reaching the best performance in terms of lowest errors in the interpolation.

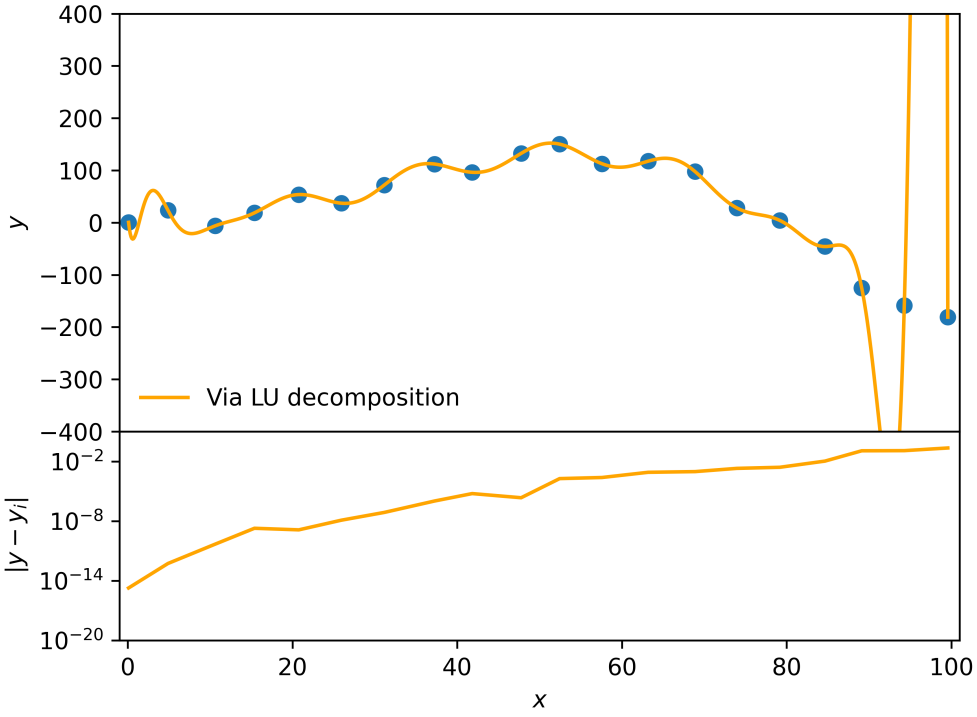

Figure 1: Upper panel: Interpolation on a set of given data points via LU decomposition. The fit is going through all the data points exactly. Nevertheless, it has an oscillating behaviour in correspondence of the last two points. Bottom panel: Absolute difference between the given points y_i and our result $y(x)$, i.e. $|y(x) - y_i|$. The error holds at values close to zero, with a small increase towards the boundaries.

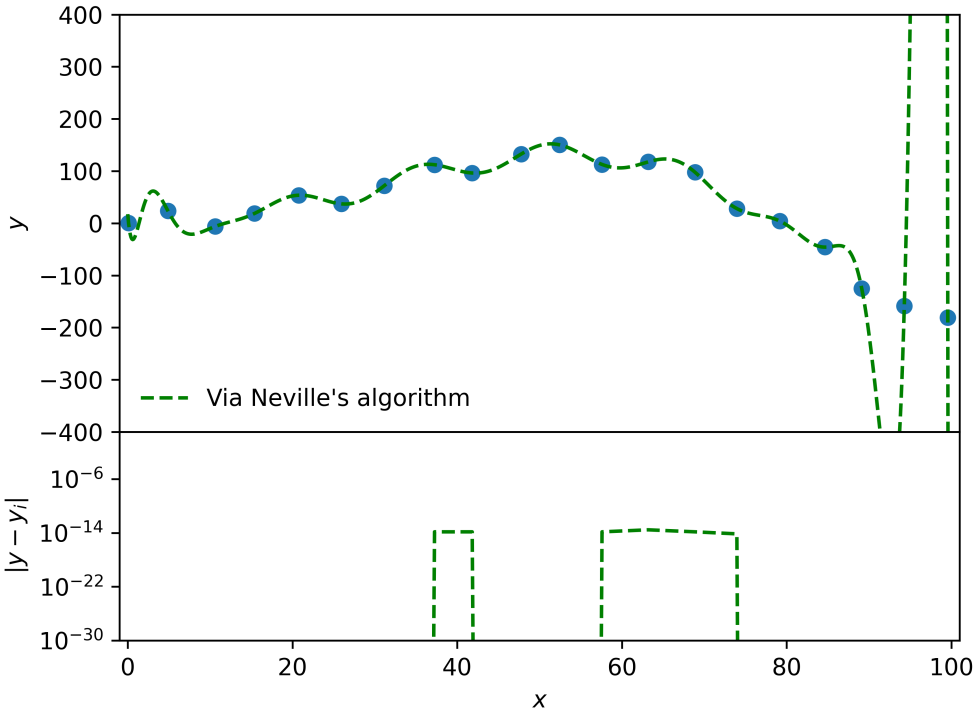

Figure 2: Upper panel: Interpolation on a set of given data points via Neville's algorithm. The polynomial obtained fits the data points well in the range considered, in the same way the interpolation of LU decomposition does. Bottom panel: Absolute difference between the given points y_i and our result $y(x)$, i.e. $|y(x) - y_i|$. The errors of Neville's algorithm reach very small values with respect to LU decomposition's ones.

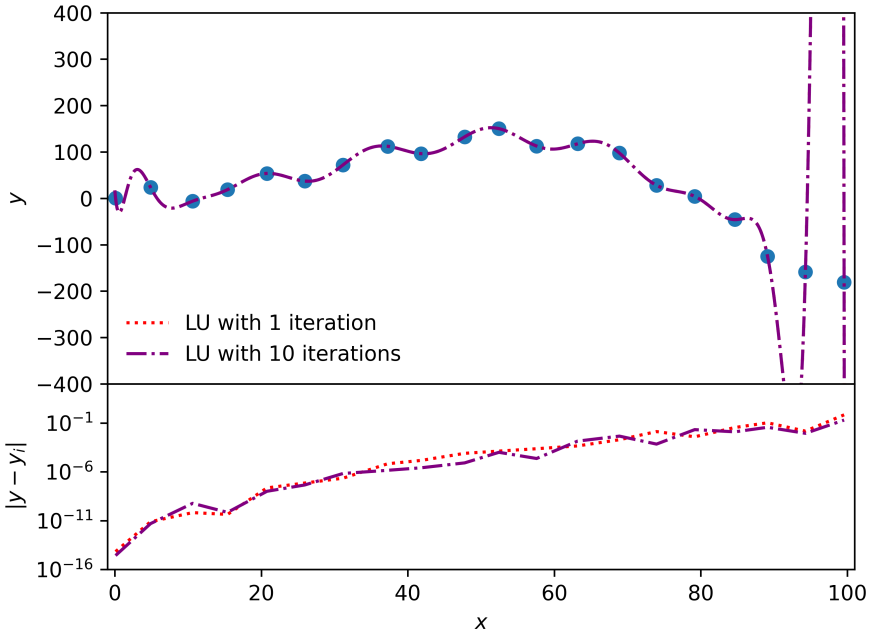

Figure 3: Upper panel: Interpolation on a set of given data points via LU decomposition, with iterative improvement of the found solution (done with 1 and 10 iterations). Both the fits are going through all the data points exactly, hence the overlap. Bottom panel: Absolute difference between the given points y_i and our result $y(x)$, i.e. $|y(x) - y_i|$. The trend is similar for the two implementations, with a small decrease in values reached by LU with 10 iterations.