Numerical Algorithms - Assignment 2

Sebastiano Smaniotto 857744

December 2020

Introduction

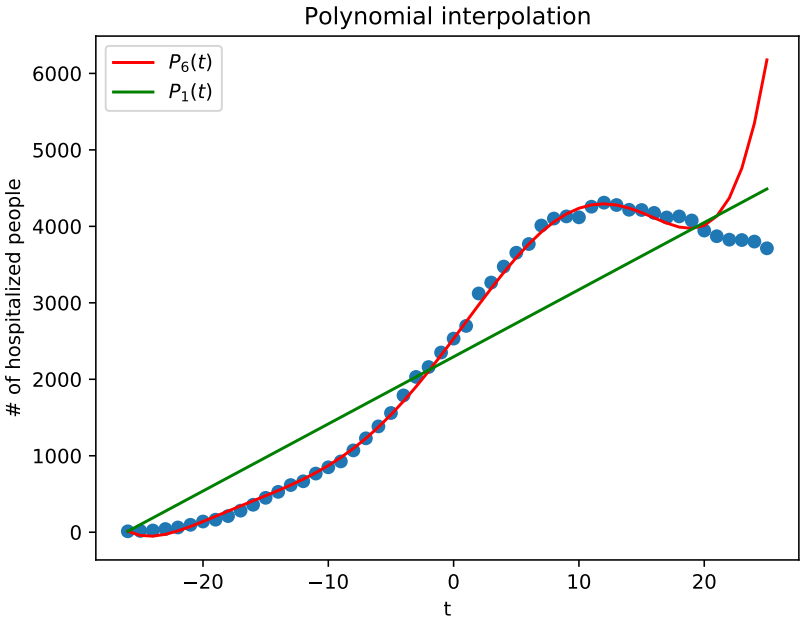
In this report we perform data extrapolation on the epidemiological data related to the Italian pandemic of covid-19. In particular, we will focus on the number of hospitalised individuals in Emilia Romagna between 24/02/2020 and 15/04/2020. We indicate the data as (t_i, y_i), where t_i is the i-th day and y_i is the number of hospitalised patient in that day.

To perform this task, we split the data in two parts: consider $N_e = 6$ extrapolation points. Then we will use the subset of data (t_i, y_i) with $i = 1, \dots, N - N_e$ to compute the interpolation/approximation function, and we call $N^{int} = N - N_e = 46$, and that with $i = N - N_e + 1, \dots, N$ as the N_e points to evaluate the error of the interpolation/approximation function during extrapolation.

In order to carry out the assignment we have used the programming language Python. To make sure that there are no compatibility problem, I have put the file `numalg.yaml` in the submission folder which contains the conda environment that we have used for the computation. In the spirit of the assignment, we have implemented from scratch all methods and functions that were required as intended.

1 Interpolant Functions

We were tasked to perform the interpolation of the data using two different family of functions: polynomial interpolant functions and cubic spline functions. Since the use of floating point conversion of date format on numpy function `numpt.polyfit()` gives rise to numerical errors, we have decided to map the time interval on the integer interval $[-N/2, \dots, N/2]$. The results are reported in Table 1. The required plots are collected in Figure 1.



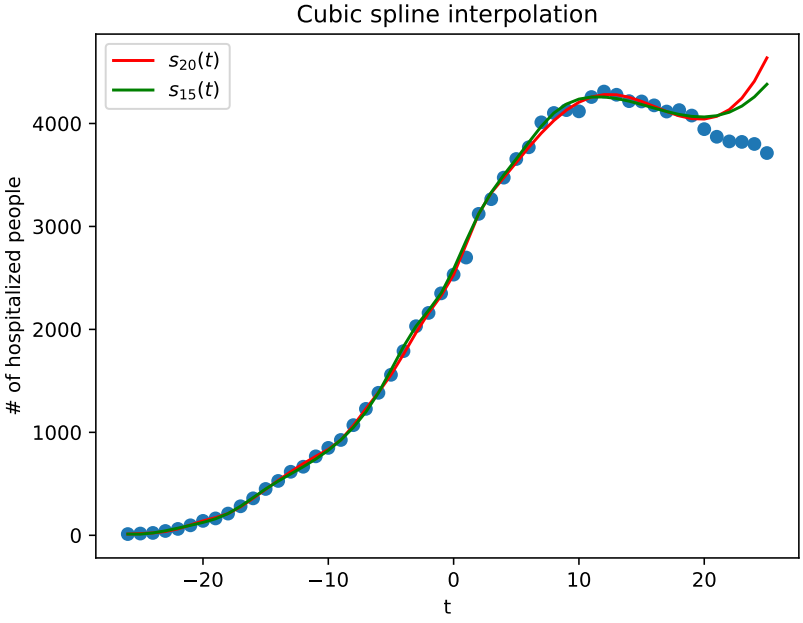


Figure 1: In these two panels we show the two models which performs better in terms of error minimization on the interpolation nodes (in red) and on extrapolation nodes (green). The results are in accordance with those in Table 1.

	Polynomial interpolant				Cubic spline			
n	e_n^{int}	e_n^{ext}	$\min_i P_n(t_i)$	$\max_i P_n(t_i)$	ee_n^{sn}	ev_n^{sn}	$\min_i s_n(t_i)$	$\max_i s_n(t_i)$
1	2.2259e-01	1.2819e-01	1.2000e+01	4.4889e+03	–	–	–	–
2	4.5436e-01	1.4973e+00	-1.1556e-01	1.0543e+04	–	–	–	–
3	2.5056e-01	8.6315e-01	1.8862e+00	7.6103e+03	2.5056e-01	8.6315e-01	1.8862e+00	7.6103e+03
4	9.4896e-02	8.4434e-01	-1.1651e+03	4.1889e+03	3.6938e-02	2.2966e-01	1.2000e+01	4.3187e+03
5	1.4904e-01	1.4326e+00	-4.5536e+03	4.3302e+03	7.3805e-02	6.8182e-01	-1.8865e+02	4.3100e+03
6	2.2755e-02	3.3218e-01	-5.1079e+01	6.1769e+03	8.4959e-02	8.0437e-01	-8.8762e+02	4.3144e+03
7	1.9560e-01	3.5266e+00	-1.9077e+04	4.3627e+03	6.2894e-02	5.7233e-01	7.9838e-01	4.3073e+03
8	1.9700e-01	4.3656e+00	1.2000e+01	3.2362e+04	2.4939e-02	1.7305e-01	1.2000e+01	4.2464e+03
9	2.7807e+00	1.0900e+02	-4.9671e+01	7.7031e+05	4.3439e-02	5.3652e-01	1.2000e+01	6.8982e+03
10	3.4858e-01	2.3983e+01	-1.7366e+05	4.2846e+03	1.8145e-02	2.3719e-01	1.2000e+01	5.1752e+03
11	6.0837e+00	5.5394e+02	-4.1634e+06	6.8269e+03	1.9158e-02	1.3093e-01	1.2000e+01	4.3100e+03
12	5.5552e+00	6.7980e+02	-7.9205e+02	5.2313e+06	3.1518e-02	4.0040e-01	1.2000e+01	4.3232e+03
13	1.8827e+01	3.2755e+03	-1.4913e+02	2.5736e+07	3.7037e-02	6.3488e-01	-2.4862e+02	4.2865e+03
14	2.6849e+01	6.0346e+03	-2.1464e+03	4.8132e+07	5.5645e-02	1.1327e+00	-3.4287e+03	4.3022e+03
15	6.2549e+01	2.8840e+04	-2.3578e+08	3.1120e+04	1.4734e-02	1.0182e-01	1.1367e+01	4.3810e+03
16	7.9341e+01	4.7694e+04	-3.9446e+08	3.2420e+04	1.4468e-02	3.0379e-01	1.1589e+01	5.7164e+03
17	2.1242e+02	1.7906e+05	-5.3336e+04	1.4999e+09	2.7019e-02	1.0484e+00	1.1577e+01	1.0633e+04
18	1.6094e+02	2.0137e+05	-1.8383e+04	1.7067e+09	1.7092e-02	8.2532e-01	1.2000e+01	9.2718e+03
19	1.4373e+02	2.0954e+05	-1.2388e+04	1.7831e+09	1.5626e-02	7.3037e-01	1.2000e+01	8.6734e+03
20	1.3044e+03	3.3756e+06	-2.9182e+10	1.0122e+05	1.3388e-02	1.3270e-01	1.2000e+01	4.6371e+03

Table 1: Results of different interpolant functions. The smallest element of each column is highlighted in blue, and the largest one in red.

2 Approximation functions

2.1 Approximation of α using linear regression

In the second part of the assignment we were required to estimate the parameter of an exponential model and to predict what would have been the number of hospitalised people at the 15/04/2020 if no lockdown measures were put in place. After some assumptions and simplification of the SIR model, the exponential model for the number of hospitalised individuals is

$$H(t) = H(t_0)e^{\alpha(t-t_0)}, \tag{1}$$

which is governed by parameter α. It is easy to estimate α by taking the logarithm of (1)

$$\log H(t) = \log H(t_0) + \alpha(t - t_0) \tag{2}$$

and use linear regression. The mapping of the date interval is as in the previous section. If we fit the model on the interpolation data (t_i, y_i), $i = 1, \ldots, N^{int}$ we obtain a function

$$\hat{y} = \beta_0 + \beta_1(t - t_0). \tag{3}$$

Note that we are only interested in fitting the value of α, which corresponds to β_1 in the linear model. Since shifts in t will only affects β_0 we can drop $-t_0$ and simply fit

$$\hat{y} = \beta_0 + \beta_1 t. \tag{4}$$

Given these premises, we have first approximated α by using linear regression on the whole interpolation dataset and obtained $\alpha = 7.2965143$. Since $H(t_0) = y_0$, the approximation function is

$$H(t) = y_0 \cdot e^{7.2965143 \cdot (t-t_0)}. \tag{5}$$

The plot can be seen in Figure 2, where we have shifted the curve so that it starts from zero. I wish to stress about what the instructions of the assignment say: to approximate α and *not* the whole $H(t)$ by using linear regression, since $H(0)$ is already known. That is why the model fit is poor: if we fitted $\hat{y} = \beta_0 + \beta_1 t$ the line would fit the data points correctly, by using only α the line will always intercept (t_0, y_0) because $H(t_0) = y_0 \exp(\alpha(t_0 - t_0)) = y_0 \exp(\alpha \cdot 0) = y_0 \exp(0) = y_0$ for all α.

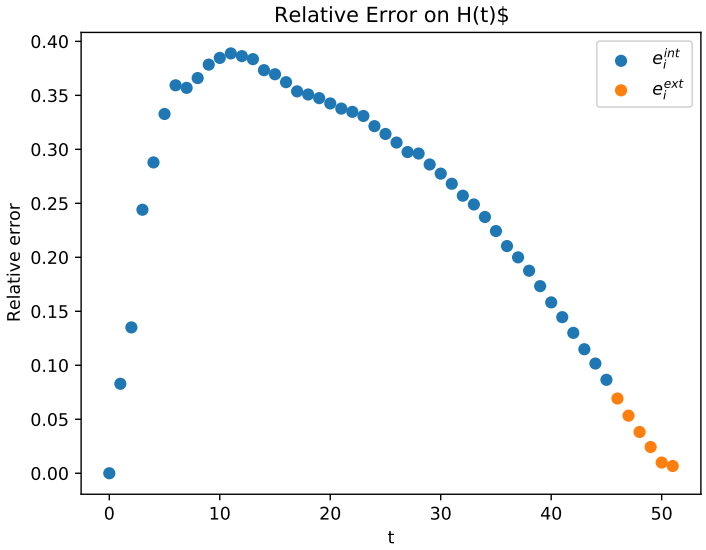


Figure 2: In this figure we can see the relative error of $H(t)$ for $\alpha = 7.2965143$.

2.2 Approximation of α on 6-day windows

It is clear that α vary as time progresses. In order to track how it changes, we compute it on windows of 6 days starting at each day, for $i = 1, \ldots, N^{int} - 6$. The evolution of α can be seen in Figure 3. As expected, the lockdown measure has made the value of α drop significantly in the first twelve days.

If we use the value of α estimated by using the last six observations of the training set, that is, $(t_i, \log y_i)$ for $i = N_e - 6, \ldots, N_e$, the estimation error is plotted on Figure 4.

2.3 What if there were no lockdown?

To answer this question we simply need to plug the estimate of α in the first six days and explicitly make the computation:

$$\begin{aligned} H_{\alpha_0}(t) &= y_0 \exp(\alpha(t_5 2 - t_0)) = 25 \cdot \exp(0.4266560048256495 \cdot (52 - 1)) \\ &= 12 \cdot \exp(21.759456246108126) \\ &= 33821512311.320892. \end{aligned} \tag{6}$$

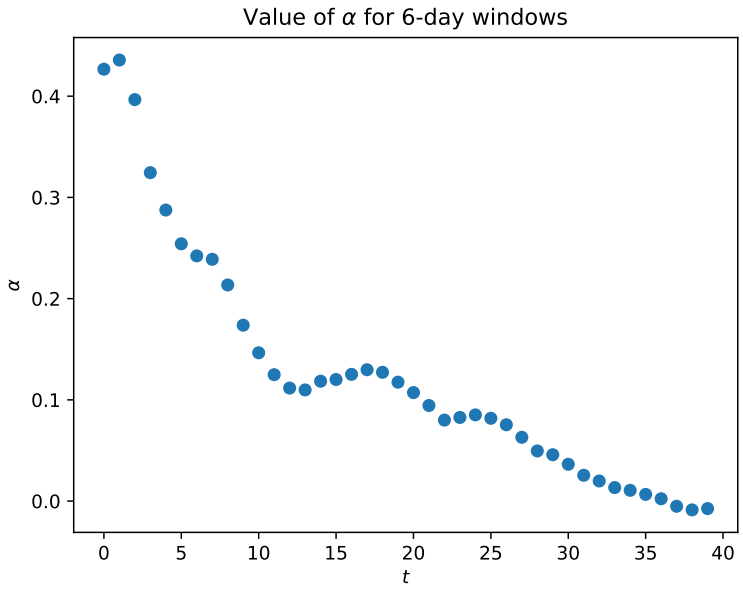


Figure 3:

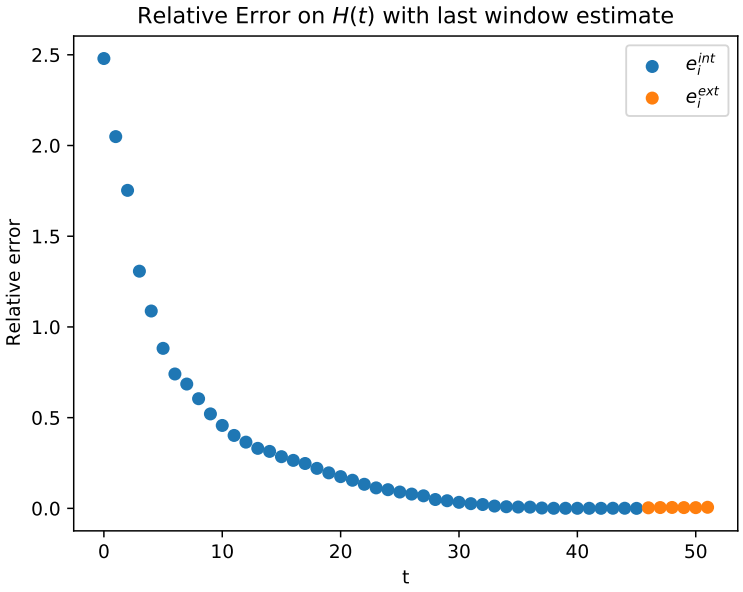


Figure 4: