Data Fitting Report

Exercise 3

Comparison between interpolating polynomial and natural cubic spline

Cesare De Cal

Professor: Annie Cuyt

Assistant Professor: Ferre Knaepkens

1 Introduction

An imaginary chemistry experiment produces the following data set:

x_i	-1	-0.96	-0.86	-0.79	0.22	0.50	0.93
f_i	-1.000	-0.151	0.894	0.986	0.895	0.500	-0.306

The goal of this exercise is to use these data points to compute and plot the interpolating polynomial together with the natural cubic spline, and to report what it is observed. In the following sections I am going to describe the computation process and list the tools I used. I will then plot the interpolating polynomial and the natural cubic spine, and draw conclusions based on the plot.

2 Tools

The following programming language and libraries have been used in this exercise:

- Python 3.7
- SciPy

The SciPy `interpolate` sub-package was used to compute the interpolating polynomial and the natural cubic spline:

- `scipy.interpolate.lagrange(x, y)`
- `scipy.interpolate.CubicSpline(x, y, type)`

The following NumPy methods of the SciPy environment have been used in this exercise:

- `numpy.array(object)`
- `numpy.linspace(start, stop, num)`
- `numpy.polynomial.polynomial.Polynomial(poly)`

The following Matplotlib methods of the SciPy environment have been used in this exercise to plot:

- `matplotlib.pyplot.plot(x, y, formatting, label)`
- `matplotlib.pyplot.legend()`
- `matplotlib.pyplot.show()`

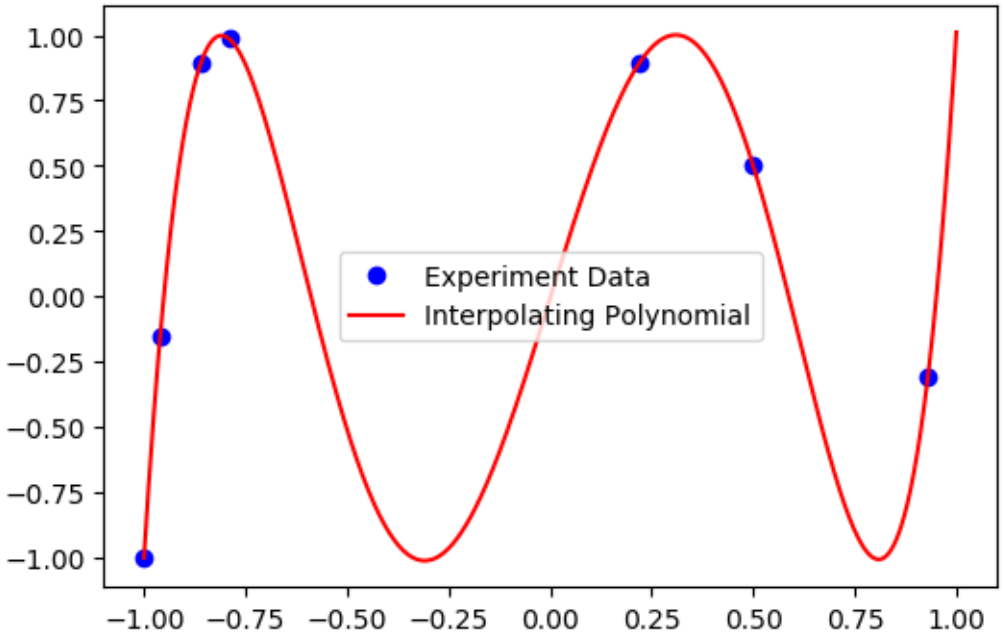
3 Computing the interpolating polynomial

The exercise asks to compute the interpolating polynomial of the given data set. To do so, I first create two arrays in Python containing the data points using `np.array` and a linear space from -1 to 1 containing 1000 points. These values are finally passed to the `lagrange` method.

The coefficients are:

- 0.04365005085
- 16.0766955610565
- -0.048402197837630
- -20.10047681678335740
- 0.0168806991536721320
- 5.029463735952404423200
- -0.006446071786954468800

The polynomial in the power form is: XXX
Here's a graph showing the 6th degree interpolating polynomial:



In order to compute this interpolating polynomial in Python, I used the `lagrange` method ... [EXPLAIN]

4 Natural Cubic Spline