Notebook

January 7, 2020

NBBinder test on a collection of notebooks about some thermodynamic properperties of water

<- Low-Dimensional Fittings | Water Contents | References | Choosing the Best Fit with AIK ->

---

1 High-Dimensional Fittings

Now we fit higher degree polynomials to the data and compare the results and errors.

1.1 Importing the libraries

```python
[1]: import csv
import numpy as np
import matplotlib.pyplot as plt
```

1.1.1 Loading the data

We load the data and define the header and the respective vectors with the temperature and with the density values.

```python
[2]: water_csv = list(csv.reader(open('water.csv',"r"), delimiter=","))
header = dict([(water_csv[0][i],water_csv[1][i]) for i in range(3)])
T, f = np.loadtxt(open('water.csv', "r"), delimiter=",", skiprows=2,
 →usecols=(0,1), unpack=True)
N = len(T)
N_half = int(N/2)
```

1.1.2 The Vandermonde matrices

We build a number of Vandermonde matrices, up to the number of data points available.

```
[3]: A = list()
     for j in range(N_half):
         A.append(np.vstack([T**i for i in range(j+1)]).T)
```

1.1.3 Solving the least-square problems

```
[4]: a = list()
     for j in range(N_half):
         a.append(np.linalg.lstsq(A[j], f, rcond=None)[0])
```

1.1.4 Building the approximating polynomials

```
[5]: p = list()
     for j in range(N_half):
         p.append(np.array(sum([a[j][i]*T**i for i in range(j+1)])))
```

1.1.5 Plotting the approximations

```
[6]: plt.figure(figsize=(10,5))
     plt.plot(T, f, 'o', label='Data', color='tab:blue')
     for j in range(N_half):
         plt.plot(T, p[j], label=f'degree {j}')
     plt.title('Plot of the data and of the polynomial approximations', fontsize=14)
     plt.xlabel(header['temp'], fontsize=12)
     plt.ylabel(header['density'], fontsize=12)
     plt.legend()
     plt.show()
```

Plot of the data and of the polynomial approximations

1.1.6 Calculating the mean quadratic errors

```
[7]: Err = list()
     for j in range(N_half):
         Err.append(np.linalg.lstsq(A[j], f, rcond=None)[1][0]/N)
         print(f'j={j}: Error={Err[j]:.2e}')
```

```
j=0: Error=1.75e-04
j=1: Error=9.22e-06
j=2: Error=1.33e-07
j=3: Error=3.16e-09
j=4: Error=3.27e-10
j=5: Error=2.64e-10
j=6: Error=2.64e-10
```

1.1.7 Plotting the mean quadratic errors

```
[8]: plt.figure(figsize=(10,5))

     plt.plot(range(len(Err)), Err, 'o', color='tab:red', markersize=10)
     plt.grid(True)
     plt.yscale('log')
     plt.ylim(10**(-10), 10**(-3))
     plt.title('Mean quadratic error in terms of the degree of the approximating␣
      ↪polynomial', fontsize=14)
```

3

```
plt.xlabel('degree', fontsize=12)
plt.ylabel('error', fontsize=12)
plt.show()
```

Mean quadratic error in terms of the degree of the approximating polynomial
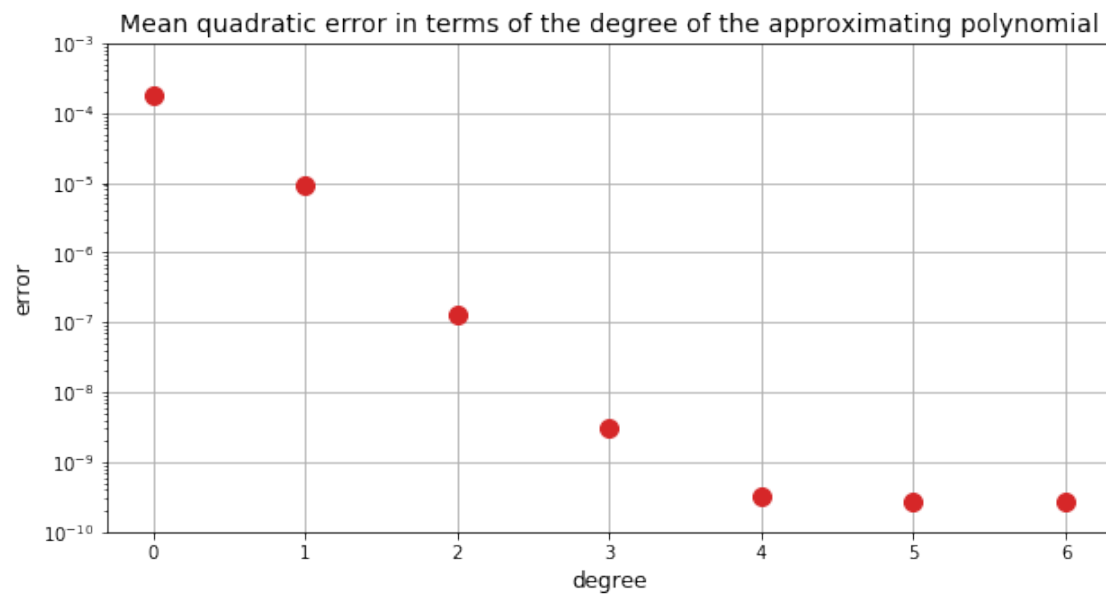
Notice how there is not much advantage going beyond degree four.

<- Low-Dimensional Fittings | Water Contents | References | Choosing the Best Fit with AIK ->