Assignment 09

March 16, 2020

1 Polynomial interpolation

Anne Kværnø (modified by André Massing and Henrik Lindell)

Date: **Mar 16, 2020**

If you want to have a nicer theme for your jupyter notebook, download the cascade stylesheet file tma4125.css and execute the next cell:

[ ]:
```
from IPython.core.display import HTML
def css_styling():
    styles = open("tma4125.css", "r").read()
    return HTML(styles)

# Comment out next line and execute this cell to restore the default notebook␣
 ↪style
css_styling()
```

And of course we want to import the required modules.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import factorial
newparams = {'figure.figsize': (8.0, 4.0), 'axes.grid': True,
             'lines.markersize': 8, 'lines.linewidth': 2,
             'font.size': 14}
plt.rcParams.update(newparams)
```

[ ]:
```
def cardinal(xdata, x):
    """
    cardinal(xdata, x):
    In: xdata, array with the nodes x_i.
        x, array or a scalar of values in which the cardinal functions are␣
 ↪evaluated.
    Return: l: a list of arrays of the cardinal functions evaluated in x.
    """
    n = len(xdata)              # Number of evaluation points x
    l = []
    for i in range(n):          # Loop over the cardinal functions
```

```
        li = np.ones(len(x))
        for j in range(n):      # Loop to make the product for l_i
            if i is not j:
                li = li*(x-xdata[j])/(xdata[i]-xdata[j])
        l.append(li)            # Append the array to the list
    return l

def lagrange(ydata, l):
    """
    lagrange(ydata, l):
    In: ydata, array of the y-values of the interpolation points.
         l, a list of the cardinal functions, given by cardinal(xdata, x)
    Return: An array with the interpolation polynomial.
    """
    poly = 0
    for i in range(len(ydata)):
        poly = poly + ydata[i]*l[i]
    return poly
```

2 Obligatory exercises

2.1 1)

Consider the data points

x_i	-1	-1/2	1/2	1
$f(x_i)$	-1/2	-5/4	1/4	5/2

a) Find the polynomial of minimal degree interpolating at these points. Express it in the form

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0.$$

Use this to find an approximation to $f(0)$.

b) Confirm your results numerically, e.g. using the functions above.

2.2 2)

The population of Norway in the period from 1993 to 2018 is, according to SSB

year	1993	1998	2003	2008	2013	2018
population	4299167	4417599	4552252	4737171	5051275	5295619

Use the interpolation polynomial, through the functions cardinal and lagrange, to estimate the population in the years 2000 and 2010. Predict the population in 2025 and 2030. Comment on the

results. (The population in 2000 was 4478497, in 2010 it was 4858199).

2.3 3)

Consider the function $f(x) = x^2\cos(x)$. **a)** Find the polynomial of degree 3 that interpolates $f(x)$ at four - equally distributed nodes - Chebyshev points

in the interval [-1,2].

b) Find by hand a bound for the maximal interpolation error in that interval in these two cases.

c) Confirm your tests numerically. More specifically, plot the function and the interpolation polynomial, measure the interpolation error and compare it with your theoretical results.

d) Repeat the experiments numerically with $n+1$ interpolation points, using $n = 5, 10, 15, 20$. In this case, we are only interested in the measured maximum error and the error bound.

Hint:

$$\frac{d^n}{dx^n}x^2\cos(x) = \begin{cases} (-1)^{n/2}(x^2\cos(x) + 2nx\sin(x) - n(n-1)\cos(x)) & n \text{ even.} \\ (-1)^{(n+1)/2}(x^2\sin(x) - 2nx\cos(x) - n(n-1)\sin(x)) & n \text{ odd.} \end{cases}$$

3 Recommended exercises

3.1 4)

Inverse interpolation. Given a sufficiently continuous function $f(x)$ with a root r in some interval $[a, b]$. Choose this sufficiently small to ensure f to be strict monotonically increasing or decreasing in $[a, b]$. We will use the fact that

$$f(r) = 0 \quad \Rightarrow \quad r = f^{-1}(0).$$

Choose $n+1$ disctinct data points (x_i, y_i) in the interval $[a, b]$, and find the interpolation polynomial $p_n(y) \approx f^{-1}(y)$ and $r \approx p_n(0)$.

a) Let $f(x) = x^3 - 7$ and $[a, b] = [1.5, 2.0]$. As nodes, choose $(x_i) = (1.5, 1.75, 2.0)$ and use the idea above to find an approximation to the root. How close to the exact solution is the approximation?

b) Repeat the example above, but now with $n+1$ uniformly distributed nodes over the interval $[1.5, 2.0]$. Use the functions **cardical** and **lagrange** to calcualte the value of the polynomial $p_n(0)$. Choose $n = 2$ (to check your hand calculations), 4, 8, and 16. Find the approximation in each case, as well as the error.

NB! (Python): Even if you only want to calculate the calue of the cardinal functions in one point, 0, it still has to be given as an array (array([0])) in thew function **cardinal**.