Physik auf dem Computer SS 2014

Worksheet 3: Taylor Series and Interpolating Polynomial

April 23, 2014

General Remarks

- The deadline for the worksheets is **Wednesday, 30 April 2014, 10:00** for the tutorial on Friday and **Friday, 2 May 2014, 10:00** for the tutorials on Tuesday and Wednesday.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Johannes (`zeman@icp.uni-stuttgart.de`; Tuesday, 9:45–11:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Shervin (`shervin@icp.uni-stuttgart.de`; Friday, 11:30–13:00)

Throughout the worksheet, the following functions are used on the specified domains:

Name	**Definition**	**Domain**
Sine Function	$f(x) = \sin x$	$[0, 2\pi]$
Runge Function	$g(x) = \frac{1}{1+x^2}$	$[-5, 5]$
Lennard-Jones Function	$h(x) = x^{-12} - x^{-6}$	$[1, 5]$

In this worksheet, different *callable* objects are used. It is possible to call an instance of the object like a function, if the class defines a method `__call__(self,args)`:

```
>>> class HelloWorld:
...   def __call__(self, who):
...     print "Hello", who, "!"
>>> hello = HelloWorld()
>>> hello("Olaf")
Hello Olaf !
```

Task 3.1: Taylor Polynomials (5 points)

In this task, you are entitled to plot the Taylor polynomials of the different functions.

- **3.1.1** (3 points) Calculate the coefficients of the truncated Taylor series of the sine function $f(x)$ at $x_0 = 0$, the Runge function $g(x)$ at $x_0 = 0$ and the Lennard-Jones function at $x_0 = 1$ up to 6th degree. Use the Python class `numpy.poly1d` to define the k-th order Taylor polynomials of $f(x)$, $g(x)$ and $h(x)$ for $k \in \{3, 5, 10\}$ at arbitrary x.
- **3.1.2** (2 points) For each of the functions $f(x)$, $g(x)$ and $h(x)$, create a plot that shows the function and their respective k-th degree Taylor polynomials ($k \in \{3, 5, 10\}$) on the specifed domain.

Hints

- You may use Mathematica or Wolfram alpha to compute the Taylor series.
- The Python class `numpy.poly1d` is used as follows:

```
# f is the polynomial f(x) = 3*x**2 + 2*x + 1
# Note the order of the coefficients!
f = numpy.poly1d([3,2,1])
# Compute the value of the polynomial at x=42
print f(42)
```

- Take care to handle the argument of the Taylor polynomial of $h(x)$ correctly!

Task 3.2: Interpolating Polynomials (5 points)

Now you should plot interpolating polynomials of the three functions.

- **3.2.1** (2 points) For each of the functions $f(x)$, $g(x)$ and $h(x)$, create a plot over the specified domain that shows the function and its k-th degree interpolating polynomials ($(k \in \{3, 5, 10\}$) for equidistant supporting points. Use the Python function `scipy.interpolate.lagrange` to compute the interpolating polynomial.
- **3.2.2** (2 points) Write a Python program that does the same as the script from the previous task. However, this time you shall implement your own class `NewtonInterpolation` that does the interpolation using Newton's representation. The class shall define two methods:
  - `__init__(self,x,y)` does the same as `neville()` from the lecture script and stores the x and γ in class variables
  - `__call__(self,x)` does the same as `horner()` from the lecture script but does not require the parameters `gamma` and `x` but uses the stored class variables instead
- **3.2.3** (1 point) Modify the Python program from any of the previous tasks such that it computes the interpolating polynomials at the Chebyshev nodes and create the same plots as in the previous tasks.