Numerical Methods (CS 357)
Worksheet

Part 1. Objectives

- Interpret behavior of functions by their Fourier components
- Understand how to use interpolants to numerically take derivatives of functions
- Understand what limits the accuracy of numerical differentiation
- Understand how to derive quadrature rules

Part 2. Splines: Counting conditions

Suppose we would like to construct a piecewise polynomial interpolant consisting of two pieces, one with a quadratic polynomial, and one with a cubic.

How many conditions do we need in addition to the \( f(x_i) = y_i \) interpolation conditions?

Part 3. Differentiation: Error prediction

Suppose we numerically take a derivative of a function \( f(x) \) using monomials at four equally-spaced points on an interval of length \( h \). The obtained interpolation error is found to be 0.1. What interpolation error do you predict for the same function on a subinterval of the original one that has length \( h/2 \)?

Part 4. Numerical Differentiation

You are given nodes and data for an interpolant. Evaluate and plot its derivative.

INPUT:

- \( x \), a 4-vector of \( x \) coordinates
- \( f_x \), a 4-vector of function values \( f(x) \)

OUTPUT:
• \( \mathbf{f}_p \cdot \mathbf{x} \), a 4-vector of point values of the \textit{derivative} of the polynomial interpolant for the data \( \mathbf{f}_x \) at the nodes \( \mathbf{x} \)

• \texttt{coeffs}, the coefficients \((\alpha_i)\) for the derivative expressed as a linear combination of monomials:
  \[
  f'(x) = \alpha_0 \cdot 1 + \alpha_1 \cdot x + \alpha_2 \cdot x^2
  \]

  Print \texttt{coeffs[-1]} (the last entry of it). What should it be?

```python
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as pt

fp_x =
coeffs =

# plotting code below, no need to modify
fp_interpolant = 0
for i, coeff in enumerate(coeffs):
    fp_interpolant += coeff*xp**i

ft.plot(x, f_x, "o", label="f")
ft.plot(x, fp_x, "o", label="f'")
ft.plot(xp, fp_interpolant)
ft.legend(loc="best")
```