1 | Taylor Series

Preliminaries

For this lab, the main mathematic formula to know is the Taylor polynomial.

Taylor Polynomial

The Taylor polynomial $P_n(x)$ of degree n of f at the point a is:

$$P_n(x) = \sum_{k=0}^{n} \frac{(x-a)^k}{k!} f^{(k)}(a)$$

Taylor bound

Let $P_n(x)$ be the Taylor polyomial of degree n of f at the point a, then the error is bounded by:

$$|f(x) - P_n(x)| \leq \left| f^{(n+1)}(c) \frac{(x-a)^{n+1}}{(n+1!} \right|$$

For some c betwen x and a.

Regarding the programming part we will use:

- **Sympy** to define and differentiate a function.
- **Numpy** to evalaute a function.
- **Matplotlib** to plot a function.
- **Pycodestyle** to make sure your python script looks good.

Before you start, you need to remember the following

- How to differentiate a function Sympy
- How to evaluate a function with Sumpy

```
>>> import sympy as sp
>>> x = sp.symbols('x')
>>> f = x**2
>>> f(2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'Pow' object is not callable
# You need to cast the function as a 'numpy' function
>>> f_eval = sp.lambdify(x, f, "numpy")
>>> f_eval(2)
4
```

- Plotting a function with Numpy and Matplotlib. Dont forget to label the axis with plt.xlabel, plt.ylabel; and to set plt.xlim and plt.ylim.

Instead of **numpy.arange** you may use **numpy.linspace** which takes in argument the number of points as opposed to the step size.

Exercises

Remark

Write all your answers in a PDF file. Your PDF file should be named “LastName_FirstName_Lab1.pdf”.

To make sure your code looks good, do not forget to run

```
$ pycodestyle pythron_script.py
```

You also need to comment your code as much as you can. If you have more that one python script, please write a quick README file to tell me which one to run.

Exercise 1: *Taylor approximations you have to know.*

1. Compute the Taylor approximation of e^x of degree n about $a = 0$.
2. Compute the Taylor approximation of $\sin(x)$ of degree $2n - 1$ about $a = 0$.
3. Compute the Taylor approximation of $\cos(x)$ of degree $2n$ about $a = 0$.
4. Compute the Taylor approximation of $\log(1 + x)$ of degree n about $a = 0$.
5. Compute the Taylor approximation of $\frac{1}{1-x}$ of degree n about $a = 0$.

Exercise 2: Compute the taylor polynomial of degree 2 center at $x = 1$ or $f(x) = \frac{1}{1+x}$.

Exercise 3: *Problem 4 from section 1.1.* Does $f(x) = \sqrt[3]{x}$ have a Taylor polynomial approximation of degree 1 based on expanding about $x = 0$? $x = 1$? Explain and justify your answers.

Exercise 4: *Problem 12 from section 1.1.* The quotient

$$g(x) = \frac{\log(1 + x)}{x}$$

is undefined for $x = 0$. Approximate $\log(1 + x)$ using Taylor polynomials of degrees 1, 2, and 3, in turn, top determine a natural definition of $g(0)$.

Exercise 5: *Problem 9 from section 1.2.* Use the Taylor Polynomials with remainder term to evaluate the following limits:

1. $$\lim_{x\to 0} \frac{1 - \cos(x)}{x^2}$$
2. $$\lim_{x\to 0} \frac{\log(1 + x^2)}{2x}$$
3. $$\lim_{x\to 0} \frac{\log(1 - x) + xe^{x/2}}{x^3}$$

Exercise 6: *Taylor Polynomials.*

1. Write a program that plots and writes in the console the Taylor polynomials of degrees $n_1, n_2, \ldots$ of a function f (defined at the beginning of the script using sympy) at the point a.

2. Plot on the $x-$interval $[0, 2\pi]$: $\sin(x)$, $P_1(x)$, $P_3(x)$, $P_5(x)$, where the Taylor polynomials are centered at π.
3. Plot on the $x-$interval $[-1, 5]$ and $y-$ interval $[-0.5, 1]$: $e^{-x}\sin(x)$, $P_8(x)$, $P_9(x)$, $P_{10}(x)$, $P_{11}(x)$, where the Taylor polynomials are centered at $x = 0$.

Exercise 7: Produce/Plot the linear and quadratic Taylor polynomials of the following cases. Graph the function and the two polynomials for each case, also write the two polynomials. (Use Sympy)

1. $f(x) = \sqrt{(}x)$, $a = 1$
2. $f(x) = \sin(x)$, $a = \frac{\sqcap}{4}$
3. $f(x) = e^{\cos(x)}$, $a = 0$
4. $f(x) = \log(1 + e^x)$, $a = 0$

Exercise 8: *Inverse of the exponential function*.

1. Produce and plot on the interval $(-10, 1)$ the Taylor polynomials of degrees 1, 2, 3, 4 for $f(x) = e^x$, with $a = 0$ (we will denote them P_1, P_2, P_3, and P_4. What happens when x is negative?
2. Now, using the fact that $e^x = \frac{1}{e^{-x}}$, compute the Taylor polynomial of degree 4 of e^{-x}, that we denote $\tilde{P}_4$, then plot $1/\tilde{P}_4(x)$, e^x, $P_4(x)$. What can you conclude?
3. Then, if you where to approximate e^{-10} with a polynomial of degree 4 how would you do it?
4. We can do even better! We want a better approximation of e^{-10} but we do not want to use a polynomial of higher degree than 4. How could you improve the previous approximation? (Hint: $e^{2x} = (e^x)^2$)

Exercise 9: *Problem 2 from section 1.2*. Find the Taylor of polynomial of degree 2 for $e^x \sin(x)$, about the point 0. Bound the error in this approximation when $-\pi/4 \leq x \leq \pi/4$.

Exercise 10: Bound the error $\sin(x) \simeq x$ for $-\pi/4 \leq x \leq \pi/4$.

Exercise 11: Let $P_n(x)$ be the Taylor polynomial of degree n of the function $f(x) = \log(1 - x)$ about $a = 0$. How large should n be chosen to have $|f(x) - P_n(x)| \leq 10^{-4}$ for $-1/2 \leq x \leq 1/2$?

Exercise 12: ***BONUS**: Evaluating polyomials*. Let us evaluate a polynomial P.

Algorithm 1: “Classic”

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Algorithm 2: “Nested multiplication”

$$P(x) = a_0 + x\left(a_1 + x\left(a_2 + \cdots + x\left(a_{n-1} + x a_n\right)\right)\right)$$

1. How many multiplications and additions are done when using each algorithm. Which one is better?
2. Modify your script that compute the Taylor expansion of functions by creating a new Python class that use the best algorithm to evaluate the Taylor polynomial.