MAS212 Assignment #1 (2017): Asymptotic series

Dr. Sam Dolan (s.dolan@sheffield.ac.uk)

In this assignment you will use Python to evaluate a Taylor series and an asymptotic series. Your completed assignment will consist of **one** `.py` script file. Your files should be submitted at `http://somas-uploads.shef.ac.uk/mas212`. The submission deadline is found on the course website. Please refer to Appendix A for the preferred format for the file, and to Appendix B for the sample output of your script. See also the 'warm-up exercise' document.

Part 0: Introduction

A function $f(x)$ that is continuous and differentiable at $x = x_0$ has a **Taylor series** expansion

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) + \ldots + \frac{1}{n!}(x - x_0)^n f^{(n)}(x_0) + \ldots \quad (1)$$

where $f'(x_0)$, $f''(x_0)$ and $f^{(n)}(x_0)$ denote the 1st, 2nd and nth derivatives evaluated at $x = x_0$. Typically, a Taylor series comes with a certain radius of convergence.

The Taylor series for $\arctan(x)$ at $x = 0$ is

$$\arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + \ldots \quad (2)$$

One way to establish this is by starting with binomial series for the derivative,

$$\frac{d}{dx}\arctan(x) = \frac{1}{1 + x^2} = 1 - x^2 + x^4 - x^6 + \ldots \quad (3)$$

and integrating. The radius of convergence for these series is $|x| < 1$.

Now consider the function $f(x)$ defined by

$$f(x) = \int_0^\infty \frac{e^{-xt}}{1 + t} dt. \quad (4)$$

If one naively applies the binomial expansion to $(1 + t)^{-1}$ in the integrand, and integrates each term by parts, one arrives at an **asymptotic series** given by

$$\mathfrak{f}(x) = \frac{1}{x} - \frac{1!}{x^2} + \frac{2!}{x^3} - \frac{3!}{x^4} + \ldots \quad (5)$$

But this series **does not converge for any value of** x! It is a divergent series. Nevertheless, the partial sums of $\mathfrak{f}(x)$ can be used in practice for approximating the function $f(x)$, as we shall see.

Challenge: Prove that the partial sums of this asymptotic series (5) approximate the function $f(x)$ with an error that is numerically smaller than the first neglected term of the series.

Part 1: Evaluating a Taylor series

(a) Enter the function below in your `.py` script file. Add comments to the function (using `#`) to explain how it works. Add a docstring to the beginning of the function.

```
import numpy as np
import matplotlib.pyplot as plt

def arctan1(x, nmax=10):
    terms = np.zeros(nmax)
    for k in range(nmax):
        terms[k] = (-1)**k * x**(2*k+1) / (2*k+1)
    return np.sum(terms)
```

(b) Write code to print to screen a table that looks similar to the following:

```
x       arctan(x)       difference
---------------------------------
0.0     0.0000000000
        0.0000000000    0.000e+00
0.2     0.1973955598
        0.1973955598    1.110e-16
0.4     0.3805063769
        0.3805063771    1.828e-10
0.6     0.5404187137
        0.5404195003    7.866e-07
0.8     0.6744634217
        0.6747409422    2.775e-04
1.0     0.7604599047
        0.7853981634    2.494e-02
```

Here the first line shows the partial sum of the first 10 terms of the Taylor series (2), and the second line shows the precise value calculated using the library function `np.arctan()`. To make this table, I called the functions `arctan1()` and `np.arctan()` from within a `for` loop, and I used a print statement with formatting. **If you are stuck on this step, please ask for help**.

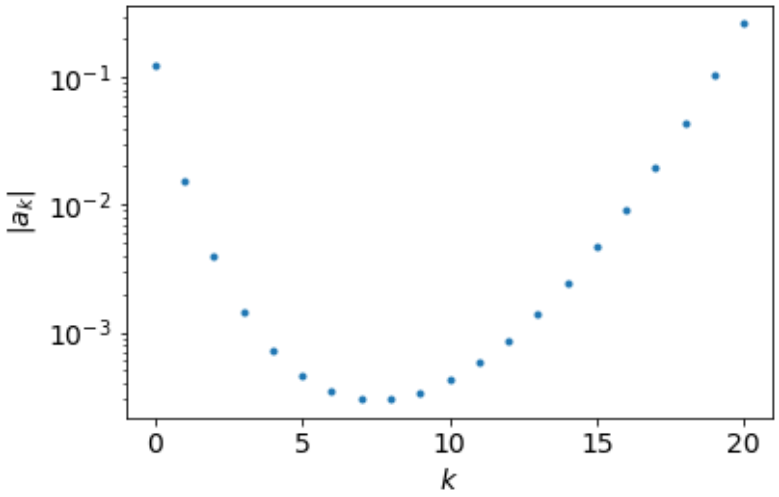
(c) Write a new function `arctan2(x, nmax=100, err=1e-6)` with the following docstring:

```
def arctan2(x, nmax=100, err=1e-6):
    """This function estimates arctan(x) using a partial sum of its Taylor series at x=0,
    up to the nth term (as in arctan1) or up to the first term whose absolute
    value is less than 'err', whichever occurs first.
    The function returns the estimate and the number of terms used, in a tuple."""
```

(d) Use the identity $\arctan(1/\sqrt{3}) = \pi/6$ and your function `arctan2()` to obtain an approximation to π that is accurate to 10 decimal places. Print your result to 12 decimal places. How many terms in the series are needed?

Part 2: Evaluating an asymptotic series

(a) The kth term in the asymptotic series (5) is $a_k = -k!(-x)^{-(k+1)}$. For the case $x = 8$, use `plt.semilogy()` to create a labelled plot showing $|a_k|$ for $k = 0, 1, 2, \ldots 20$, on a logarithmic scale. Your plot may look similar (but not identical) to the one below:



(b) The plot shows that the terms in the series (5) decrease at first, like a convergent series, but eventually begin to diverge. This is the typical behaviour of an asymptotic series. To get the best approximation from a partial sum of the asymptotic series, we should truncate the series just after the smallest term. In other words, find

$$S = \sum_{k=0}^{k=n} a_k, \quad \text{where n is the smallest positive integer s.t.} \quad |a_n| \le |a_{n-1}|, \quad |a_n| \le |a_{n+1}|.$$

Write a function `asymp(x, nmax=30)` to evaluate S, the best estimate. The function should return a tuple (S, a_{n+1}), where a_{n+1} is the next term in the series.

(c) Write a new function `f(x)` to evaluate $f(x)$ defined in Eq. (4) using numerical integration. You may wish to use functions available in the `scipy` package.

(d) Print a table, similar in format to that in part 1(b), showing $\mathfrak{f}$ (the partial series in part 2b), $f(x)$ (the numerically-evaluated function of part 2c), as well as the difference $f(x) - \mathfrak{f}(x)$ and the first neglected term a_{n+1}. Show data for $x = 5, 6, 7, 9, 10$. Check that the absolute value of the difference is indeed smaller than $|a_{n+1}|$ in each case.

Guidance:

This assignment will count for 25%–30% of your module mark. This assignment is intended to help you learn as well as to test your skills. If stuck, please don't struggle in silence: talk to your classmates and tutors.

For this assignment, **fair means** include: discussing the assignment in general terms with classmates; making use of Appendix B to check the output of your code; using a code example from this brief; etc. **Unfair means** include (but are not limited to): sharing or distributing files; copying-and-pasting from work that is not your own; passing off other's work as your own; posting your work online, etc. Note that you **may** discuss the assignment in general terms on the online discussion board. However, you **may not** post any code there. If in doubt, please contact the course tutor.

Note that all submissions will be checked for excessive similarities. Where there is circumstantial evidence of unfair means, I reserve the right to award zero marks for this assignment.

Appendix A: File format. Include a docstring at the top of the script file with your name and registration number. Next, import any modules you will use. Next, define all the functions used. Finally, call the functions to produce human-readable output for each of the 'tasks'. Please make the output easy to read for the marker running the script (for example, indicate which task is being completed). See the example below.

```
"""
Sam Dolan. Registration number: xxxxxxxxx
"""
#####
# (1) Import modules
import numpy as np
import matplotlib.pyplot as plt

#####
# (2) Define all the functions we will use here

def arctan1(x, nmax=10):
    terms = np.zeros(nmax)
    for k in range(nmax):
        terms[k] = (-1)**k * x**(2*k+1) / (2*k+1)
    return np.sum(terms)

def arctan2(x, nmax=100, err=1e-10):
    """This function estimates arctan(x) using a partial sum of its Taylor series around x=0,
    up to the nth term (as in arctan1) or up to the first term whose absolute
    value is less than 'err', whichever occurs first.
    The function returns the estimate and the number of terms used, in a tuple."""
    ...

#####
# (3) Produce nicely-formatted output for each task on the sheet.

print("Part 1(a)")
print("See code: I have added comments and a docstring to the function.")
print("-----")

print("Part 1(b)")
...
```

Appendix B: Example output of your script

```
Part 1(a)
See code: I have added comments and a docstring to the function.
-----
Part 1(b)
...
```