MATH 210 Assignment 4

NumPy and Matplotlib

INSTRUCTIONS

- Create a new Python 3 Jupyter notebook
- Answer each question in the Jupyter notebook and clearly label the solutions with headings
- Functions should include documentation strings and comments
- There are 24 total points and each question is worth 4 points
- Submit the `.ipynb` file to Connect by **6pm Tuesday, February 14, 2017**
- You may work on these problems with others but you must write your solutions on your own

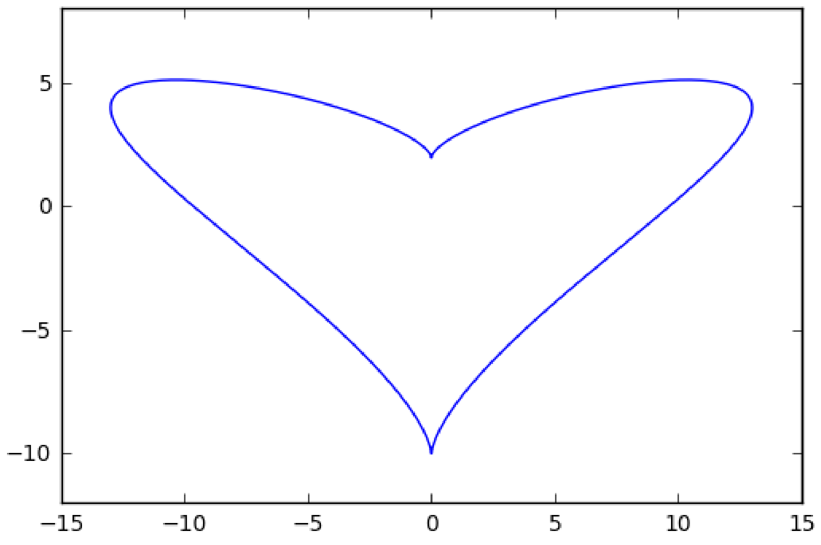
QUESTIONS

1. Define a function called `curve` which takes inputs A, B and C and plots the parametric curve:

$$x = A\sin^3(t) \ , \quad y = B\cos(t) - C\cos(2t) \ , \ t \in [0, 2\pi] \ ,$$

Use the `plt.axis('equal')` command to display the figure with equal units on both axes. For example:

```
curve(13,6,4)
```



2. Write a function called `power_series` which takes 2 input parameters `a` and `x` where `a` is a 1-dimensional NumPy array representing a sequence $a_0, a_1, \dots, a_N$ and `x` is a number, and the function returns the (partial) power series sum

$$\sum_{k=0}^{N} a_0 x^k$$

For example:

```
power_series(np.ones(100),0.5)
```

```
2.0
```

```
from scipy.special import factorial
```

```
power_series(1 / factorial(np.arange(0,100)),1)
```

```
2.718281828459045l
```

3. (a) Write LaTeX code to display the Fourier series of the triangle wave:

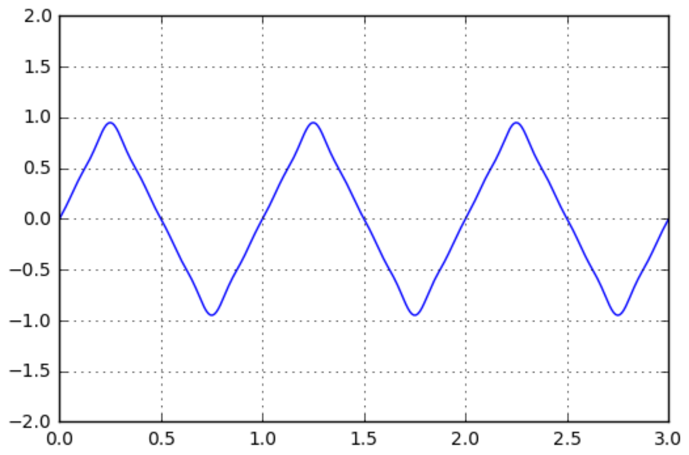
$$f_{\text{triangle}}(t) = \frac{8}{\pi^2} \sum_{k=0}^{\infty} (-1)^k \frac{\sin(2\pi(2k+1)t)}{(2k+1)^2}$$

(b) Write a function called `triangle_wave` which takes a positive integer `N` and a Python list `interval` of length 2 and plots the Nth partial sum of the Fourier series:

$$f_{\text{triangle},N}(t) = \frac{8}{\pi^2} \sum_{k=0}^{N} (-1)^k \frac{\sin(2\pi(2k+1)t)}{(2k+1)^2}$$

over the interval given by the list `interval`. For example:

```
triangle_wave(3,[0,3])
```



4. (a) Write LaTeX code to display the Fourier series of the sawtooth wave:

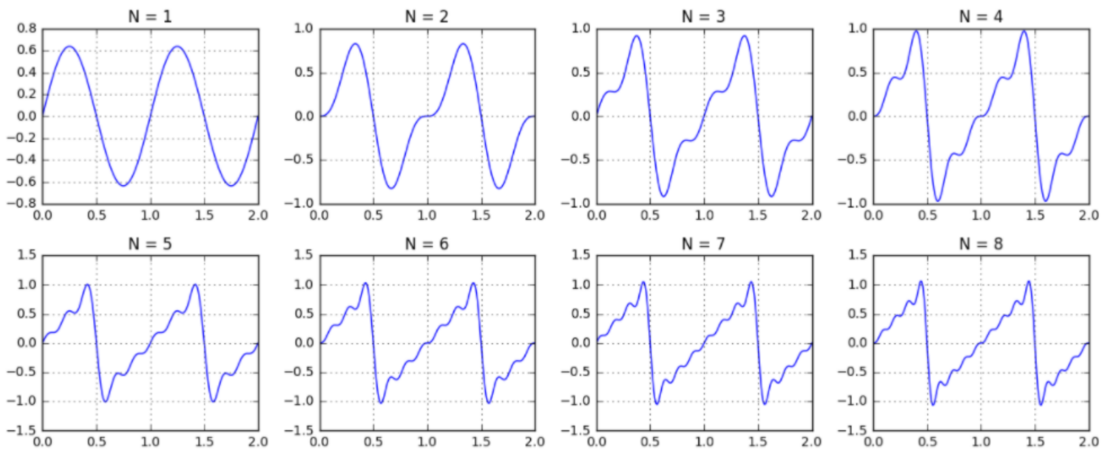
$$f_{\text{sawtooth}}(t) = \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{(-1)^{k+1} \sin(2\pi kt)}{k}$$

(b) Write a function called **`sawtooth_waves`** which takes 3 parameters n, m and T and creates a n by m grid of subplots (with nm total plots) where the Nth partial sum of the Fourier series

$$f_{\text{sawtooth},N}(t) = \frac{2}{\pi}\sum_{k=1}^{N} \frac{(-1)^{k+1}\sin(2\pi k t)}{k}$$

is plotted in the Nth subplot position over the interval $[0,T]$. For example:

```
sawtooth_waves(2,4,2)
```



Note: the command `plt.tight_layout()` will provide spacing between subplots to display the figure properly.

5. (a) Write LaTeX code to display the Euler product formula for the Riemann zeta function:

$$\zeta(s) = \sum_{n=1}^{\infty}\frac{1}{n^s} = \prod_{p\ \text{prime}}\frac{1}{1-p^{-s}} = \frac{1}{1-2^{-s}}\cdot\frac{1}{1-3^{-s}}\cdot\frac{1}{1-5^{-s}}\cdot\frac{1}{1-7^{-s}}\cdots\frac{1}{1-p^{-s}}\cdots$$

(b) Write a function called **`euler_product`** which take 2 input parameters `s` and `N` and computes the partial Euler product

$$\prod_{p\le N}\frac{1}{1-p^{-s}} = \frac{1}{1-2^{-s}}\cdot\frac{1}{1-3^{-s}}\cdot\frac{1}{1-5^{-s}}\cdot\frac{1}{1-7^{-s}}\cdots\frac{1}{1-p_N^{-s}}$$

where p_N denotes the largest prime less than or equal to N. For example:

```
euler_product(4,10000)
```

```
1.0823232337111559
```

```
np.pi**4/90
```

```
1.082323233711138
```

The example above shows an approximation for the special value formula:

$$\zeta(4) = \frac{\pi^4}{90}$$

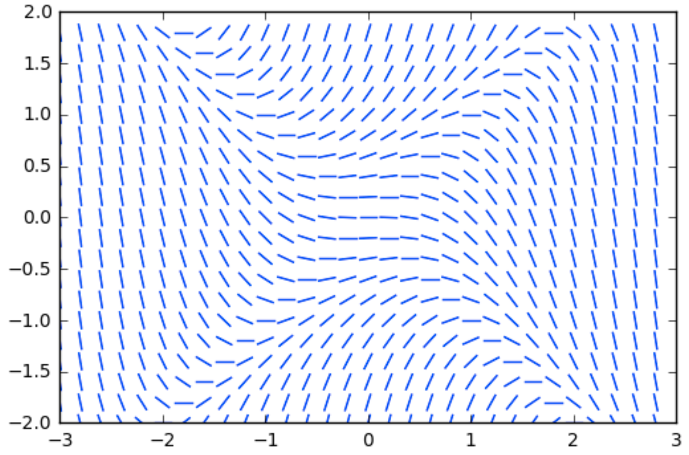
6. Write a function called `slope_field` which takes 4 input parameters `f`, `tlims`, `ylims` and `grid_step` where

   - `f` is a function of 2 variables $f(t, y)$ representing the right side of a first order differential equation $y' = f(t, y)$
   - `tlims` and `ylims` are Python lists of length 2 which set the display limits of the figure
   - `grid_step` is a number which sets the distance between grid points in the plot

   The function should plot a small line (ie. length smaller than `grid_step`) of slope $f(t_i, y_j)$ centred at (t_i, y_j) for each point (t_i, y_j) in the grid of points defined by the t and y limits and the grid step. The result is the slope field for the equation $y' = f(t, y)$. For example:

```
def f(t,y):
    return y**2 - t**2
```

```
slope_field(f,[-3,3],[-2,2],0.2)
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−3
−2
−1
0
1
2
3