MATH 210 Exam 2

March 23, 2016

INSTRUCTIONS

- Create a new Jupyter notebook, set the kernel to Python 3, present your solutions in the notebook and clearly label the solutions
- This is **an open book exam** and you may consult any online resources (such as `python.org`), notes from class and past assignments, but **the only rule is that you may NOT communicate with others in the class** (via email, text, Snapchat, Slack, Facebook, etc.)
- There are 6 questions and 25 total points: each question is worth 4 points and 1 point will be awarded for the overall presentation of your notebook
- Submit the completed `.ipynb` file to Connect **by 7:30pm**

QUESTIONS

1. Define a function called `exp_integral` with parameters `a`, `b`, `c` and `n` (in that exact order `exp_integral(a,b,c,n)`) where `a`, `b` and `c` are all positive numbers and `n` is a positive integer. The function `exp_integral` should use the function `scipy.integrate.trapz` to compute and return an approximation of the integral

   $$\int_0^c x^a e^{-bx} \, dx$$

   The integer `n` is the number of evenly spaced points in the NumPy array of x values (over the interval $[0, c]$) used in the function `scipy.integrate.trapz`. If *any* of the input arguments are less than 0, the function should display the message "Error: Negative parameters" and return `None`.

2. Define a function called `pq_integral` with parameters `p`, `q` and `b` (in that exact order `pq_integral(p,q,b)`) where `p`, `q` and `b` are positive numbers. The function `pq_integral` should use the function `scipy.integrate.quad` to compute and return an approximation of the definite integral

   $$\int_0^b \frac{x^p}{\sqrt{1+x^q}} dx$$

   All three parameters `p`, `q` and `b` should be positive numbers therefore, if any of the input arguments are less than 0, the function `pq_integral` should display the error message "Error: Negative parameters" and return `None`.

3. Define a function called `sin_cos_fun` with parameters `a`, `b` and `t` (in that exact order `sin_cos_fun(a,b,t)`) where `a` and `b` are numbers and `t` is a NumPy array. The function should plot the function

   $$F_{a,b}(t) = \int_0^t \left(\sin(ax) + \cos(bx^2)\right) dx$$

over the interval defined by the array `t`. (Note: the function $F_{a,b}(t)$ is the solution of the first order differential equation $y' = \sin(at) + \cos(bt^2)$ with $y(0) = 0$.)

4. Define a function called `mathieu` with parameters `a`, `q`, `y0` and `t` (in that exact order `mathieu(a,q,y0,t)`) where

   `a` and `q` are numbers

   `y0` is a Python list of length 2, the initial conditions $[y(t_0), y'(t_0)]$

   `t` is a NumPy array of t values where the first entry is t_0 as in the initial conditions

   The function `mathieu` should use the function `scipy.integrate.odeint` to solve and plot the solution $y(t)$ of the differential equation

   $$\frac{d^2y}{dt^2} + (a - 2q\cos(2t))y = 0$$

   over the interval defined by the array `t` and given the initial conditions in `y0`. (A solution to this differential equation is called a Mathieu function.)

5. Define a function called `ode_system` with parameters `a`, `b`, `c`, `u0` and `t` (in that exact order `ode_system(a,b,c,u0,t)`) where

   `a`, `b` and `c` are numbers

   `u0` is a Python list of length 3, the initial conditions $[x(t_0), y(t_0), z(t_0)]$

   `t` is a NumPy array of t values where the first entry is t_0 as in the initial conditions

   The function `ode_system` should use the function `scipy.integrate.odeint` to solve the first order system of differential equations

   $$\begin{aligned} x' &= a(y - x) \\ y' &= x(b - z) \\ z' &= xy - cz \end{aligned}$$

   over the interval defined by the array `t` and given the initial conditions in `u0`, and then plot the solutions $x(t)$ **versus** $y(t)$. (The solution is 3-dimensional $[x(t), y(t), z(t)]$ but you are being asked to plot $x(t)$ versus $y(t)$ in a 2D plot. See `http://www.math.ubc.ca/~pwalls/data/ode_system.png` for sample output.)

6. Recall that if A is any matrix then the product AA^T is a symmetric matrix and it is well-known that the eigenvalues of a symmetric matrix are *real* numbers. Define a function called `eigvals_squared` with parameter `A` which is a NumPy array. The function should return the sum of the squares of the eigenvalues $\lambda_1, \dots, \lambda_n$ of AA^T

   $$\sum_{i=1}^{n} \lambda_i^2$$

   Since all the eigenvalues are real, the output of this function should be a positive real number (ie. a float as opposed to a complex datatype). You may use the array method `.real` to convert a NumPy array of complex numbers into an array of real numbers.