ME314 Homework 0

Submission instructions

Deliverables that should be included with your submission are shown in **bold** at the end of each problem statement and the corresponding supplemental material. Your homework will be graded IFF you submit a **single** PDF and a link to a Google colab file that meet all the requirements outlined below.

- List the names of students you've collaborated with on this homework assignment.
- Include all of your code (and handwritten solutions when applicable) used to complete the problems.
- Highlight your answers (i.e. **bold** and outline the answers) and include simplified code outputs (e.g. .simplify()).
- Enable Google Colab permission for viewing
  - Click Share in the upper right corner
  - Under "Get Link" click "Share with..." or "Change"
  - Then make sure it says "Anyone with Link" and "Editor" under the dropdown menu
- Make sure all cells are run before submitting (i.e. check the permission by running your code in a private mode)
  - Please don't make changes to your file after submitting, so we can grade it!
- Submit a link to your Google Colab file that has been run (before the submission deadline) and don't edit it afterwards!

Problem 1 (20pts)

Given a function $f(x) = \sin(x)$, find the derivative of $f(x)$ and find the directional derivative of $f(x)$ in the direction v. Moreover, compute these derivatives using Pythons's SymPy package.

As an example, let's solve the problem when $f(x) = x^2$ (feel free to take the following code as a start point for your solution):

```
import sympy as sym
from sympy import symbols

####################################
# Part 1: compute derivative of f

# define your symbolic variable here
x = symbols('x')

# define the function f
f = x**2 # if you're using Jupyter-Notebook, try "display(f)"

# compute derivative of f
# (uncomment next line and add your code)
df = f.diff(x)

# output resutls
print("derivative of f: ")
display(df)

####################################
# Part 2: compute directional derivative of f

# define dummy variable epsilon, and the direction v
# note 1: here the character 'r' means raw string
# note 2: here I define the symbol for epsilon with
#         the name "\epsilon", this is for LaTeX printing
#         later. In your case, you can give it any other
#         name you want.
eps, v = symbols(r'\epsilon, v')
```

```
# add eplision into function f
new_f = (x + v*eps)**2

# take derivative of the new function w.r.t. epsilon
df_eps = new_f.diff(eps)

# output this derivative
print("derivative of f wrt eps: ")
display(df_eps)

# now, as you've seen the class, we need evaluate for eps=0 to ...
# ... get the directional derivative. To do this, we need to ...
# ... use SymPy's built-in substitution method "subs()" to ...
# ... replace the epsilon symbol with 0
new_df = df_eps.subs(eps, 0)

# output directional derivative
print("directional derivative of f on v: ")
display(new_df)
```

Turn in: A scanned (or photograph from your phone or webcam) copy of your hand written solution for both derivatives (or you can use LaTeX, instead of hand writing). Also, turn in the code used to compute the symbolic solutions for both derivatives.

Problem 2 (20pts)

Given a function of trajectory:

$$J(x(t)) = \int_0^{\pi/2} \frac{1}{2} x(t)^2 dt$$

Compute the analytical solution when $x = \cos(t)$, verify your answer by numerical integration.

The code for numerical integration is provided below:
(A small challenge with zero credit: if you're interested, implement this function within two lines!)

```
def integrate(func, xspan, step_size):
    '''
    Numerical integration with Euler's method

    Parameters:
    ====================
    func: Python function
        func is the function you want to integrate for
    xspan: list
        xspan is a list of two elements, representing
        the start and end of integration
    step_size:
        a smaller step_size will give a more accurate result

    Returns:
    ====================
    int_val:
        result of the integration
    '''
    import numpy as np
    x = np.arange(xspan[0], xspan[1], step_size)
    int_val = 0
    for xi in x:
        int_val += func(xi) * step_size
    return int_val

# a simple test
def square(x):
    return x**2
```

```
print( integrate(func=square, xspan=[0, 1], step_size=0.01) )
# or you just call the function without indicating parameters
# print( integrate(square, [0, 1], 0.01) )
```

Turn in: A scanned (or photograph from your phone or webcam) copy of your hand written analytical solution (or you can use LaTeX). Also, turn in the code you used to numerically evaluate the result.

Problem 3 (20pts)

For the function $J(x(t))$ in Problem 2, compute and evaluate the analytical solution for the directional derivative of J at $x(t) = \cos(t)$, in the direction $v(t) = \sin(t)$.

Turn in: A scanned (or photograph from your phone or webcam) copy of your hand written analytical solution (or you can use LaTeX), you need to evaluate the integration in this problem.

Problem 4 (20pts)

Verify your answer in Problem 3 symbolically using Python's SymPy package, this means you need to compute the directional derivative and evaluate the integration all symbolically.

Hint 1: Different from computing directional derivative in Problem 1, this time the function includes integration. Thus, instead of defining x as a symbol, you should define x as a function of symbol t. An example of defining function and taking the derivative of the function integration is provided below.

```
import sympy as sym
from sympy import symbols, integrate, Function, pi, cos, sin
from sympy.abc import t

# define function x and y
x = Function('x')(t)
y = Function('y')(t)
# define J(x(t), y(t))
J = integrate(x**2 + x*y, [t, 0, pi])
print('J(x(t), y(t)) = ')
display(J)

# take the time derivative of J(x(t))
dJdx = J.diff(x)
print('derivative of J(x(t), y(t)) wrt x(t): ')
display(dJdx)

# now, we have x(t)=sin(t) and y(t)=cos(t), we substitute them
# in, and evaluate the integration
dJdx_subs = dJdx.subs({x:sin(t), y:cos(t)})
print('derivative of J, after substitution: ')
display(dJdx_subs)
print('evaluation of derivative of J, after substitution: ')
display(sym.N(dJdx_subs))
```

Turn in: A copy of the code you used to numerically and symbolically evaluate the solution.

Problem 5 (20pts)

Given the equation:

$$xy + \sin(x) = x + y$$

Use Python's SymPy package to symbolically solve this equation for y, thus you can write y as a function of x. Transfer your symbolic solution into a numerical function and plot this function for

$x \in [0, \pi]$ with Python's Matplotlib package.

In this problem you will use two methods in SymPy. The first is its symbolic sovler method `solve()`, which takes in an equation or expression (in this it equals 0) and solve it for one or one set of variables. Another method you will use is `lambdify()`, which can transfer a symbolic expression into a numerical function automatically (of course in this problem we can hand code the function, but later in the class we will have super sophisticated expression to evaluate.

Below is an example of using these two methods for an equation $2x^3 \sin(4x) = xy$ (feel free to take this as the start point for your solution):

```
import sympy as sym
import numpy as np
from sympy import sin, cos
from sympy.abc import x, y # it's same as defining x, y using symbols()
import matplotlib.pyplot as plt

# define an equation
eqn = sym.Eq(x**3 * 2*sin(4*x), x*y)
print('original equation')
display(eqn)

# solve this equation for y
y_sol = sym.solve(eqn, y) # this method returns a list,
                          # which may include multiple solutions
print('symbolic solutions: ')
print(y_sol)
y_expr = y_sol[0] # in this case we just have one solution

# lambdify the expression wrt symbol x
func = sym.lambdify(x, y_expr)
print('Test: func(1.0) = ', func(1.0))

###############
# now it's time to plot it from 0 to pi

# generate list of values from 0 to pi
x_list = np.linspace(0, np.pi, 100)

# evaluate function at those values
f_list = func(x_list)

# plot it
plt.plot(x_list, f_list)
plt.show()
```

Turn in: A copy of the code used to solve for symbolic solution and evaluate it as a numerical function. Also, include the plot of the numerical function.