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!

NOTE: This Jupyter Notebook file serves as a template for you to start homework. Make sure you first copy this template to your own Google drive (click "File" -> "Save a copy in Drive"), and then start to edit it.

Sean Morton

Students I worked with on this homework: N/A

In [1]:
```
#IMPORT ALL NECESSARY PACKAGES AT THE TOP OF THE CODE
import sympy as sym
import numpy as np
import matplotlib.pyplot as plt

from IPython.display import Markdown, display

# How to print in bold.You could wrap the "display(Markdown())""
# in a function if you want a more concise alternative.
```

```
# answer = 42

# display(Markdown("**Hello World: {}**".format(answer)))
```

In [2]:
```
# ####################################################################################
# # If you're using Google Colab, uncomment this section by selecting the whole sectio
# # ctrl+'/' (Linux/Windows) or cmd+'/' (MacOS) on your and keyboard. Run it before yo
# # programming, this will enable the nice LaTeX "display()" function for you. If you'
# # the local Jupyter environment, leave it alone
# ####################################################################################


# def custom_latex_printer(exp,**options):
#     from google.colab.output._publish import javascript
#     url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/3.1.1/latest.js?config=TeX
#     javascript(url=url)
#     return sym.printing.latex(exp,**options)
# sym.init_printing(use_latex="mathjax",latex_printer=custom_latex_printer)
```

In [ ]:

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.

Hint 1: feel free to take the starting code as a start point for your solution.

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 and the code output.

In [3]:
```
####################################
# Part 1: compute derivative of f

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

# define the function f
f = sym.sin(x) # 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
```

```python
# define dummy variable epsilon, and the direction v
eps, v = sym.symbols(r'\epsilon, v')

fnew = sym.sin(x + eps*v)

#differentiate w.r.t. the scaling factor
dfde = fnew.diff(eps)
print("derivative of f wrt epsilon:")
display(dfde)

#evaluated at epsilon = 0, this gives the directional deriv in v direction
dfnew = dfde.subs(eps, 0)
display(Markdown("**directional derivative of f in v direction:**"))
display(dfnew)
```

derivative of f:

$\cos(x)$

derivative of f wrt epsilon:

$v\cos(\epsilon v + x)$

directional derivative of f in v direction:

$v\cos(x)$

For Problem 1 handwritten work: see PDF

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:

In [4]:
```python
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:
```

ME314 HW0 Scan

- Sean Morton, spm6133

1. Derivative and directional derivative of $f(x) = \sin(x)$:

a) $f(x) = \sin(x)$

$Df(x) = \frac{d}{dx}(\sin(x))$

$\boxed{Df(x) = \cos(x)}$ $\longrightarrow$ derivative

b) Directional derivative:

$f(x) = \sin(x)$

$f(x + \varepsilon N) = \sin(x + \varepsilon N)$

$\frac{d}{d\varepsilon} f(x + \varepsilon N) = \frac{d}{dx}(\sin(x + \varepsilon N)) \cdot \frac{d}{d\varepsilon}(x + \varepsilon N)$

$= N\cos(x + \varepsilon N)$

$Df(x) \cdot N = \frac{d}{d\varepsilon} f(x + \varepsilon N) \Big|_{\varepsilon = 0}$

$= N\cos(x + \varepsilon N) \Big|_{\varepsilon = 0}$

$\boxed{Df(x) \cdot N = N\cos(x)}$ $\longrightarrow$ directional derivative

2. ~~Function of a trajectory~~: $\cancel{J(x(t)) = \int_0^{\pi/2} \frac{1}{2} x(t)^2 \, dt}$

```
        result of the integration
    ====================
    '''
    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
```

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 and the code output.

In [5]:
```
#my implementation:

#given x(t) = cos(t) and J(x(t)) = int{0}{pi/2} (1/2*x(t)^2) dt:

def integrand0(t):
    return 0.5 * (np.cos(t))**2

tspan = [0, np.pi/2.0]
step = 0.00005
res = integrate(integrand0, tspan, step)

print("Analytical solution for J(x(t)):")
print(f'Result: {res}')
print(f'Result * 8: {res*8}')
print(f'np.pi/8: {np.pi/8.0}')
```

```
Analytical solution for J(x(t)):
Result: 0.3927115816987216
Result * 8: 3.141692653589773
np.pi/8: 0.39269908169872414
```

For problem 2 written work, see PDF

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 $v(t) = \sin(t)$. The directional derivative should be in the form of integration, evaluate the integration analytically, and verify it using numerical integration.

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. Also, include the code used to numerically verify the integration result.

In [6]:
```
#analytical form of integration: int{0}{pi/2}[ cos(t)sin(t)dt  ]

#numerical integration below

def integrand1(t):
    return np.sin(t) * np.cos(t)
```

(ME314 HW0, pg. 2)
-Sean Morton, spm6133

2. Function of a trajectory: $J(x(t)) = \int_0^{\pi/2} \frac{1}{2} x(t)^2 \, dt$

Given $x(t) = \cos(t)$, $J(x(t)) = \int_0^{\pi/2} \frac{1}{2}\cos^2(t)\,dt$;
solve for $J(t)$.

$\cos^2(t)$ is related to $\cos(2t)$ via this double-angle formula:

$$\cos(2t) = \cos^2(t) - \sin^2(t)$$
$$= 2\cos^2(t) - 1$$
$$\cos^2(t) = \frac{\cos(2t)+1}{2}$$

Analytical solution of $J(x(t))$:

$$J(\cos(t)) = \int_0^{\pi/2} \frac{1}{2}\cos^2(t)\,dt$$
$$= \int_0^{\pi/2} \frac{1}{4}(\cos(2t)+1)\,dt$$
$$= \frac{1}{4}\left[\frac{1}{2}\sin(2t) + t\right]\Big|_0^{\pi/2}$$
$$= \frac{1}{4}\left[\cancel{\frac{1}{2}\sin(\pi)}^{0} + \frac{\pi}{2} - \cancel{\frac{1}{2}\sin(0)}^{0} - 0\right]$$

$$\boxed{J(\cos(t)) = \frac{\pi}{8}} \approx 0.392699$$

```
bounds = [0, np.pi/2.0]
step = 0.001

intval = integrate(integrand1, bounds, step)
print(f"Expected answer: {0.5}")
display(Markdown(f"**Analytical integration: {intval}**"))
```

```
Expected answer: 0.5
```

Analytical integration: 0.4999999144285416

For Problem 2 written work, see PDF

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.

In [7]:
```
#example code:

t = sym.symbols('t')
# define function x and y
x = sym.Function('x')(t)
y = sym.Function('y')(t)
# define J(x(t), y(t))
J = sym.integrate(x**2 + x*y, [t, 0, sym.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:sym.sin(t), y:sym.cos(t)})
print('derivative of J, after substitution: ')
display(dJdx_subs)
display(Markdown("**evaluation of derivative of J, after substitution:**"))
display(sym.N(dJdx_subs))
```

```
J(x(t), y(t)) =
```

$$\int_0^{\pi} (x(t) + y(t))\, x(t)\, dt$$

```
derivative of J(x(t), y(t)) wrt x(t):
```

(ME314 HW0, pg. 3)
- Sean Morton, spm6133

3. Directional derivative of $J(x(t))$ at $x(t) = \cos(t)$ in the direction $\eta(t) = \sin(t)$:

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

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

For $J(x + \varepsilon\eta)$, modify $x(t) = \cos(t)$ to give $x(t) + \varepsilon\eta(t) = \cos(t) + \varepsilon\sin(t)$

$$J(x(t) + \varepsilon\eta(t)) = \int_0^{\pi/2} \frac{1}{2}\left(\cos(t) + \varepsilon\sin(t)\right)^2 dt$$

$$\frac{d}{d\varepsilon} J(x(t) + \varepsilon\eta(t))\Big|_{\varepsilon=0} = \frac{d}{d\varepsilon} \int_0^{\pi/2} \frac{1}{2}\left(\cos(t) + \varepsilon\sin(t)\right)^2 dt\Big|_{\varepsilon=0}$$

differentiation inside the integral

$$= \int_0^{\pi/2} \frac{d}{d\varepsilon} \frac{1}{2}\left(\cos(t) + \varepsilon\sin(t)\right)^2\Big|_{\varepsilon=0} dt$$

$$= \int_0^{\pi/2} \frac{d}{dt}\left(\frac{1}{2}\left(\cos(t) + \varepsilon\sin(t)\right)^2\right) \cdot \frac{d}{d\varepsilon}\left(\cos(t) + \varepsilon\sin(t)\right)\Big|_{\varepsilon=0} dt$$

$$= \int_0^{\pi/2} \left[\left(\cos(t) + \varepsilon\sin(t)\right) \cdot \sin(t)\right]_{\varepsilon=0} dt$$

$$= \int_0^{\pi/2} \cos(t)\sin(t) \, dt$$

$$= \frac{1}{2}\sin^2(t)\Big|_0^{\pi/2}$$

$$= \frac{1}{2}\left[\sin^2(\pi/2) - \overset{0}{\cancel{\sin^2(0)}}\right]$$

$$\boxed{= \frac{1}{2}}$$

$$\int_{0}^{\pi} (2x(t) + y(t))\, dt$$

derivative of J, after substitution:

$$\int_{0}^{\pi} (2\sin(t) + \cos(t))\, dt$$

evaluation of derivative of J, after substitution:

4.0

Turn in: A copy of the code you used to symbolically evaluate the solution as well as the corresponding code output for both.

In [8]:

```
# You can start your implementation here :)

eps, t = sym.symbols(r'\epsilon, t')

#function representations
x = sym.Function('x')(t)
v = sym.Function('v')(t)
f_x = 0.5 * x**2

#form of J(x(t))
J = sym.integrate(f_x, [t, 0, sym.pi/2])
print('J(x(t)): ')
display(J)

#substitute in x + eps*v
J_dir = J.subs(x, (x + eps * v))
print('J(x(t) + eps*v(t)): ')
display(J_dir)

#differentiate wrt epsilon
dJde = J_dir.diff(eps)
print('d/dEps (J(x+eps*v)): ')
display(dJde)

#evaluate at epsilon = 0
dJ_new = dJde.subs(eps, 0)
print('Directional derivative of J * v: ')
display(dJ_new)

#substitute in x(t) = cos(t), v = sin(t)
dJ_subbed = dJ_new.subs({x: sym.cos(t), v: sym.sin(t)})
print('Derivative of J in v direction after substitution: ')
display(dJ_subbed)

#solve.
display(Markdown(" **Evaluation of DJ(x(t))*v after substitution:** "))
display(dJ_subbed.evalf())
```

J(x(t)):

$$0.5 \int\limits_{0}^{\frac{\pi}{2}} x^{2}(t)\, dt$$

```
J(x(t) + eps*v(t)):
```

$$0.5 \int\limits_{0}^{\frac{\pi}{2}} \left(\epsilon v(t) + x(t)\right)^{2}\, dt$$

```
d/dEps (J(x+eps*v)):
```

$$0.5 \int\limits_{0}^{\frac{\pi}{2}} 2 \left(\epsilon v(t) + x(t)\right) v(t)\, dt$$

```
Directional derivative of J * v:
```

$$0.5 \int\limits_{0}^{\frac{\pi}{2}} 2 v(t) x(t)\, dt$$

```
Derivative of J in v direction after substitution:
```

$$0.5 \int\limits_{0}^{\frac{\pi}{2}} 2 \sin{(t)} \cos{(t)}\, dt$$

Evaluation of DJ(x(t))*v after substitution:

$$0.5$$

~~For problem 4 written work, see PDF~~

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 expressions to evaluate.

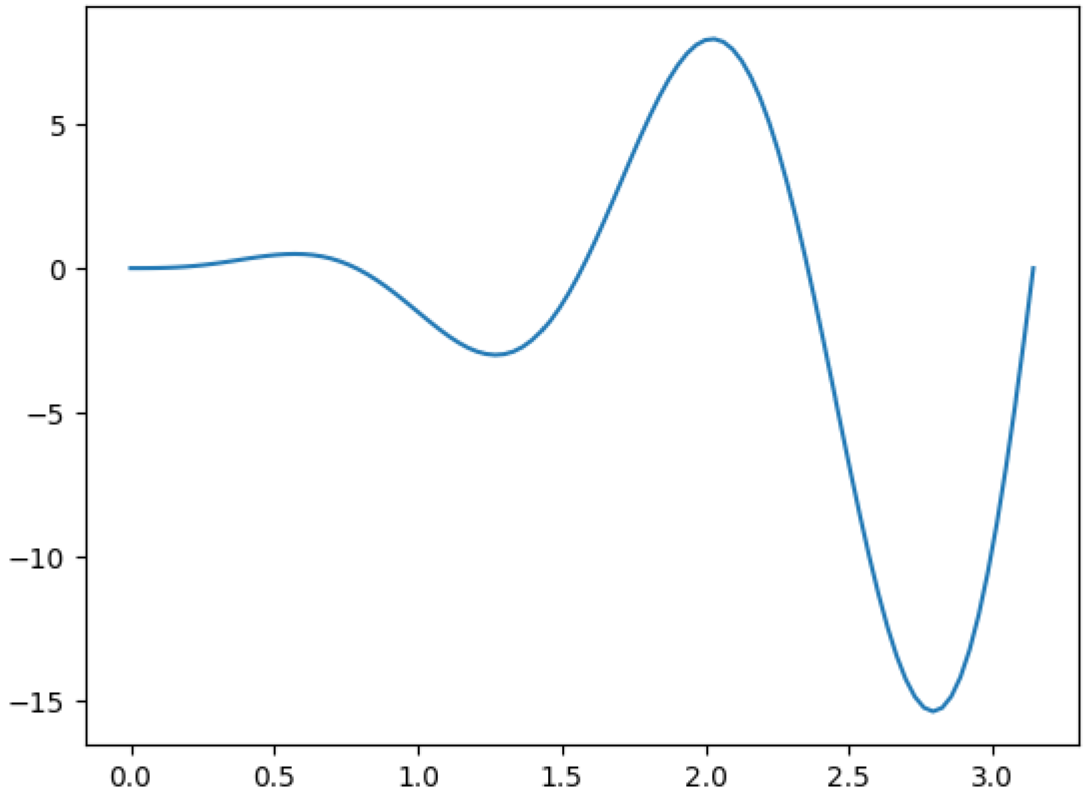
Below is an example of using these two methods for an equation $2x^3 \sin(4x) = xy$ (feel free to take this as the start point for your solution):

In [9]:
```python
#example code

# from sympy.abc import x, y # it's same as defining x, y using symbols() OR
x, y = sym.symbols(r'x,y')

# define an equation
eqn = sym.Eq(x**3 * 2*sym.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()
```

```
Original equation
```

$$2x^3 \sin(4x) = xy$$

```
Symbolic solutions
[2*x**2*sin(4*x)]
Test: func(1.0) =  -1.5136049906158564
```



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.

In [10]:
```
# You can start your implementation here :)

#setup expression, symbols
x, y = sym.symbols('x, y')
eqn = sym.Eq(x*y + sym.sin(x), x + y)
print('Original equation: ')
display(eqn)

#solve()
y_sol = sym.solve(eqn, y)
display(Markdown(f" **Symbolic solutions: <br> {y_sol}** "))

#lambdify()
new_func = sym.lambdify(x, y_sol[0])
print(f'\nTest of lambdified function: {new_func(np.pi/2)}')
print(f'Expected value: 1.0')

#np arrays, then plot
x_list = np.arange(0, np.pi, 0.01)
y_list = new_func(x_list)


fig1 = plt.figure()
plt.plot(x_list, y_list)
plt.show()
```

```
Original equation:
```

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

Symbolic solutions:

[(x - sin(x))/(x - 1)]

```
Test of lambdified function: 1.0
Expected value: 1.0
```

```
<lambdifygenerated-2>:2: RuntimeWarning: divide by zero encountered in divide
  return (x - sin(x))/(x - 1)
```