ME370 Postlab 11

Shihong Yuan

Lab TA: Tim Han
Lab Session: Friday 1:30am-2:50am
November 19, 2025

Inlab Activities

Question 1

```
% --- Section 1 ---

# Comments:
# Because all the link has there center in the middle of the object, so we use 0.5
cmlink = [0.5, 0.5, 0.5, 0.5] #Fractional link distance from 1st node where the
    center of mass lies
# Because all are only one rigid body
cmatbody = [[0], [1], [2], [3]] #Defining rigid bodies as a array
# In the manaual it suggests to use this value
gconst=9.81 #gravitational constant in m/s^2 (positive)
#If neglecting gravity, gconst = 0.
wdth=0.0254 #Width of each link (meters).
tks=0.003175 #thickness of each link (meters).
den=1180 #Density for each link (kg/m^3).
```

Listing 1: Python code for Section 1: Input.

Question 2

```
% --- Section 3 ---


# Comments:
# rcenter=(1-c)*r1 + c*r2
        rcglink[i][k] = r1link[i][k] * (1 - cmlink[i]) + r2link[i][k] * cmlink[i] #
            ################## FILL IN LINK CG CALCULATIONS HERE
```

Listing 2: Python code for Section 3: Moments of Inertia.

```
% --- Section 4 ---
for k in range(tnum-1):
# Comments:
# for derivation we will use the difference divide by 2*dt
# for k=0, we have to use 1*dt
    if k==0:
        omega[:,k] = (theta[:,k+1] - theta[:,k]) / dt
        # FILL IN OMEGA CALCULATION ##################
    else:
        omega[:,k] = (theta[:,k+1] - theta[:,k-1]) / (2*dt)
        # FILL IN OMEGA CALCULATION ##################
#Calculation similar to velocity calculation in PVA
for i in range(len(cmatbody)):
    if cmatbody[i][0] == crlink:
        # from crankvel find omega
        omega[i] = crangvel *np.pi/180 # Fill in omega calculation COMPLETE THIS CODE
            ##########################
alpha=np.zeros([len(cmatbody),tnum-2]) #omega for each body in radians/sec
# Comments:
# alpha is very similar to velocity also divide by two
for k in range(tnum-2):
    if k==0:
        alpha[:,k]=(omega[:,k+1]-omega[:,k])/dt  # alpha[:,k]= COMPLETE THIS CODE
            ##########################
    else:
        alpha[:,k] = (omega[:,k+1] - omega[:,k-1]) / (2*dt) # alpha[:,k]= COMPLETE
            THIS CODE ##########################
```

Listing 3: Python code for Section 4: Angular Kinematics.

Question 3

```
% --- Section 7  ---
    # Comments:
    # Rij is the distance from mass center to the force in vector form, then we
      divide it into x and y axis.

    R12 = node_positions[0,k] - rcg[1,k]
    R32 = node_positions[1,k] - rcg[1,k]
    R23 = node_positions[1,k] - rcg[2,k]
    R43 = node_positions[2,k] - rcg[2,k]
    R34 = node_positions[2,k] - rcg[3,k]
    R14 = node_positions[3,k] - rcg[3,k]

    R12x = np.real(R12); R12y = np.imag(R12)
    R32x = np.real(R32); R32y = np.imag(R32)
    R23x = np.real(R23); R23y = np.imag(R23)
    R43x = np.real(R43); R43y = np.imag(R43)
    R34x = np.real(R34); R34y = np.imag(R34)
    R14x = np.real(R14); R14y = np.imag(R14)

    # Comments:
    # We follow the matrix from the manual to complete all the A matrix

    A = np.array([
        [1, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 1, 0, 0, 0, 0, 0],
        [-R12y, R12x, -R32y, R32x, 0, 0, 0, 0, 1],
        [0, 0, -1, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, -1, 0, 1, 0, 0, 0],
        [0, 0, R23y, -R23x, -R43y, R43x, 0, 0, 0],
        [0, 0, 0, 0, -1, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, -1, 0, 1, 0],
        [0, 0, 0, 0, R34y, -R34x, -R14y, R14x, 0]
    ], dtype=float)

    for i in range(len(bodynotgnd)):

        # Comments:
        # bodynotgnd is the number of link
        # We are putting the F into C matrix
        C[3*i] = M[bodynotgnd[i]] * np.real(acg[bodynotgnd[i], k])
        C[3*i+1] = M[bodynotgnd[i]] * np.imag(acg[bodynotgnd[i], k]) + M[bodynotgnd[i
            ]] * gconst
        C[3*i+2] = I[bodynotgnd[i]] * alpha[bodynotgnd[i], k]
```

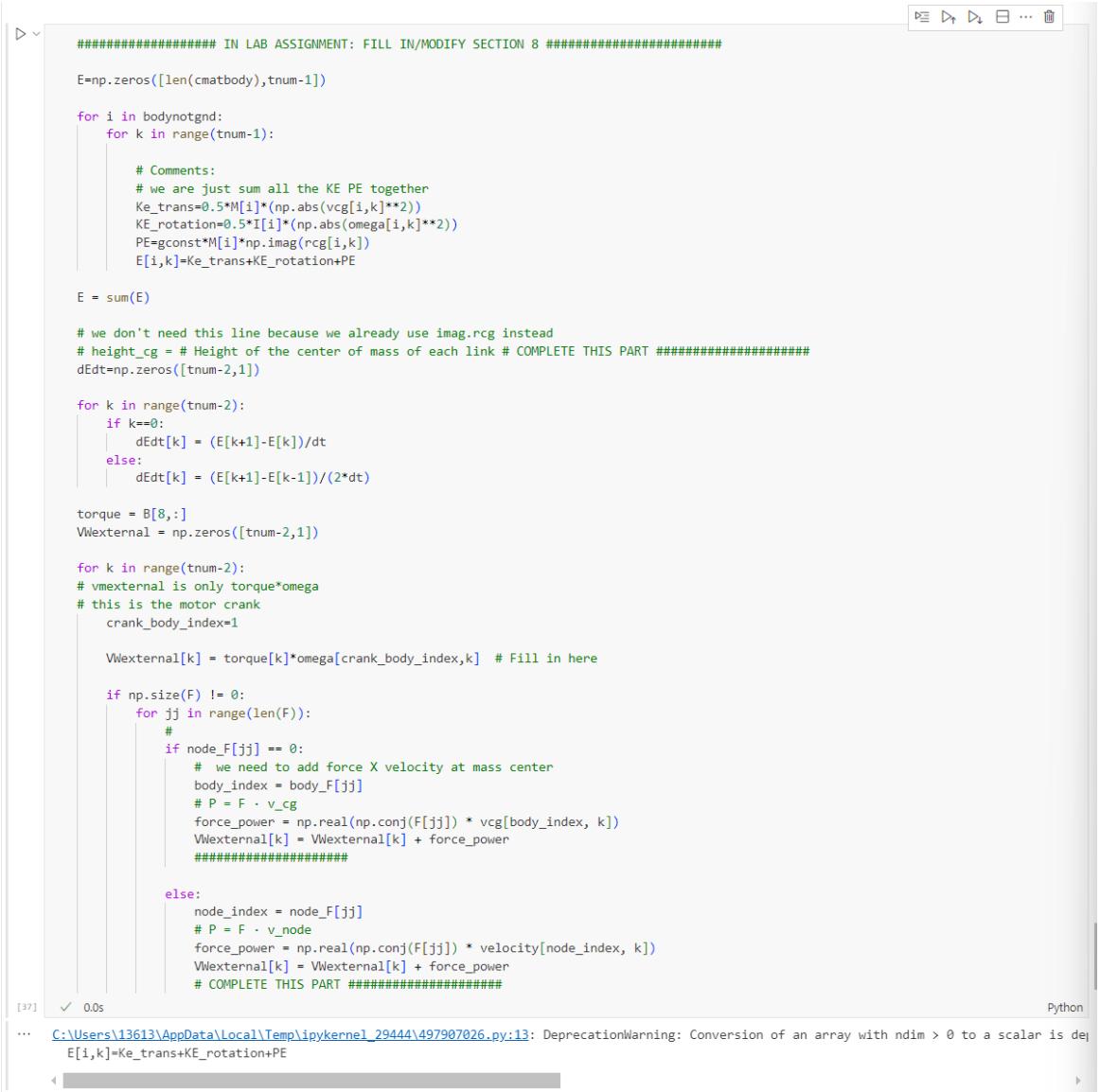
Listing 4: Python code for Section 7: Torque Equation Vectors.

Question 4

```
% --- Section 8  ---
E=np.zeros([len(cmatbody),tnum-1])

for i in bodynotgnd:
    for k in range(tnum-1):

        # Comments:
        # we are just sum all the KE PE together
        Ke_trans=0.5*M[i]*(np.abs(vcg[i,k]**2))
        KE_rotation=0.5*I[i]*(np.abs(omega[i,k]**2))
        PE=gconst*M[i]*np.imag(rcg[i,k])
        E[i,k]=Ke_trans+KE_rotation+PE

E = sum(E)

# we don't need this line because we already use imag.rcg instead
# height_cg = # Height of the center of mass of each link # COMPLETE THIS PART
    ######################
dEdt=np.zeros([tnum-2,1])

for k in range(tnum-2):
    if k==0:
        dEdt[k] = (E[k+1]-E[k])/dt
    else:
        dEdt[k] = (E[k+1]-E[k-1])/(2*dt)

torque = B[8,:]
VWexternal = np.zeros([tnum-2,1])

for k in range(tnum-2):
# vmexternal is only torque*omega
# this is the motor crank
    crank_body_index=1

    VWexternal[k] = torque[k]*omega[crank_body_index,k]  # Fill in here

    if np.size(F) != 0:
        for jj in range(len(F)):
            #
            if node_F[jj] == 0:
                #  we need to add force X velocity at mass center
                body_index = body_F[jj]
                # P = F    v_cg
                force_power = np.real(np.conj(F[jj]) * vcg[body_index, k])
                VWexternal[k] = VWexternal[k] + force_power
                ######################

            else:
                node_index = node_F[jj]
                # P = F    v_node
                force_power = np.real(np.conj(F[jj]) * velocity[node_index, k])
                VWexternal[k] = VWexternal[k] + force_power
```

Listing 5: Python code for Section 8: Virtual Work Analysis.

Screenshots



```
################### IN LAB ASSIGNMENT: FILL IN/MODIFY SECTION 8 ########################

E=np.zeros([len(cmatbody),tnum-1])

for i in bodynotgnd:
    for k in range(tnum-1):

        # Comments:
        # we are just sum all the KE PE together
        Ke_trans=0.5*M[i]*(np.abs(vcg[i,k]**2))
        KE_rotation=0.5*I[i]*(np.abs(omega[i,k]**2))
        PE=gconst*M[i]*np.imag(rcg[i,k])
        E[i,k]=Ke_trans+KE_rotation+PE

E = sum(E)

# we don't need this line because we already use imag.rcg instead
# height_cg = # Height of the center of mass of each link # COMPLETE THIS PART ####################
dEdt=np.zeros([tnum-2,1])

for k in range(tnum-2):
    if k==0:
        dEdt[k] = (E[k+1]-E[k])/dt
    else:
        dEdt[k] = (E[k+1]-E[k-1])/(2*dt)

torque = B[8,:]
VWexternal = np.zeros([tnum-2,1])

for k in range(tnum-2):
# vmexternal is only torque*omega
# this is the motor crank
    crank_body_index=1

    VWexternal[k] = torque[k]*omega[crank_body_index,k]  # Fill in here

    if np.size(F) != 0:
        for jj in range(len(F)):
            #
            if node_F[jj] == 0:
                #  we need to add force X velocity at mass center
                body_index = body_F[jj]
                # P = F · v_cg
                force_power = np.real(np.conj(F[jj]) * vcg[body_index, k])
                VWexternal[k] = VWexternal[k] + force_power
                ####################

            else:
                node_index = node_F[jj]
                # P = F · v_node
                force_power = np.real(np.conj(F[jj]) * velocity[node_index, k])
                VWexternal[k] = VWexternal[k] + force_power
                # COMPLETE THIS PART ####################
```

[37] ✓ 0.0s Python

```
C:\Users\13613\AppData\Local\Temp\ipykernel_29444\497907026.py:13: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is de
  E[i,k]=Ke_trans+KE_rotation+PE
```

Figure 1: Screenshot of completed code for power analysis.

Question 5

```
% --- input torque Section 9 ---
# Comments
# we can just use the torque variable with crank angle variable

plt.figure()
plt.plot(crank_angle[0:tnum-2], torque, linewidth = 2)
plt.title('Motor Torque vs Crank Angle')
plt.xlabel('Crank Angle (Degrees)')
plt.ylabel('Torque (Nm)')
plt.grid(1)
```

Listing 6: Python code for Power Analysis: Input Torque.

Screenshots

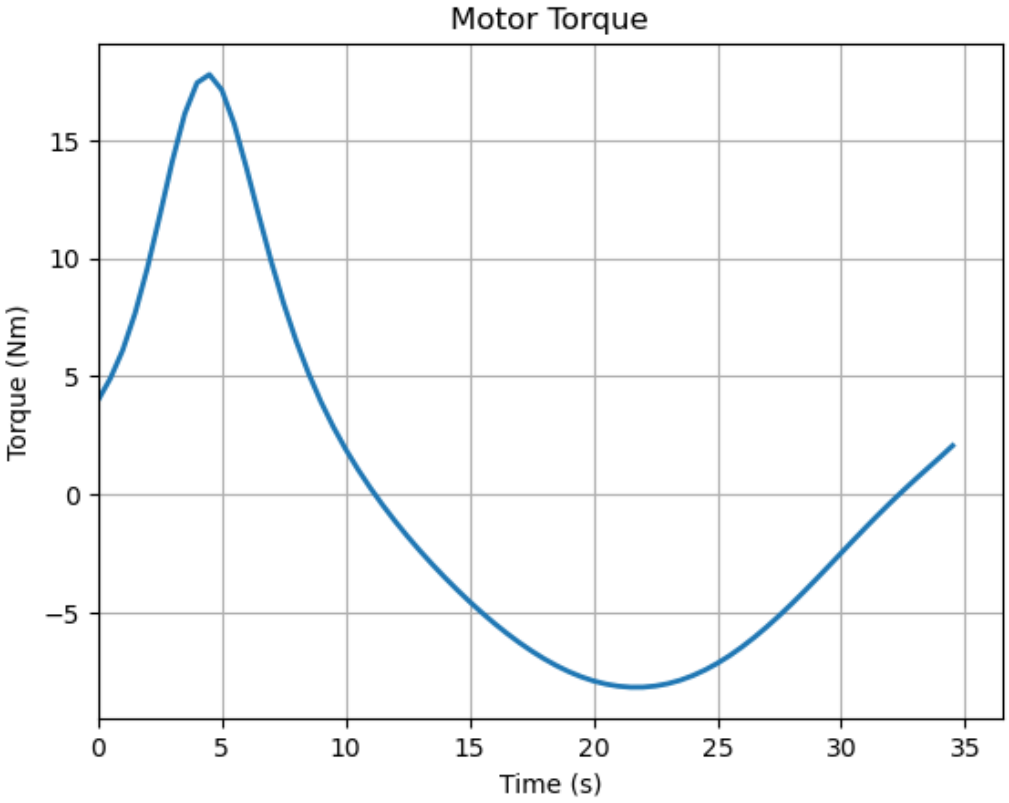


Figure 2: Torque vs Time

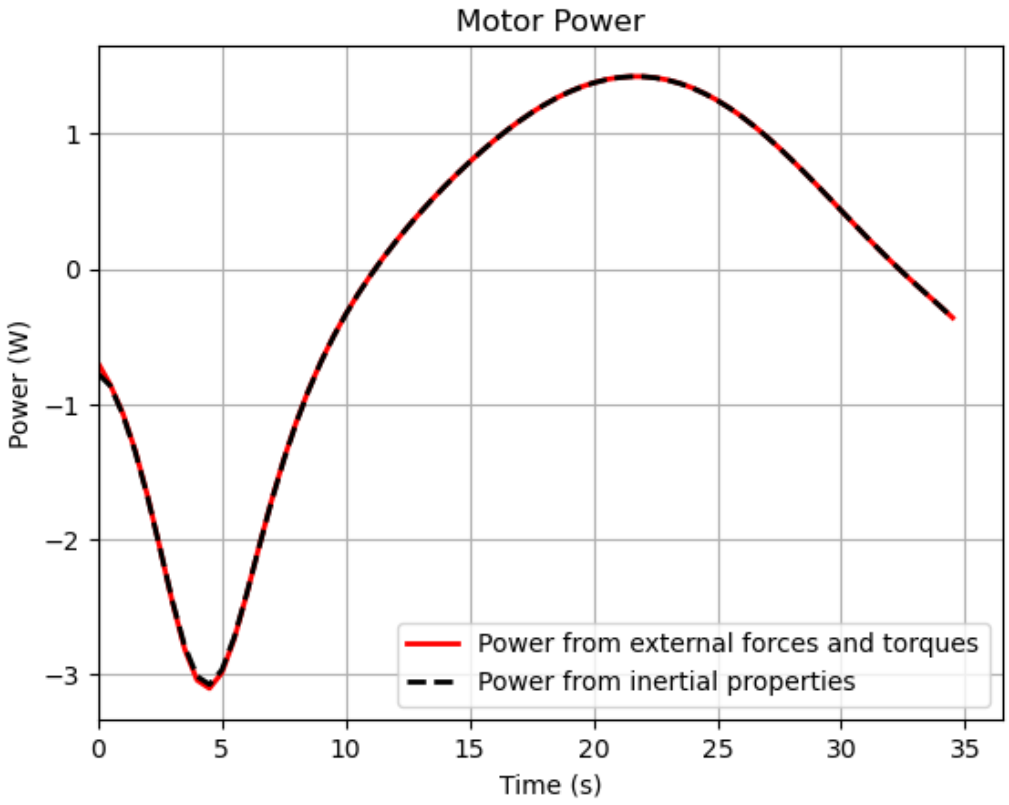


Figure 3: Two kinds of Power vs Time

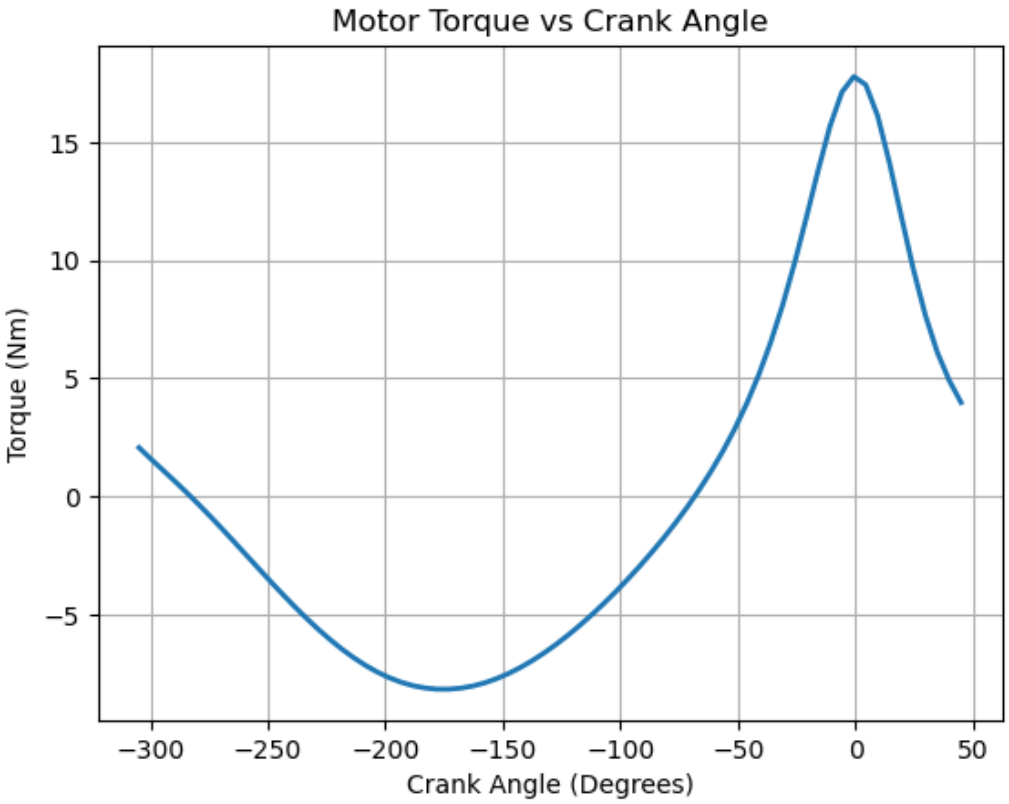


Figure 4: Toque vs Crank Angle

Postlab Q1: Plots of Slider Motion

The following plots show the position, velocity, and acceleration of the horizontal slider (node 5) as a function of time for four full cycles of the crank. The crank angular velocity was set to 40 degrees/second to complete one revolution in 9 seconds.

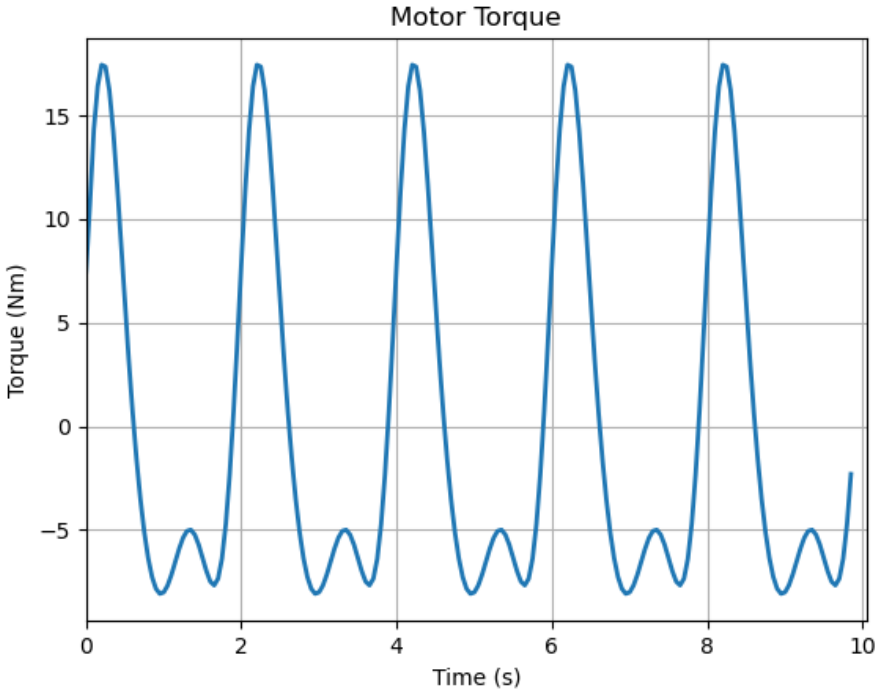


Figure 5: Position of the horizontal slider (Node 5) vs. Time.

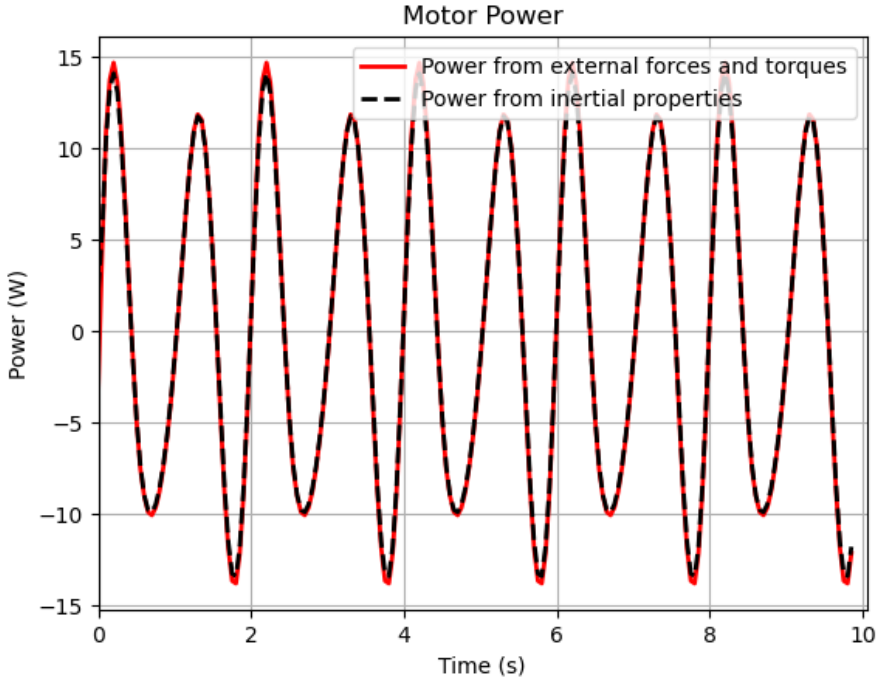


Figure 6: Velocity of the horizontal slider (Node 5) vs. Time.

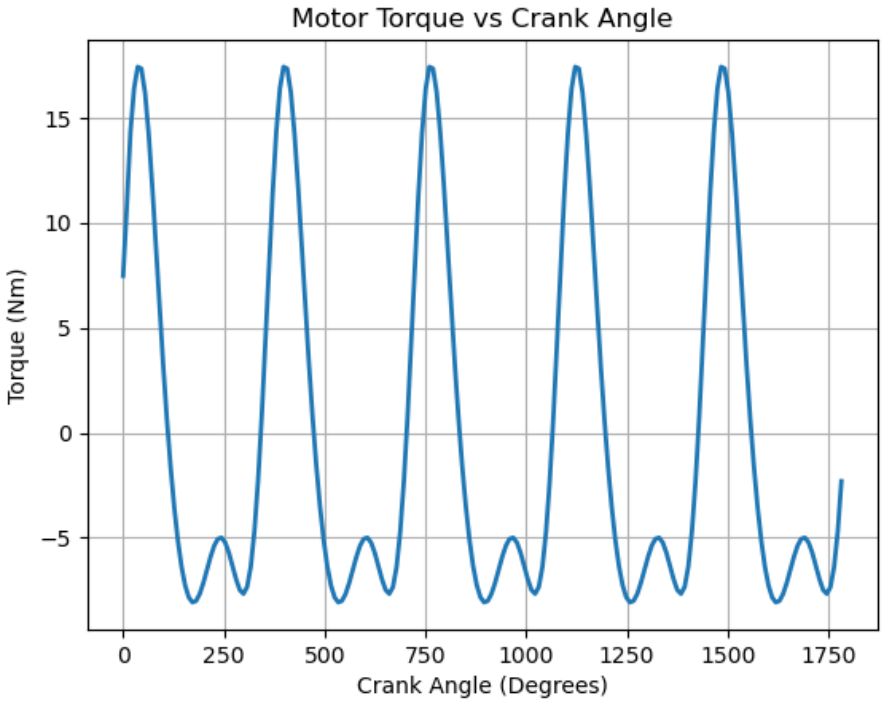


Figure 7: Acceleration of the horizontal slider (Node 5) vs. Time.

Postlab Q2: Modified Code and Commentary

Section1

```
cmlink = [0.5, 0, 0.5, 0.5, 0.5] #Fractional link distance from 1st node where the
    center of mass lies
#Defined in the order cmat is defined in the PVA
cmatbody = [[0], [1], [2,4], [3]] #Defining rigid bodies as a array
#Note that in PVA, each link was defined to be binary. Links modeled with a fixed
    angle constraint are physically rigid
gconst= 0#gravitational constant in m/s^2 (positive)
r=0.136
#If neglecting gravity, gconst = 0.
wdth=0.1#Width of each link (meters).
tks= 0.1#thickness of each link (meters).
den= 2500#Density for each link (kg/m^3).
```

Listing 7: Python code for Section 1 Input.

Section2

```
# Commments:
# we are measuring m of the circle
Mlink[crlink] = den*(np.pi*((r/2)**2)*link_lengths[crlink])

# Mlink[crlink] = # FILL IN HERE ################################
```

Listing 8: Python code Modification

Section3

```
1
2 # Comments:
3 # For the circle
4 # I = 0.5*m*r^2
5 Ilink[crlink] = 0.5*Mlink[crlink]*((r/2)**2) # FILL IN HERE
    ##########################################
```

Listing 9: Python code Modification

Section4

```
1 # Comments:
2 # For the circle, we use the crangvel to calculate omega and we have to transform it
    to radians/sec
3
4 for i in range(len(cmatbody)):
5     if cmatbody[i][0] == crlink:
6         omega[i] = crangvel * np.pi / 180.0    # FILL IN HERE
              ###############################
```

Listing 10: Python code Modification

Section6

```
1
2 F = [80*np.cos(np.pi/4)+80*np.sin(np.pi/4)*1j] #F1 as a complex number
3 body_F = [2] #Link Index where F1 Acts
4 node_F = [4] #Node where F1 acts
```

Listing 11: Python code Modification

Section8

```
1 for k in range(tnum-2):
2
3     # Comments:
4     # The same in in-lab activity Firstly add torque x omega
5
6
7     P_motor = torque[k] * omega[crlink, k]
8     VWexternal[k] = P_motor
9
10    ##################### Fill in here
11    if np.size(F) != 0:
12        for jj in range(len(F)):
13            if node_F[jj] == 0:
14                # Comments:
15                # The same in in-lab activity, add F x V
16                # If force acts on a node
17                P_force = np.real(np.conj(F[jj]) * vcg[body_F[jj], k])
18                VWexternal[k] = VWexternal[k] + P_force
19                ##################### Fill in here
20            else:
21                # If force acts on a node
22                P_force = np.real(np.conj(F[jj]) * velocity[node_F[jj], k])
23                VWexternal[k] = VWexternal[k] + P_force
24
25                ##################### Fill in here
```

Listing 12: Python code Modification

Postelab Q3. AI Statement

Select one of the following options:
a) My answer was created by a Gen AI algorithm, and I have not modified it
b) My answer was created by a Gen AI algorithm, and I have some minor changes.
c) My answer was created by a Gen AI algorithm, and I have made major changes.
d) My answer was created solely by myself.