Homework3_SUN

Shixuan Sun

***Abstract*— This is HW3 for 263E**

I. QUESTION1

Pseudocode and explanations of functions are fellows:

Because the problem states the middle node of the rod has to follow a quarter circle path, the simplest approach is to have to end effector sweep the right end side of the rod from [1,0] to [0,-1]. Below are the pseudocode for main, hessEb, gradEb, and getFb, pseudocode for hessEs, gradEb, and getFs were discussed in Hw1.

Figure1. Pseudocode for main.py

```
Pseudocode for HW3

Import necessary toolbaox and functions
set number of node to be 51
set dof to be 2* # of node, and set the middle node
dt = 1
rod length = 1
dl = 1m/ndof
pre-allocate an vector for R that is nv-1 long
      for every entry of R, let it be delta/10
set up the material properties
set up the running time, iterations
calcualte the moment of interia, and the area of moment
set the tolerance
initialize matrix called  nodes that has size of nv by 2
for evrery row in nodes,
      the first entry is x coordinates, and second entry is y coordinate

initialize an arrey called m, that has of 2*nv
for every entry in the array, assign mass to them
      make a diagional matry whoseebtry is array m

initialize an array called W that has size of 2*nv
initialize an array called g that has entry[0,-9.81]
for every entry in array W
      assign even number to be 0, and odd number to have value of -9.81
initialize an array called qo that has size of 2*nv
for every entry in the array
      assign even entry to have the same value as even nodes array
      assign odd entry to have the same value as odd nodes array

initialize an array called all_DOF that range form 0 to ndof-1
initialize an array called fixed_index that is has value [0,1]
initialize an array called free_indx that inly contain dof that is not fixed
initialize a variable called Nsteps that has value totaltime/dt
create a variable called ctime
intiailize all_pos, all_vel, and mid_angle arrays that has the length of Nsteps
set the first entry to be 0 for all_pos, all_vel, and mid_angle arrays
initialize y_min_series, x_at_ymin arrays to have the size of Nsteps
create iN and iNm to represent the last one and last two node
assign ixN,iyN to the the x and y position of last node
assign ixNm and iyNm to the x and y position of the last 2 node

define a function called end_eff_traj that take t as input
      assign variable phi as an variable goes from 0 to 2pi as a function of time
      x_c = cos(phi)
      y_c = -sin(phi)
      theat_c = atan2(x_c,y_c)
      return x_c,y_c, and theat_c
initialize xc_series,yc_series, and th_series arrays to have size of Nsteps
make the first enerty of the initialize xc_series,yc_series, and th_series to have value of
end_eff_traj(0)

initialize a set that has range form 0 to 1000 and have interval of 1

for number form 1 to Nsteps
      evaluate end_eff_traj at ctime, and store in xc,yc,and th
      store xc, yc, xc-dl*cos(th), and yc-dl*sin(th) in q0[ixN], q0[iyN], q0[ixNm], and q0[iyNm]
      store nodes 0,1, ixNm, iyNm, ixN, iyN in fixed_index
      all other nodes are store in free_index
      store current xc,yc,and th in the xc_series, yc_series,  th_series
      call objfun to obtain q_new and error
      calculate u_new using derivative
      advance ctime by dt
      store even number in q_new to x_arr
      store odd number in q_new to y_arr
      if timeStep is multiple of plotStep
            plot x_arr vs_yrr
      copy q_new and u_new to q0 and u0
create variable called t_arr that start form 0 end in 1001 and have total number of Nstep

plot the time vs. xc_series
plot the time vs. yc_series
plot the time vs. theat
plot the shape of rod every 100 s
```

Figure2. Pseudocode for hessEb.py

```
Pseudocode for hessEb

define function hessEb(xkm1, ykm1, xk, yk, xkp1, ykp1, curvature0, l_k, EI)
set node0 = [xkm1, ykm1, 0]
set node1 = [xk, yk, 0]
set node2 = [xkp1, ykp1, 0]
set m2e = [0, 0, 1], m2f = [0, 0, 1]
set ee = node1 - node0, ef = node2 - node1
set norm_e = ||ee||, norm_f = ||ef||
set te = ee / norm_e, tf = ef / norm_f
set chi = 1 + dot(te, tf)
set tilde_t = (te + tf) / chi
set tilde_d2 = (m2e + m2f) / chi
set kb = 2 * cross(te, tf) / chi
set kappa1 = kb[2]
set Dkappa1De = (-kappa1 * tilde_t + cross(tf, tilde_d2)) / norm_e
set Dkappa1Df = (-kappa1 * tilde_t - cross(te, tilde_d2)) / norm_f
initialize gradKappa as length 6 zero array
gradKappa[0:2] = -Dkappa1De[0:2]
gradKappa[2:4] = Dkappa1De[0:2] - Dkappa1Df[0:2]
gradKappa[4:6] = Dkappa1Df[0:2]
set norm2_e = norm_e^2, norm2_f = norm_f^2
set Id3 = identity(3)
compute helper matrices
tt_o_tt = outer(tilde_t, tilde_t)
tmp = cross(tf, tilde_d2)
tf_c_d2t_o_tt = outer(tmp, tilde_t)
kb_o_d2e = outer(kb, m2e)
compute D2kappa1De2
compute D2kappa1Df2
compute D2kappa1DeDf
set D2kappa1DfDe = transpose(D2kappa1DeDf)
initialize 6×6 zero matrix DDkappa1
fill 2×2 blocks of DDkappa1 using the xy parts of D2kappa1De2, D2kappa1Df2, D2kappa1DeDf,
D2kappa1DfDe
set dkappa = kappa1 - curvature0
set dJ = (EI / l_k) * outer(gradKappa, gradKappa)
set dJ += (EI / l_k) * dkappa * DDkappa1
return dJ
```

Figure3. Pseudocode for gradEb.py

```
pseudocode for gradEb

define function gradEb(xkm1, ykm1, xk, yk, xkp1, ykp1, curvature0, l_k, EI)
set node0 = [xkm1, ykm1, 0]
set node1 = [xk, yk, 0]
set node2 = [xkp1, ykp1, 0]
set m2e = [0,0,1]
set m2f = [0,0,1]
initialize gradKappa as zeros(6)
set ee = node1 - node0
set ef = node2 - node1
set norm_e = ||ee||
set norm_f = ||ef||
set te = ee / norm_e
set tf = ef / norm_f
set chi = 1 + dot(te, tf)
set tilde_t = (te + tf) / chi
set tilde_d2 = (m2e + m2f) / chi
set kb = 2 * cross(te, tf) / chi
set kappa1 = kb[2]
set Dkappa1De = ( -kappa1 * tilde_t + cross(tf, tilde_d2) ) / norm_e
set Dkappa1Df = ( -kappa1 * tilde_t - cross(te, tilde_d2) ) / norm_f
set gradKappa[0:2] = -Dkappa1De[0:2]
set gradKappa[2:4] = Dkappa1De[0:2] - Dkappa1Df[0:2]
set gradKappa[4:6] = Dkappa1Df[0:2]
set dkappa = kappa1 - curvature0
set dF = (EI / l_k) * dkappa * gradKappa
return dF
```

Figure4. Pseudocode for getFb.py

```
Pseudocode for getFb

define function getFb(q, EI, deltaL)
set ndof = size(q)
set N = ndof // 2
initialize Fb = zeros(ndof)
initialize Jb = zeros(ndof, ndof)
for k from 1 to N-2
  set xkm1 = q[2*k-2]
  set ykm1 = q[2*k-1]
  set xk = q[2*k]
  set yk = q[2*k+1]
  set xkp1 = q[2*k+2]
  set ykp1 = q[2*k+3]
  set ind = [2*k-2, 2*k-1, 2*k, 2*k+1, 2*k+2, 2*k+3]
  set gradEnergy = gradEb(xkm1, ykm1, xk, yk, xkp1, ykp1, curvature0=0, l_k=deltaL,
EI)
  set hessEnergy = hessEb(xkm1, ykm1, xk, yk, xkp1, ykp1, curvature0=0, l_k=deltaL,
EI)
  set Fb[ind] = Fb[ind] - gradEnergy
  set Jb[ind, ind] = Jb[ind, ind] - hessEnergy
end for
return Fb, Jb
```

II. QUESTION 2

Plots of the control inputs $xc(t), yc(t), \theta c(t)$ over time.

Figure5. xc vs.time

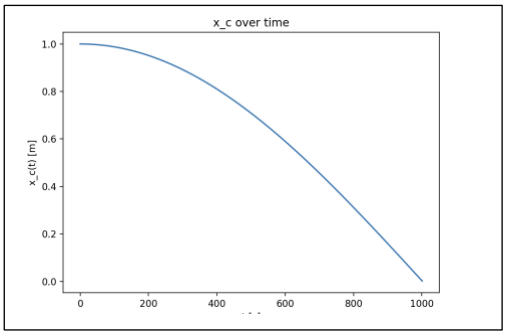


Figure6. yc vs.time

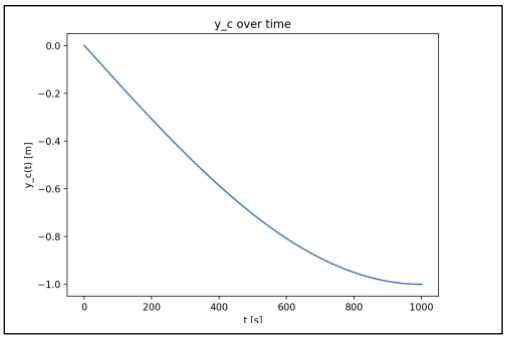
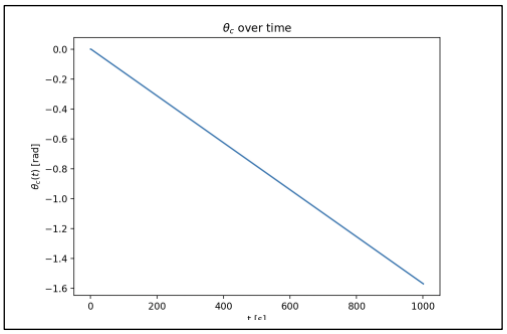


Figure7. θc vs.time



III. QUESTION 3

Five snapshots of the beam shape (node positions) during the motion :

Figure8. shape of the rod when t = 200s

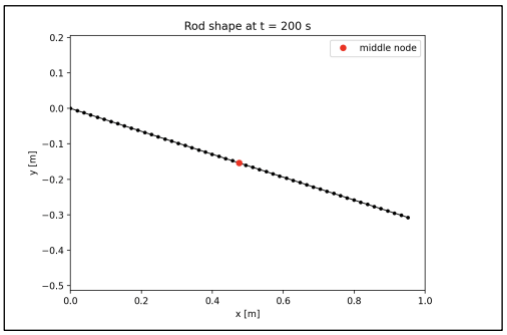


Figure9. shape of the rod when t = 400s

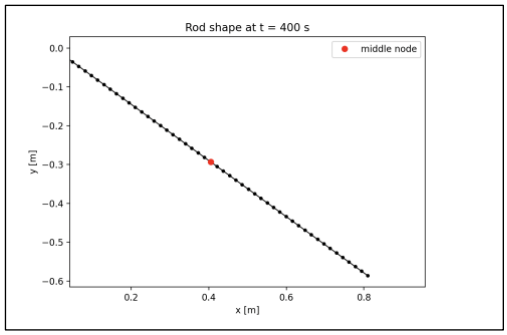


Figure10. shape of the rod when t = 600s

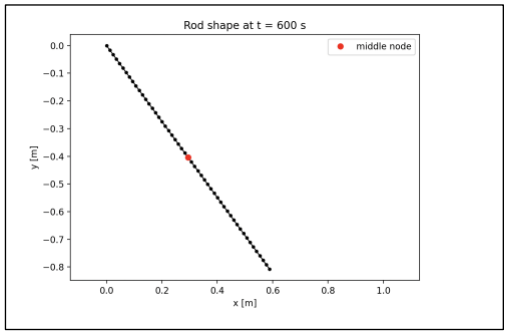


Figure11. shape of the rod when t = 800s

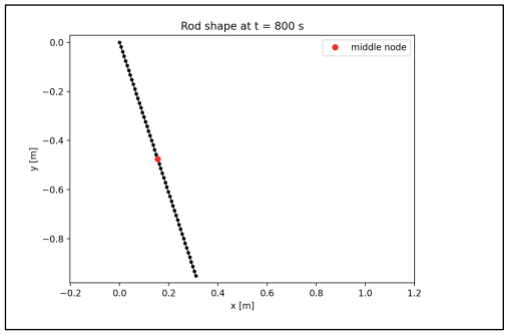
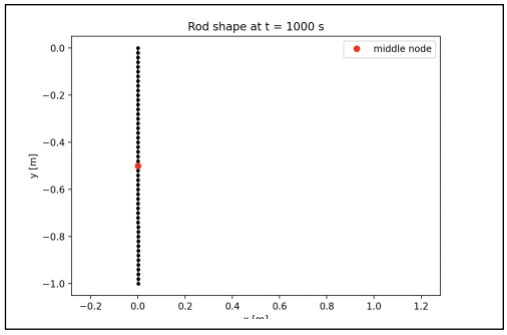


Figure12. shape of the rod when t = 1000s



IV. QUESTION 4

From the drawing and problem statement, a 4R planar robotic arm is used to accomplish the task, whose workspace is roughly a solid circle, with the exception of its base to the tool tip when the robotic arm is pointing straight up. My

approach is to let the tool tip flow a quarter circular path that goes from [1,0] to [0,-1] poses no interference with the robotic arm itself. If the position of the right end of the rod is too far (outside of the workspace) or too close (in the exception of the workspace), the tool tip will not be able to reach the rod, thus unable to accomplish the task.

ACKNOWLEDGMENT

The source code is taken from Dr.khalid's Google Colab notebook shared on Slack, then modified by Shixuan Sun to accomplish HW3