ME 572: Fall 2022 Computer Project

TJ Wiegman

2022-11-21

In [1]:
```
# Read text file input
with open("RR_HW5.txt") as file:
    RR = file.readlines()

for i in range(len(RR)):
    RR[i] = RR[i].rstrip() # rstrip removes newline characters

    # For documentation purposes: print contents of input file here
    print(RR[i])
```

```
.25,.01,2,2,RR_HW5
3,.01,0
-2,.01,0
```

In [2]:
```
# Parse the input
## First line
timeTotal, timeStep, Sacc, Sdec, Title = RR[0].split(",")
timeTotal, timeStep = float(timeTotal), float(timeStep)
Sacc, Sdec = int(Sacc), int(Sdec)

## Second line
P0 = []
for coord in RR[1].split(","): P0.append(float(coord))

## Third line
PF = []
for coord in RR[2].split(","): PF.append(float(coord))

## For documentation purposes: print results
print(
    f"{Title}: Moving from {P0} to {PF} "
    f"over {timeTotal} seconds in {timeStep} second steps"
)
```

```
RR_HW5: Moving from [3.0, 0.01, 0.0] to [-2.0, 0.01, 0.0] over 0.25 seconds in 0.01 second st
eps
```

In [3]:
```
# Import inverse kinematics and jacobian based on input file
from importlib import import_module
robot = import_module(Title)
IK = robot.inverseKinematics
jac = robot.jacobian

# For documentation purposes: print out the code from that file here
fileTitle = Title + ".py"
print(fileTitle)
print("==========")
print(open(fileTitle).read())
```

```
RR_HW5.py
==========
from math import sqrt, atan2, acos, sin, cos

def inverseKinematics(px, py, pz):
    L1, L2 = 20, 20
    a = (px*px + py*py + L1*L1 - L2*L2) / (2*L1)
    b = sqrt(px*px + py*py)
    th1 = atan2(py, px) + acos(a/b)
    num = py - (L1 * sin(th1))
    den = px - (L1 * cos(th1))
    th2 = atan2(num, den) - th1
    th3 = 0
    return [th1, th2, th3]

from numpy import matrix
from numpy.linalg import inv

def jacobian(th1, th2, th3, vx, vy, vz, px, py, pz):
    L1, L2 = 20, 20
    th12 = th1 + th2

    A = (-L1 * sin(th1))  + (-L2 * sin(th12))
    B = (-L2) * sin(th12)
    C = (L1 * cos(th1)) + (L2 * cos(th12))
    D = L2 * cos(th12)

    J = matrix([[A, B],
                [C, D]])
    Jinv = inv(J)
    vxy = matrix([[vx], [vy]])
    output = Jinv @ vxy # the "@" is matrix multiplication

    th1d = output[0,0]
    th2d = output[1,0]
    th3d = 0
    return [th1d, th2d, th3d]
```

In [4]:

```
# Calculate top velocities
assert (timeTotal / timeStep) % 1 == 0, "Total time must divide evenly by timestep!"
S = int(timeTotal / timeStep)
assert (Sacc + Sdec) <= S, "Cannot accelerate + decelerate longer than total time!"
Smod = S - 0.5*Sacc - 0.5*Sdec

vMax = [0, 0, 0]
for i in range(3):
    vMax[i] = (PF[i] - P0[i]) / (Smod * timeStep)

print(vMax)
```

```
[-21.73913043478261, 0.0, 0.0]
```

In [5]:

```
# Calculate velocity curves
velocity = [[0], [0], [0]]
for i in range(3):
    # Acceleration
    for j in range(Sacc):
        velocity[i].append(vMax[i] * ((j+1) / Sacc))
```

```python
    # Steady state
    for j in range(S - (Sacc + Sdec)):
        velocity[i].append(vMax[i])

    # Deceleration
    for j in range(Sdec):
        velocity[i].append(vMax[i] * ((Sdec - j - 1)/Sdec))
```

In [6]:
```python
# Calculate position curves
position = [[P0[0]], [P0[1]], [P0[2]]]
for i in range(3):
    for t in range(1,S):
        position[i].append(position[i][-1] + velocity[i][t]*timeStep)
    position[i].append(PF[i])
```

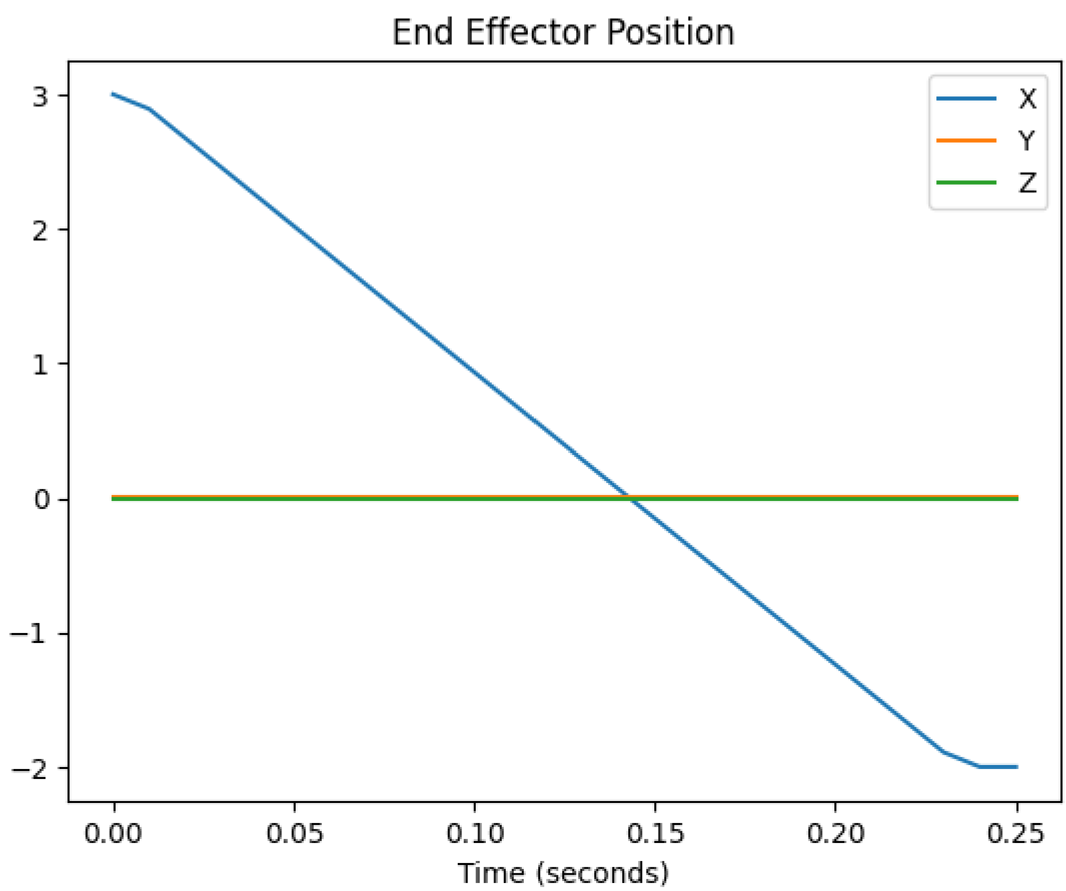
In [7]:
```python
# Make position plot
import matplotlib.pyplot as plt

time = [0]
for _ in range(S): time.append(time[-1]+timeStep)

for i in range(3): plt.plot(time, position[i])

plt.title("End Effector Position")
plt.legend(["X", "Y", "Z"])
plt.xlabel("Time (seconds)")
```

Out[7]: `Text(0.5, 0, 'Time (seconds)')`



In [8]:
```python
# Calculate joint angles
from math import degrees
```

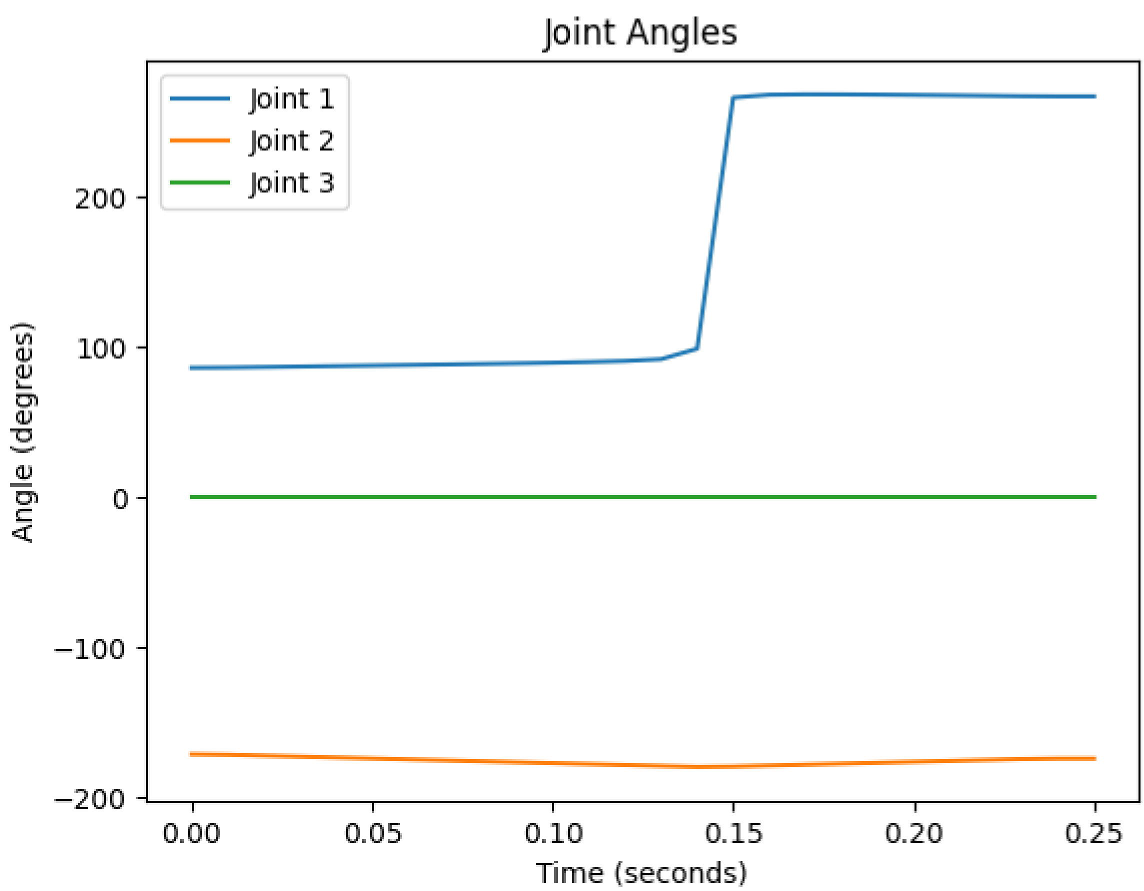
```python
th1, th2, th3 = IK(P0[0], P0[1], P0[2])
thetas = [[degrees(th1)], [degrees(th2)], [degrees(th3)]]

for i in range(S):
    th1, th2, th3 = IK(position[0][i+1], position[1][i+1], position[2][i+1])
    thetas[0].append(degrees(th1))
    thetas[1].append(degrees(th2))
    thetas[2].append(degrees(th3))
```

In [9]:
```python
# Make angle plots
for i in range(3): plt.plot(time, thetas[i])

plt.title("Joint Angles")
plt.legend(["Joint 1", "Joint 2", "Joint 3"])
plt.ylabel("Angle (degrees)")
plt.xlabel("Time (seconds)")
```

Out[9]:
```
Text(0.5, 0, 'Time (seconds)')
```



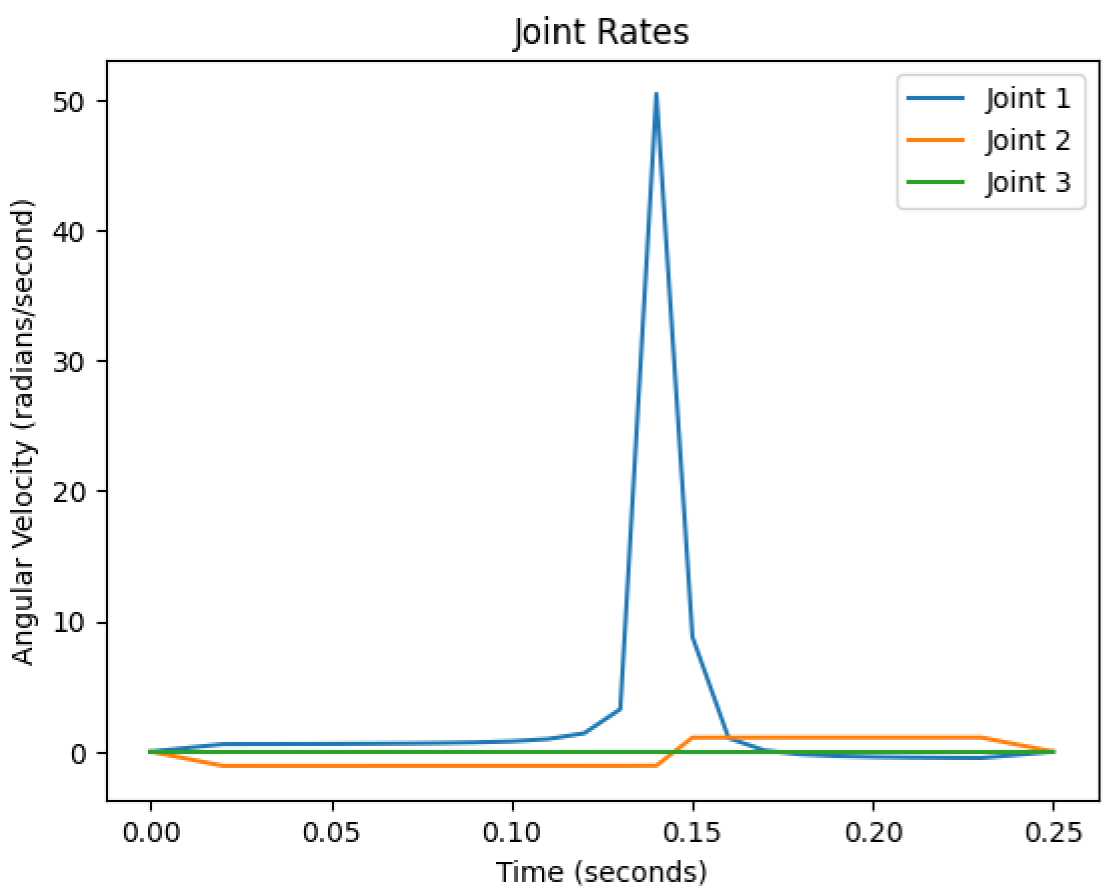
In [10]:
```python
# Get joint rates
from math import radians
dotThetas = [[0], [0], [0]]
for i in range(S):
    dth1, dth2, dth3 = jac(
        radians(thetas[0][i+1]),
        radians(thetas[1][i+1]),
        radians(thetas[2][i+1]),
        velocity[0][i+1],
        velocity[1][i+1],
        velocity[2][i+1],
```

```
        position[0][i+1],
        position[1][i+1],
        position[2][i+1]
    )
    dotThetas[0].append(dth1)
    dotThetas[1].append(dth2)
    dotThetas[2].append(dth3)
```

In [11]:
```
# Plot joint rates
for i in range(3): plt.plot(time, dotThetas[i])

plt.title("Joint Rates")
plt.legend(["Joint 1", "Joint 2", "Joint 3"])
plt.ylabel("Angular Velocity (radians/second)")
plt.xlabel("Time (seconds)")
```

Out[11]:
```
Text(0.5, 0, 'Time (seconds)')
```



In [12]:
```
# Generate output report
report = [",".join([
    "Time",
    "Px",
    "Py",
    "Pz",
    "Theta 1",
    "Theta 2",
    "Theta 3",
    "Theta-dot 1",
    "Theta-dot 2",
    "Theta-dot 3"
])]
```

```python
for i in range(len(time)):
    report.append(",".join([
        str(time[i]),             # Time
        str(position[0][i]),      # Px
        str(position[1][i]),      # Py
        str(position[2][i]),      # Pz
        str(thetas[0][i]),        # Theta 1
        str(thetas[1][i]),        # Theta 2
        str(thetas[2][i]),        # Theta 3
        str(dotThetas[0][i]),     # Theta-dot 1
        str(dotThetas[1][i]),     # Theta-dot 2
        str(dotThetas[2][i])      # Theta-dot 3
    ]))
```

In [13]:

```python
# Save the report to a file
outputTitle = Title + ".csv"
with open(outputTitle, mode = "w") as file:
    file.write("\n".join(report))

# For documentation purposes: print the file here
with open(outputTitle, mode = "r") as file:
    print(file.read())
```

```
Time,Px,Py,Pz,Theta 1,Theta 2,Theta 3,Theta-dot 1,Theta-dot 2,Theta-dot 3
0,3.0,0.01,0.0,85.8897389790859,-171.3975075094524,0.0,0,0,0
0.01,2.891304347826087,0.01,0.0,86.05303692087429,-171.70974371416503,0.0,0.2854524666942258,
-0.5449003826825275,0
0.02,2.6739130434782608,0.01,0.0,86.38129244970918,-172.33403309939018,0.0,0.575097612737642
7,-1.0893857117792778,0
0.03,2.4565217391304346,0.01,0.0,86.71228515974431,-172.95809398334686,0.0,0.580525706008401
5,-1.0890031110864664,0
0.04,2.2391304347826084,0.01,0.0,87.04685481133592,-173.5819448974027,0.0,0.5876849154541973,
-1.0886527368690893,0
0.05,2.0217391304347823,0.01,0.0,87.38619828195013,-174.2056042274859,0.0,0.5973511889758886,
-1.0883343015828213,0
0.060000000000000005,1.804347826086956,0.01,0.0,87.73208470599124,-174.8290901653967,0.0,0.61
0794767129871,-1.0880474029656548,0
0.07,1.58695652173913,0.01,0.0,88.08724742768845,-175.45242061586956,0.0,0.6302124156581418,-
1.0877914158970188,0
0.08,1.3695652173913038,0.01,0.0,88.45614920697545,-176.07561301027067,0.0,0.659674660307004
1,-1.0875652554863033,0
0.09,1.1521739130434776,0.01,0.0,88.8466135897912,-176.69868390367833,0.0,0.7074304103862107,
-1.087366800742035,0
0.09999999999999999,0.9347826086956514,0.01,0.0,89.27373221645989,-177.32164800281865,0.0,0.7
923503001158292,-1.087191283288173,0
0.10999999999999999,0.7173913043478253,0.01,0.0,89.77087442779911,-177.94451541789812,0.0,0.9
658367206583467,-1.0870258081112059,0
0.11999999999999998,0.49999999999999917,0.01,0.0,90.42940370565952,-178.56728173496867,0.0,1.
4126296009038877,-1.0868241410493538,0
0.12999999999999998,0.28260869565217306,0.01,0.0,91.62147975483518,-179.18987228711234,0.0,3.
2616416657611866,-1.0863038336627175,0
0.13999999999999999,0.06521739130434695,0.01,0.0,98.62294810083199,-179.8109822724644,0.0,50.
47423650866489,-1.074401194835588,0
0.15,-0.15217391304347916,0.01,0.0,266.02181080366495,-179.56311251933772,0.0,8.8050771199573
92,1.0846250397665111,0
0.16,-0.36956521739130527,0.01,0.0,267.92045734783756,-178.94087104171322,0.0,1.0472283327240
108,1.086605228335021,0
0.17,-0.5869565217391314,0.01,0.0,268.1830386272387,-178.318189030623,0.0,0.0873603373404054
2,1.0869158653850544,0
0.18000000000000002,-0.8043478260869574,0.01,0.0,268.1354006809083,-177.69537978292811,0.0,-
0.20758642683166742,1.0870923728960848,0
0.19000000000000003,-1.0217391304347836,0.01,0.0,267.9754878864581,-177.0724743718893,0.0,-0.
3354105527240865,1.0872592592221162,0
0.20000000000000004,-1.2391304347826098,0.01,0.0,267.7623576077636,-176.44946912529116,0.0,-
0.4021487305840946,1.0874430710139784,0
0.21000000000000005,-1.456521739130436,0.01,0.0,267.5198079385294,-175.82635153283874,0.0,-0.
44135824000381785,1.0876522420020187,0
0.22000000000000006,-1.673913043478262,0.01,0.0,267.25927057922695,-175.2031059647499,0.0,-0.
4663630940066193,1.0878901561873666,0
0.23000000000000007,-1.8913043478260883,0.01,0.0,266.9869173426723,-174.57971544340467,0.0,-
0.48330669394514697,1.0881584111526907,0
0.24000000000000007,-2.0000000000000013,0.01,0.0,266.8475036526389,-174.26796032583206,0.0,-
0.24490281644795678,0.5441521003211889,0
0.25000000000000006,-2.0,0.01,0.0,266.847503652639,-174.26796032583206,0.0,0.0,0.0,0
```