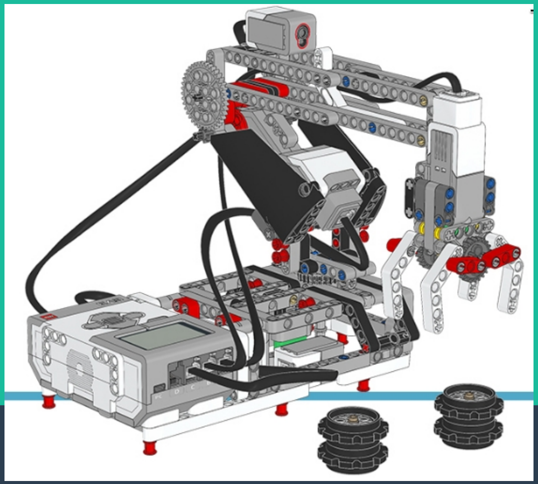
Let go of my (bot)Arm!



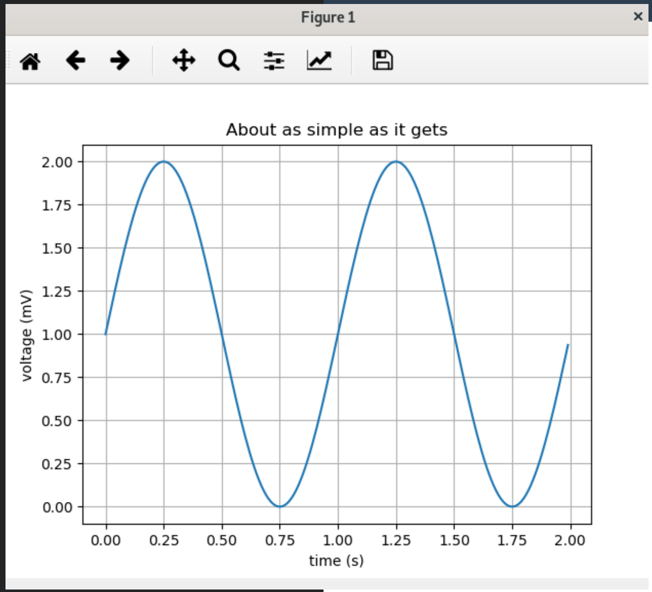
Nasri Academy
Thurs. Oct. 17th, 2019
By Julio B. Figueroa

Overview

- Review
- 1 DoF Arm
- 2 DoF Arm
- 3 Dof Arm

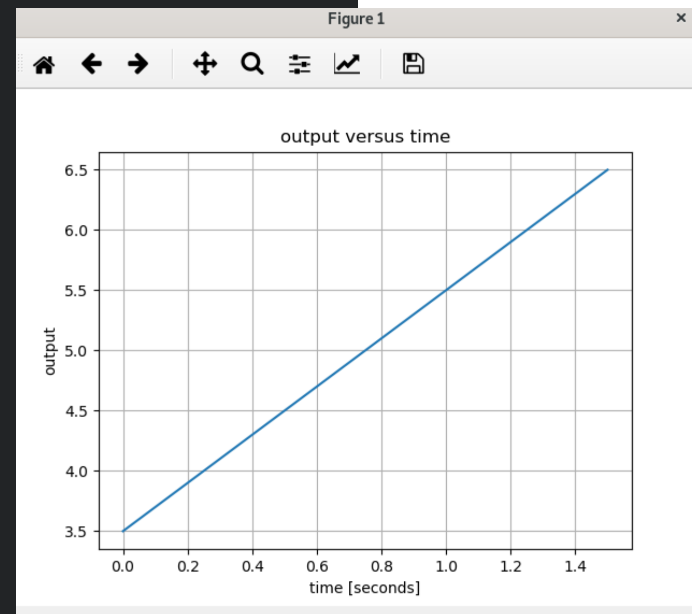
Review: Trig. Function

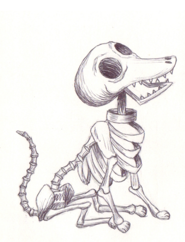
```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1+np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets')

ax.grid()

fig.savefig("test.png")
plt.show()
# soh-cah-toa
```



Review: Line Function

```
# necessary libraries
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# define descrete time values
t = np.arange(0.0, 2.0, 0.5)

# Parameters for 2nd line
m = 2.0
b = 3.5
y = m*t+b

fig, ax = plt.subplots()
ax.plot(t, y)

ax.set(xlabel='time [seconds]', ylabel='output',
       title='output versus time')
# y is the dependent value, t is the indepedent values
# some people will call this "Y versus T"

ax.grid()

fig.savefig("test.png")
plt.show()
# run with F5 or ctrl+shift+b
```



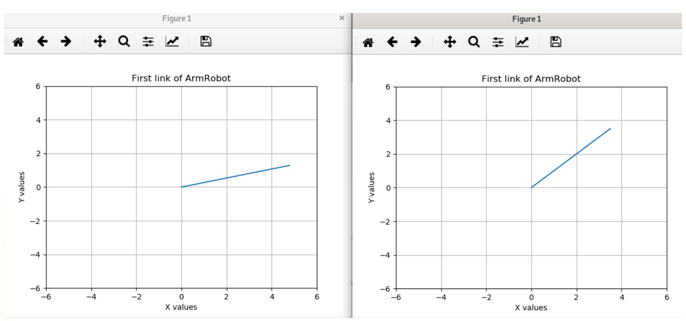


1 Degree of Freedom Linkage

Welcome Guide | lineExampleUS2_01b.py | lineExampleUS2_03.py | lineExampleUS2_02.py

```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# this example will plot two linear functions sharing the same plot
# this is useful so you can compare the two

# define descrete time values
# both linear functions will share the independent value t
# t = np.arange (0.0, 2.0, 0.5)

# theta_A = np.pi/4.0   # [radians] or 45 [degrees]
theta_A = np.deg2rad(45)
# again, t is the shared independent value in both functions

# point A
x_A = 0
y_A = 0

# point B
l_1 = 5  # [cm] from point A to point B
x_B = l_1 * np.cos(theta_A)
y_B = l_1 * np.sin(theta_A)

```

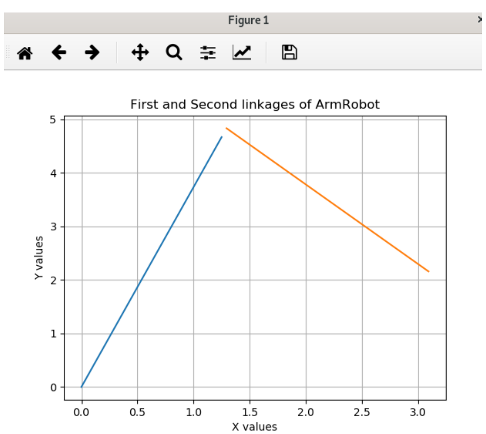
```python

# plot line AB
# You have the coordinates for two points
# pointA(x_A, x_B)
# pointB(y_B, y_B)
# therefore, to define this line we must use point slope form
# -----------------------------
# ClassWork: plot first linkage of the robot
# use the point slope form that you learned in math class to get your linear
# when you finish, mail to jfigueroa@nasriacademy.org
# -------------------------------
# Solution is provided below
m = (y_B - y_A) / (x_B - x_A)

# m * (x_B - x_A)     = (y_B - y_A)
# m * x_B - m * x_A = y_B - y_A
# y_B = m * x_B - m * x_A + y_A
# y_B = m * x_B + (-m * x_A + y_A)

b_intercept = (-m * x_A + y_A)
slope = m
x = np.arange(x_A, x_B, 0.1)
print(x_A, x_B, x)
y = slope * x + b_intercept
```



```python

# ClassWork: plot first linkage of the robot
# use the point slope form that you learned in math class to get your linear
# when you finish, mail to jfigueroa@nasriacademy.org
# ----------------------------

fig, ax = plt.subplots()
# plot your points below
ax.plot(x, y)
plt.xlim(-6, 6)  # fixes the coordinate axis so you can distinguish between lines
plt.ylim(-6, 6)
ax.set(xlabel='X values', ylabel='Y values',
       title='First link of ArmRobot')
ax.grid()
fig.savefig("RobotArmPart1.png")
plt.show()
```

2 DoF Arm

```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# this example will plot two linear functions sharing the same plot
# this is useful so you can compare the two

# theta_ = np.pi/3.0   # [radians] or 45 [degrees]
theta_degA = 75.0  # [degrees]
theta_degB = 25.0  # [degrees]

# point A
x_A = 0.0
y_A = 0.0

# point B
l_1 = 5.0  # [cm] from point A to point B
#x_B = l_1 * np.cos(theta_A)
#y_B = l_1 * np.sin(theta_A)
# you can also use np.deg2rad() to convert from degrees to radians
x_B = l_1 * np.cos(np.deg2rad(theta_degA))
y_B = l_1 * np.sin(np.deg2rad(theta_degA))

# point C

```



```python

l_2 = 2.0  # [cm] from point B to C
x_C = x_B + l_2 * np.cos(np.deg2rad(theta_degB))
y_C = x_B + l_2 * np.sin(np.deg2rad(theta_degB))

# plot line AB
# You have the cordinates for two points
# pointA(x_A, x_B)
# pointB(y_B, y_B)
# therefore, to define this line we must use point slope form
# ----------------------------
# ClassWork: plot first linkage of the robot
# use the point slope form that you learned in math class to get your linear
# when you finish, mail to jfigueroa@nasriacademy.org
# the program below demonstrates an attempt at this. Is the solution correct?
# ----------------------------
# point slope form for linkage 1
m = (y_B - y_A) / (x_B - x_A)              # find the slope m, rise over run
t = np.arange(x_A, x_B, 0.05)                 # declare indy value
yB = m * t - m * x_A + y_A

# point slope form for linkage 2
m2 = (y_C - y_B) / (x_C - x_B)
t2 = np.arange(x_B, x_C, 0.05)
yC = m2 * t2 - m2 * x_B + y_B
# ----------------------
fig, ax = plt.subplots()
ax.plot(t, yB)
ax.plot(t2, yC)
ax.set(xlabel='X values', ylabel='Y values',
       title='First and Second linkages of ArmRobot')
ax.grid()
fig.savefig("RobotArmPart2.png")
plt.show()

# print helpful flags below
# this should be right. notice how the x-axis grid change
# can you freeze the grid so that you get a similar plot
# with different parameters?

```

Autopep8: √   LF | UTF-8 Python