Shivani Patel 3033800943 Homework 3 EECS16a

1. Matrix Operations Practice

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 8 & 6 & 4 \\ 1 & 3 & 5 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 3 & 10 \\ 7 & 2 \end{bmatrix}$$

ai) AB

$A = \begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix}$ $B = \begin{bmatrix} 8 & 6 & 4 \\ 1 & 3 & 5 \end{bmatrix}$

$\begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 8 & 6 & 4 \\ 1 & 3 & 5 \end{bmatrix}$

$3 \times 2 \quad 2 \times 3$

$\begin{bmatrix} 3\cdot8+2\cdot1 & 3\cdot6+2\cdot3 & 3\cdot4+2\cdot5 \\ 5\cdot8+6\cdot1 & 5\cdot6+6\cdot3 & 5\cdot4+6\cdot5 \\ 1\cdot8+2\cdot1 & 1\cdot6+2\cdot3 & 1\cdot4+2\cdot5 \end{bmatrix}$

$\begin{bmatrix} 48 & 24 & 22 \\ 46 & 48 & 50 \\ 10 & 12 & 14 \end{bmatrix}$

aii) BA

$B = \begin{bmatrix} 8 & 6 & 4 \\ 1 & 3 & 5 \end{bmatrix}$ $A = \begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix}$

$\begin{bmatrix} 8 & 6 & 4 \\ 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix}$

$2 \times 3 \quad 3 \times 2$

$\begin{bmatrix} 8\cdot3+6\cdot5+4\cdot1 & 8\cdot2+6\cdot6+4\cdot2 \\ 1\cdot3+3\cdot5+5\cdot1 & 1\cdot2+3\cdot6+5\cdot2 \end{bmatrix}$

$\begin{bmatrix} 24+30+4 & 16+36+8 \\ 3+15+5 & 2+18+10 \end{bmatrix}$

$\begin{bmatrix} 58 & 58 \\ 23 & 30 \end{bmatrix}$

aiii) AD

$A = \begin{bmatrix} 3 & 2 \\ 5 & 6 \\ 1 & 2 \end{bmatrix}$ $D = \begin{bmatrix} 3 & 10 \\ 7 & 2 \end{bmatrix}$

$\begin{bmatrix} 3\cdot3+2\cdot7 & 3\cdot10+2\cdot2 \\ 5\cdot3+6\cdot7 & 5\cdot10+6\cdot2 \\ 1\cdot3+2\cdot7 & 1\cdot10+2\cdot2 \end{bmatrix}$

$\begin{bmatrix} 9+14 & 30+4 \\ 15+42 & 50+12 \\ 3+14 & 10+4 \end{bmatrix}$

$\begin{bmatrix} 23 & 34 \\ 57 & 62 \\ 17 & 14 \end{bmatrix}$

bi) A^T

$\begin{bmatrix} 3 & 5 & 1 \\ 2 & 6 & 2 \end{bmatrix}$

bii) B^T

$\begin{bmatrix} 8 & 1 \\ 6 & 3 \\ 4 & 5 \end{bmatrix}$

biii) C^T

$\begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix}$

1c) $(AB)C = A(BC)$

1d) $(AB)^T = B^T A^T$

$(A\vec{b}_i)^T = \left([\vec{a}_1\ \vec{a}_2 \dots \vec{a}_n]\,\vec{b}_i\right)^T$ set up equation

$= (b_{i1}\vec{a}_1 + b_{i2}\vec{a}_2 + \dots + b_{in}\vec{a}_n)^T$ distribute $\vec{b}_i$

$= (b_{i1}\vec{a}_1^T + b_{i2}\vec{a}_2^T + \dots + b_{in}\vec{a}_n^T)$ distribute Transpose

$= [b_{i1}\ b_{i2} \dots b_{in}] \begin{bmatrix} \vec{a}_1^T \\ \vec{a}_2^T \\ \vdots \\ \vec{a}_n^T \end{bmatrix}$ create matrix

$= \vec{b}_i^T A^T$

$(AB)^T = (A[\vec{b}_1\ \vec{b}_2 \dots \vec{b}_p])^T$ set up equation

$= [A\vec{b}_1\ A\vec{b}_2 \dots A\vec{b}_p)^T$ distribute A to b_i

$= \begin{bmatrix} (A\vec{b}_1)^T \\ (A\vec{b}_2)^T \\ \vdots \\ (A\vec{b}_p)^T \end{bmatrix}$ distribute transpose but transpose Abp into matrix

$= \begin{bmatrix} \vec{b}_1^T A^T \\ \vec{b}_2^T A^T \\ \vdots \\ \vec{b}_p^T A^T \end{bmatrix}$ distribute transpose within the matrix.

$= B^T A^T$ shows that $(BA)^T = B^T A^T$ is equivalent

2. Mechanical Inverses

2a)

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\left[\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right]$$

$R_2 \leftrightarrow R_1$

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right]$$

$$\boxed{A^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}$$

2b

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\left[\begin{array}{cc|cc} -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

$R_1 \leftarrow -R_1$

$$\left[\begin{array}{cc|cc} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

$$\boxed{A^{-1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}}$$

1c)

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

there is no inverse because we cannot get 1 into the leading entry.

e) $A = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{array}\right]$$

$R_2 \leftarrow -2 + R_2$

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & -2 & -2 & 1 \end{array}\right]$$

$R_2 \leftarrow \frac{R_2}{-2}$

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & -1/2 \end{array}\right]$$

$R_1 \leftarrow -R_1 + R_2$

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 1/2 \\ 0 & 1 & 1 & -1/2 \end{array}\right]$$

$$\boxed{A^{-1} = \begin{bmatrix} 0 & 1/2 \\ 1 & -1/2 \end{bmatrix}}$$

f) $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 1 & 4 & 4 \end{bmatrix}$

Determinate is zero, therefore the inverse matrix does not exist

3. Quadcopter Transformations

a) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\sqrt{3}}{2} & 1/2 \\ 0 & 1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 1/2 & \frac{-\sqrt{3}}{2} & 0 \\ \pi/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1/2 & -\sqrt{3}/2 & 0 \\ 3/4 & \sqrt{3}/4 & 1/2 \\ \sqrt{3}/4 & 1/4 & \sqrt{3}/2 \end{bmatrix}$

b) $\begin{bmatrix} 1/2 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3}/2 & -1/2 \\ 0 & 1/2 & \sqrt{3}/2 \end{bmatrix} = \begin{bmatrix} 1/2 & -3/4 & \sqrt{3}/4 \\ \sqrt{3}/2 & \sqrt{3}/4 & -1/4 \\ 0 & 1/2 & \sqrt{3}/2 \end{bmatrix}$

c) what you want $\begin{bmatrix} 1/2 & \sqrt{3}/2 & 0 \\ 3/4 & \sqrt{3}/4 & -1/2 \\ \sqrt{3}/4 & 1/4 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1 - \sqrt{3}}{2} \\ 1 + \sqrt{3} \\ \frac{3\sqrt{3} + 1}{4} \end{bmatrix}$

actually $\begin{bmatrix} 1/2 & -3/4 & \sqrt{3}/4 \\ \sqrt{3}/2 & \sqrt{3}/4 & -1/4 \\ 0 & 1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3} - 1}{4} \\ \frac{3\sqrt{3} - 1}{4} \\ \frac{1 + \sqrt{3}}{2} \end{bmatrix}$

they aren't the same order matters!!!

d) linear combo?

4. Properties of pump systems

a) $\vec{x}_A[n+1] = 1x_a(n) + 1x_b(n)$

$\vec{x}_b[n+1] = 0xa(n) - 0xb(n)$

b) $A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$

c) $x_a[0] = 0.5 \quad x_b[0] = 0.5$

$x_a[0] = 0.3 \quad x_b[0] = 0.7$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5+0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} = \begin{bmatrix} 0.3+0.7 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

d) It is not possible because the transition matrix A has linearly dependent columns s.t. it is not invertible. It would be very hard to figure out $x_i[0]$ is when $x_i[1]$ will equal the same matrix:

e) you cannot know what the initial state was. Matrix A does not have a unique solution therefore there must be a linearly dependent column

f) $x_1(n+1) = 0$

$x_2(n+1) = 0.4x_1(n) + 0.5x_2(n) + 0.2x_3(n)$

$x_3(n+1) = 0.6x_2(n) + 0.35x_3(n)$

total = $0.4x_1 + 1.1x_2 + 0.55x_3$ the total amt of water is non-conservative, some water is being lost

5. Segway Tours

a) $\vec{x}[1] = A\vec{x}[0] + \vec{b}u[0]$

b) $\vec{x}[2] = A\vec{x}[1] + \vec{b}u[1]$

$= A(A\vec{x}[0] + \vec{b}u[0]) + \vec{b}u[1]$

$= A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1]$

$\vec{x}[3] = A\vec{x}[2] + \vec{b}u[2]$

$= A(A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1]) + \vec{b}u[1]$

$= A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]$

$\vec{x}[4] = A\vec{x}[3] + \vec{b}u[3]$

$= A(A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]) + \vec{b}u[1]$

$= A^4\vec{x}[0] + A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}u[3]$

c) solving for N steps

$$\vec{x}[N] = A^N\vec{x}[0] + \sum_{i=0}^{N-1} A^i\vec{b}u[N-i-1]$$

d) No we cannot solve $\vec{x}_f$ in two step time because in order to do so, the following system of LEQ's needs to be solved.

$$A\vec{b}u[0] + \vec{b}u[1] = \vec{x}_f - A^2\vec{x}[0]$$

where $u[0]$ and $u[1]$ are not known

Ideally we want $\vec{x}_f$ should equal a 4×1 matrix of 0's.

From this info we can remove it from the system of LEQs:

$$A\vec{b}u[0] + \vec{b}u[1] = -A^2\vec{x}[0]$$

$$\begin{bmatrix} \vdots & \vdots \\ A\vec{b} & \vec{b} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \end{bmatrix} = -A^2\vec{x}[0]$$

turn into matrix by seperating u and vectors

$$\left[\begin{array}{cc|c} A\vec{b} & \vec{b} & -A^2\vec{x}[0] \end{array}\right]$$

turn into Augmented matrix

$$\left[\begin{array}{cc|c} 0.0208 & 0.01 & 0.022434752 95 \\ 0.0557 & 0.21 & -0.3078511661 1 \\ -0.0572 & -0.03 & 0.06193476 08 \\ -0.2385 & -0.44 & 1.3867132 5508 \end{array}\right]$$

substitute values A, $\vec{b}$ and $x[0]$

through Gaussian Elimination we get:

$$\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array}\right]$$

we see from this rref that the third row is inconsistent with an identity matrix therefore there is no solution for $u[0]$ and $u[1]$

e) NO we cannot reach $\vec{x}_f$ in three steps because of the similar issue in part d)

$$\vec{x}[3] = A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]$$
$$= A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2] = \vec{x}[3] - A^3\vec{x}[0]$$

this means that

$$A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2] = \vec{x}_f - A^3\vec{x}[0]$$

in this case $u[0]$, $u[1]$, $u[2]$ are the unknowns we also want $\vec{x}_f$ should equal a 4×1 matrix of 0's.

$$\left[\begin{array}{ccc} | & | & | \\ A^2\vec{b} & A\vec{b} & \vec{b} \\ | & | & | \end{array}\right] \left[\begin{array}{c} u[0] \\ u[1] \\ u[2] \end{array}\right] = -A^3\vec{x}[0]$$

Now we turn it into an augmented matrix

$$\left[\begin{array}{ccc|c} | & | & | & | \\ A^2\vec{b} & A\vec{b} & \vec{b} & -A^3\vec{x}[0] \\ | & | & | & | \end{array}\right]$$

Now plug in A, $\vec{b}$, $\vec{x}[0]$

$$\left[\begin{array}{ccc|c} 0.024157 & 0.0208 & 0.01 & 0.006422847\,03365 \\ 0.024363 & 0.0557 & 0.21 & -0.092123298\,431 \\ -0.083488 & -0.0572 & -0.03 & 0.178491836\,209001 \\ -0.342448 & -0.2385 & -0.44 & 1.246334\,2433\,28597 \end{array}\right]$$

↓ turn this into Gaussian elimination

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

← this is inconsistent because of the 4th row

f) yes!

We start with the same equation but for $x[4]$:

$$A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}u[3] = \vec{x} - A^4\vec{x}[0]$$

again, $u[0]$, $u[1]$, $u[2]$, $u[3]$ are unknown

And we want $\vec{x}_f$ to be a 4×1 matrix of 0's. We reduce our equation to:

$$A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}[2] + \vec{b}u[3] = -A^4\vec{x}[0]$$

$$\begin{bmatrix} | & | & | & | \\ A^3\vec{b} & A^2\vec{b} & A\vec{b} & \vec{b} \\ | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} u[0] \\ u[1] \\ u[2] \\ u[3] \end{bmatrix} = -A^4\vec{x}[0]$$

Q $\qquad$ $\vec{u}_4$

$$Q\vec{u}_4 = -A^4\vec{x}[0]$$

$$\vec{u}_4 = \begin{bmatrix} -13.28750075 \\ 23.73325125 \\ -11.5718182 \\ 1.46515973 \end{bmatrix}$$

h) This is the same as earlier bc. Any 2 vectors ($\vec{V}_{10}$, $\vec{V}_{20}$) this will result in the same vector $\vec{V}_n$
A must be linearly dependent & that $\vec{v}_n$ can result from an initial # of vectors

i) Given a transition matrix A where every initial state has a unique state vector in the next timestep ($\vec{v}_i$) we can say that each $\vec{v}_i$ also can be an initial state vector with its own unique mapping. Therefore all vectors in this space have a one-to-one mapping. This implies that A has linearly independent columns & must be invertible. Since its invertible, that means we can use A^{-1} to reverse the timestamp vector to its initial vector

6. Homework process & study Group

I worked on this homework with sadia Qureshi (3034541667)

I first worked on this by myself and then asked sadia when I got stuck

I spent roughly 6 hours on this homework

EECS16A: Homework 3

Problem 5: Segway Tours

Run the following block of code first to get all the dependencies.

In [1]:
```
# %load gauss_elim.py
from gauss_elim import gauss_elim
```

In [2]:
```
from numpy import zeros, cos, sin, arange, around, hstack
from matplotlib import pyplot as plt
from matplotlib import animation
from matplotlib.patches import Rectangle
import numpy as np
from scipy.interpolate import interp1d
import scipy as sp
```

Dynamics

In [3]:
```
# Dynamics: state to state
A = np.array([[1, 0.05, -.01, 0],
              [0, 0.22, -.17, -.01],
              [0, 0.1, 1.14, 0.10],
              [0, 1.66, 2.85, 1.14]]);
# Control to state
b = np.array([.01, .21, -.03, -0.44])
nr_states = b.shape[0]

# Initial state
state0 = np.array([-0.3853493, 6.1032227, 0.8120005, -14])

# Final (terminal state)
stateFinal = np.array([0, 0, 0, 0])
```

Part (d), (e), (f)

In [ ]:
```
# You may use gauss_elim to help you find the row reduced echelon form.
```

Part (g)

Preamble

This function will take care of animating the segway.

In [4]:
```
# frames per second in simulation
fps = 20
# length of the segway arm/stick
stick_length = 1.

def animate_segway(t, states, controls, length):
    #Animates the segway

    # Set up the figure, the axis, and the plot elements we want to anim
ate
    fig = plt.figure()

    # some config
    segway_width = 0.4
    segway_height = 0.2

    # x coordinate of the segway stick
    segwayStick_x = length * np.add(states[:, 0],sin(states[:, 2]))
    segwayStick_y = length * cos(states[:, 2])

    # set the limits
    xmin = min(around(states[:, 0].min() - segway_width / 2.0, 1), aroun
d(segwayStick_x.min(), 1))
    xmax = max(around(states[:, 0].max() + segway_height / 2.0, 1), arou
nd(segwayStick_y.max(), 1))

    # create the axes
    ax = plt.axes(xlim=(xmin-.2, xmax+.2), ylim=(-length-.1, length+.1),
aspect='equal')

    # display the current time
    time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)

    # display the current control
    control_text = ax.text(0.05, 0.8, '', transform=ax.transAxes)

    # create rectangle for the segway
    rect = Rectangle([states[0, 0] - segway_width / 2.0, -segway_height
/ 2],
         segway_width, segway_height, fill=True, color='gold', ec='blue')
    ax.add_patch(rect)

    # blank line for the stick with o for the ends
    stick_line, = ax.plot([], [], lw=2, marker='o', markersize=6, color=
'blue')

    # vector for the control (force)
    force_vec = ax.quiver([],[],[],[],angles='xy',scale_units='xy',scale
=1)

    # initialization function: plot the background of each frame
    def init():
        time_text.set_text('')
        control_text.set_text('')
        rect.set_xy((0.0, 0.0))
        stick_line.set_data([], [])
```

```
        return time_text, rect, stick_line, control_text

    # animation function: update the objects
    def animate(i):
        time_text.set_text('time = {:2.2f}'.format(t[i]))
        control_text.set_text('force = {:2.3f}'.format(controls[i]))
        rect.set_xy((states[i, 0] - segway_width / 2.0, -segway_height /
2))
        stick_line.set_data([states[i, 0], segwayStick_x[i]], [0, segway
Stick_y[i]])
        return time_text, rect, stick_line, control_text

    # call the animator function
    anim = animation.FuncAnimation(fig, animate, frames=len(t), init_fun
c=init,
             interval=1000/fps, blit=False, repeat=False)
    return anim
    # plt.show()
```

Plug in your controller here

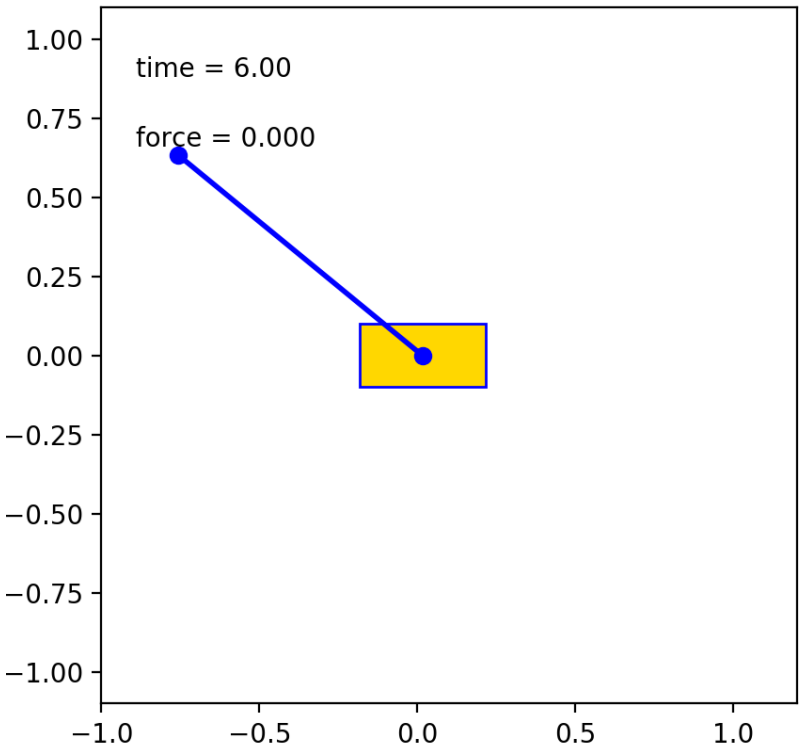
In [5]:
```
controls = np.array([0,0,0,0]) # here
```

Simulation

In [6]:
```
# This will add an extra couple of seconds to the simulation after the i
nput controls with no control
# the effect of this is just to show how the system will continue after
 the controller "stops controlling"
controls = np.append(controls,[0, 0])

# number of steps in the simulation
nr_steps = controls.shape[0]

# We now compute finer dynamics and control vectors for smoother visuali
zation
Afine = sp.linalg.fractional_matrix_power(A,(1/fps))
Asum = np.eye(nr_states)
for i in range(1, fps):
    Asum = Asum + np.linalg.matrix_power(Afine,i)

bfine = np.linalg.inv(Asum).dot(b)

# We also expand the controls in the "intermediate steps" (only for visu
alization)
controls_final = np.outer(controls, np.ones(fps)).flatten()
controls_final = np.append(controls_final, [0])

# We compute all the states starting from x0 and using the controls
states = np.empty([fps*(nr_steps)+1, nr_states])
states[0,:] = state0;
for stepId in range(1,fps*(nr_steps)+1):
    states[stepId, :] = np.dot(Afine,states[stepId-1, :]) + controls_fin
al[stepId-1] * bfine

# Now create the time vector for simulation
t = np.linspace(1/fps,nr_steps,fps*(nr_steps),endpoint=True)
t = np.append([0], t)
```

Visualization

In [7]:
```
%matplotlib nbagg
# %matplotlib qt
anim = animate_segway(t, states, controls_final, stick_length)
anim
```



Out[7]: <matplotlib.animation.FuncAnimation at 0x11c72eed0>

In [ ]: