EECS 16A HW02

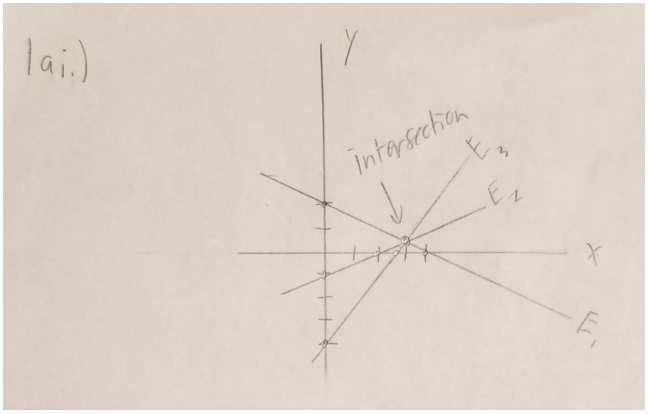
Bryan Ngo

2019-09-09

1 Gaussian Elimination

1.a

1.a.i



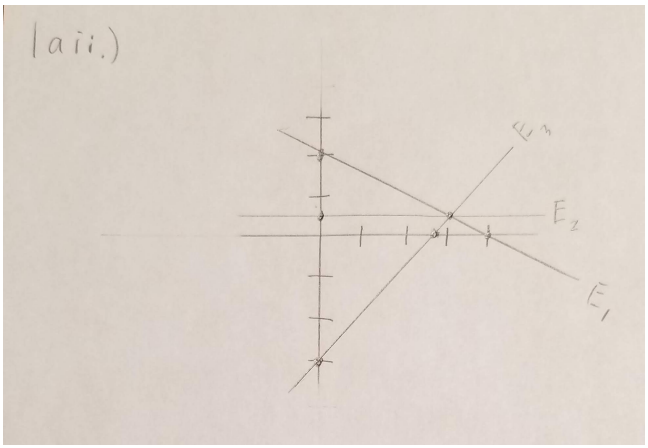
1.a.ii

The equation can be written as the augmented matrix

$$\left[\begin{array}{cc|c} 1 & 2 & 4 \\ 2 & -4 & 4 \\ 3 & -2 & 8 \end{array}\right] \tag{1}$$

$$\Rightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & -8 & -4 \\ 3 & -2 & 8 \end{array}\right] \quad r_2 - 2r_1 \to r_2 \tag{2}$$

$$\Rightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 3 & -2 & 8 \end{array}\right] \quad r_2/2 \to r_2 \tag{3}$$



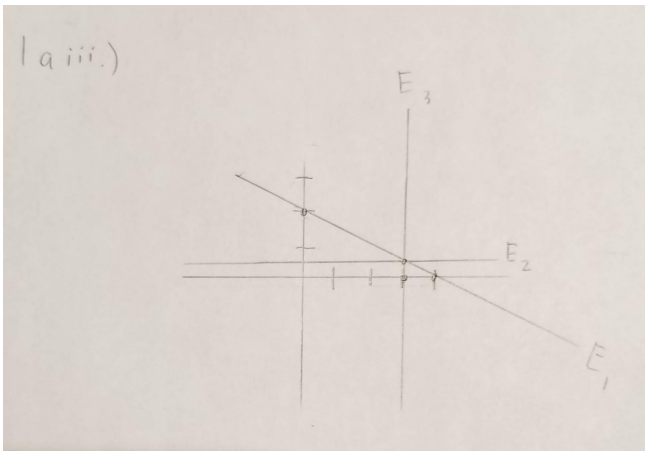
1.a.iii

$$\left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 4 & 0 & 12 \end{array}\right] \quad r_3 + r_1 \to r_3 \tag{4}$$

$$\Rightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 1 & 0 & 3 \end{array}\right] \quad r_3/4 \to r_3 \tag{5}$$

$$\Rightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 1 & 0 & 3 \\ 0 & 1 & \frac{1}{2} \end{array}\right] \quad r_2 \longleftrightarrow r_3 \tag{6}$$

When we graph the system, we notice that E_1 is redundant because it already intersects with $E_{2,3}$. Specifically, the *unique* solution to the system is $(3, \frac{1}{2})$.



1.b

The given system can be written as the augmented matrix

$$\left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 1 & 12 & 6 & 1 \\ 0 & 2 & 1 & 4 \\ 3 & 16 & 16 & 7 \end{array}\right] \tag{7}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 1 & -2 \\ 0 & 2 & 1 & 4 \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_2 - r_1 \to r_2 \tag{8}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 1 & -2 \\ 0 & 0 & \frac{4}{5} & \frac{22}{5} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_3 - r_2/5 \to r_3 \tag{9}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 1 & -2 \\ 0 & 0 & 1 & \frac{11}{2} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad \frac{5}{4} r_3 \to r_3 \tag{10}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 0 & \frac{-15}{2} \\ 0 & 0 & 1 & \frac{11}{2} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_2 - r_3 \to r_2 \tag{11}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 1 & 0 & \frac{-3}{4} \\ 0 & 0 & 1 & \frac{11}{2} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_2/10 \to r_2 \tag{12}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 5 & \frac{15}{4} \\ 0 & 1 & 0 & \frac{-3}{4} \\ 0 & 0 & 1 & \frac{11}{2} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_1 - 2r_2 \to r_1 \tag{13}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & -23 \\ 0 & 1 & 0 & \frac{-3}{4} \\ 0 & 0 & 1 & \frac{11}{2} \\ 3 & 16 & 16 & 7 \end{array}\right] \quad r_1 - 5r_3 \to r_1 \tag{14}$$

If we plug in $(-23, -\frac{3}{4}, \frac{11}{2})$, we can see that it satisfies E_4:

$$3(-23) + 16\left(-\frac{3}{4}\right) + 16\left(\frac{11}{2}\right) = -69 - 12 + 88 = 7 \tag{15}$$

which yields our *unique* solution.

1.c

Proof. Given our system

$$x+2y+5z=6 \tag{16}$$
$$3x+9y+6z=3 \tag{17}$$

we can rewrite our system as a matrix-vector product

$$\begin{bmatrix}1 & 2 & 5\\ 3 & 9 & 6\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix}=\begin{bmatrix}6\\ 3\end{bmatrix} \tag{18}$$

Turning our attention to our set S, where

$$S=\left\{\ \mathbf{v}\ \middle|\ \mathbf{v}=\begin{bmatrix}16\\ -5\\ 0\end{bmatrix}+\begin{bmatrix}-11\\ 3\\ 1\end{bmatrix}t,\ t\in\mathbb{R}\ \right\} \tag{19}$$

we can reduce the definition of S to a single matrix since both matrices in its definition are in $\mathbb{R}^3$ and $t \in \mathbb{R}$. This yields us

$$S=\left\{\ \mathbf{v}\ \middle|\ \mathbf{v}=\begin{bmatrix}16-11t\\ -5+3t\\ t\end{bmatrix},\ t\in\mathbb{R}\ \right\} \tag{20}$$

Note that the only thing differentiating elements of S is the value of t. Referring back to Equation 18, if we consider our equation in the form $\mathbf{Av} = \mathbf{b}$, where

$$\mathbf{v}=\begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{21}$$

all we have to do is substitute our definition of v into Equation 18:

$$\begin{bmatrix}1 & 2 & 5\\ 3 & 9 & 6\end{bmatrix}\begin{bmatrix}16-11t\\ -5+3t\\ t\end{bmatrix}=\begin{bmatrix}6\\ 3\end{bmatrix} \tag{22}$$

Carrying out the matrix multiplication,

$$\begin{bmatrix}(16-11t)+2(-5+3t)+5(t)\\ 3(16-11t)+9(-5+3t)+6(t)\end{bmatrix}=\begin{bmatrix}6\\ 3\end{bmatrix} \tag{23}$$
$$\begin{bmatrix}16-\cancel{11t}-10+\cancel{6t}+\cancel{5t}\\ 48-\cancel{33t}-45+\cancel{27t}+\cancel{6t}\end{bmatrix}=\begin{bmatrix}6\\ 3\end{bmatrix} \tag{24}$$
$$\begin{bmatrix}6\\ 3\end{bmatrix}=\begin{bmatrix}6\\ 3\end{bmatrix} \tag{25}$$

Thus, we have proved that no matter what value of t, and therefore any element of S since t is the only free variable, they will always cancel and you will get a solution to the system, since the resulting calculation satisfies the resultant vector. □

1.d

Given our rref'ed matrix

$$\left[\begin{array}{ccccc|c} 1 & 1 & 0 & 0 & 3 & 16 \\ 0 & 0 & 1 & 0 & -3 & -17 \\ 0 & 0 & 0 & 1 & 1 & 5 \end{array}\right] \tag{26}$$

A *free variable* is one that is not a pivot. This means that there are 2 free variables in the system, that in the 2nd column and that in the last column. The matrix can be reconverted into a system of equations into

$$x+y \quad +3v= \ 16 \tag{27}$$
$$z \ -3v=-17 \tag{28}$$
$$w+ \ v= \quad 5 \tag{29}$$

for which our free variables are y, v and our pivots are x, z, w. In order to generate the infinite set of solutions, we must solve for the pivots in terms of the free variables. Using simple algebra, we can do this:

$$x(t) = 16 - y - 3v \tag{30}$$
$$z(t) = 3v + 17 \tag{31}$$
$$w(t) = 5 - v \tag{32}$$

2 Boser's Optimal Boba

2.a

We can represent the single rating as a linear combination of the pure teas weighted by their relative composition in the teas, leading to the system of equations

$$\frac{1}{3}\beta+\frac{1}{3}\Omega+0\gamma+\frac{1}{3}\epsilon= \ 7 \tag{33}$$
$$\frac{1}{3}\beta+\frac{1}{3}\Omega+\frac{1}{3}\gamma+0\epsilon= \ 7 \tag{34}$$
$$0\beta+\frac{2}{5}\Omega+\frac{3}{5}\gamma+0\epsilon=\frac{37}{5} \tag{35}$$
$$\frac{2}{3}\beta+\frac{1}{3}\Omega+0\gamma+0\epsilon=\frac{19}{3} \tag{36}$$

for βlack, Ωolong, γreen, and ϵarl Grey tea ratings. This looks like the following matrix-vector multiplication and augmented matrix:

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \frac{2}{5} & \frac{3}{5} & 0 \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ \Omega \\ \gamma \\ \epsilon \end{bmatrix} = \begin{bmatrix} 7 \\ 7 \\ \frac{37}{5} \\ \frac{19}{3} \end{bmatrix} \tag{37}$$

$$\left[\begin{array}{cccc|c} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 7 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 7 \\ 0 & \frac{2}{5} & \frac{3}{5} & 0 & \frac{37}{5} \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 & \frac{19}{3} \end{array}\right] \tag{38}$$

Given the complexity of this system, we employ a faster means of solving the system, yielding Prof. Ranade's ratings for the pure teas:

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & 9 \end{array}\right] \tag{39}$$

or $7 = \beta$lack, $5 = \Omega$olong, $9 = \gamma$reen, and $9 = \epsilon$arl Grey.

2.b

Prof. Ranade's optimal tea can be created using her ratings for the pure teas by the following linear equation

$$7a + 5b + 9c + 9d = S \tag{40}$$

where pure tea proportions $a + b + c + d = 1$ and S is the final score. It can be easily shown that it is impossible to get $S > 9$ from Prof. Ranade within the constraints of the problem because having any weight other than 0 in any non-9 term would bring the score below 9. Thus, a combination that would maximize Ranade's score could be any proportion of Oolong and Earl Grey tea, e.g. $\frac{28}{53}$ Oolong tea and $\frac{25}{53}$ Earl Grey.

3 Finding Charges from Potential Measurements

Assuming we ignore k for the purposes of calculation (since it cancels out), we can simply substitute the potentials $U_{1,2,3}$ into the formulas for the linear combination of the charges,

$$\frac{Q_1}{\sqrt{2}} + \frac{Q_2}{\sqrt{5}} + \frac{Q_3}{2} = \frac{4 + 3\sqrt{5} + \sqrt{10}}{2\sqrt{5}} \tag{41}$$

$$Q_1 + \frac{Q_2}{\sqrt{2}} + Q_3 = \frac{2 + 4\sqrt{2}}{\sqrt{2}} \tag{42}$$

$$\frac{Q_1}{2} + \frac{Q_2}{\sqrt{5}} + \frac{Q_3}{\sqrt{2}} = \frac{4 + \sqrt{5} + 3\sqrt{10}}{2\sqrt{5}} \tag{43}$$

Converting into a matrix-vector multiplication,

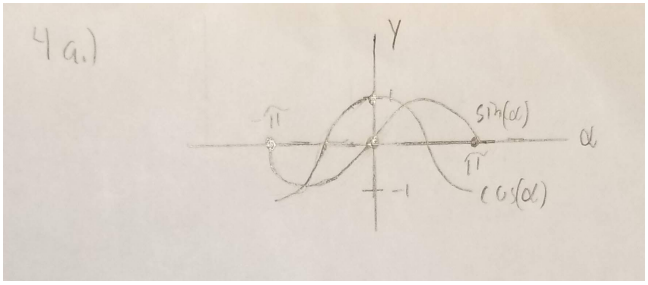
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{5}} & \frac{1}{2} \\ 1 & \frac{1}{\sqrt{2}} & 1 \\ \frac{1}{2} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \begin{bmatrix} \frac{4+3\sqrt{5}+\sqrt{10}}{2\sqrt{5}} \\ \frac{2+4\sqrt{2}}{\sqrt{2}} \\ \frac{4+\sqrt{5}+3\sqrt{10}}{2\sqrt{5}} \end{bmatrix} \tag{44}$$

See the iPython pages for the charge calculations, where we get

$$Q_1 = 1 \tag{45}$$
$$Q_2 = 2 \tag{46}$$
$$Q_3 = 3 \tag{47}$$

4 Kinematic Model for a Simple Car

4.a



We can justify the small-angle approximations

$$\sin(\alpha) \approx 0 \tag{48}$$
$$\cos(\alpha) \approx 1 \tag{49}$$

for some $\alpha \approx 0$ because, as the graphs of the sine and cosine functions clearly indicate, they get very close to those approximate values around $\alpha = 0$.

4.b

See the iPython pages for 5.b-d.

First we simplify the *nonlinear* system of equations

$$x[k+1] = x[k] + v[k]\cos(\theta[k])\Delta t \tag{50}$$
$$y[k+1] = y[k] + v[k]\sin(\theta[k])\Delta t \tag{51}$$
$$\theta[k+1] = \theta[k] + \frac{v[k]}{L}\tan(\phi[k])\Delta t \tag{52}$$
$$v[k+1] = v[k] + a[k]\Delta t \tag{53}$$

Substituting our small angle approximations,

$$x[k+1] = x[k] + v[k]\Delta t \tag{54}$$
$$y[k+1] = y[k] \tag{55}$$
$$\theta[k+1] = \theta[k] \tag{56}$$
$$v[k+1] = v[k] + a[k]\Delta t \tag{57}$$

Splitting up the system into 2 vectors,

$$\begin{bmatrix} x[k+1] \\ y[k+1] \\ \theta[k+1] \\ v[k+1] \end{bmatrix} = \begin{bmatrix} x[k] + 0.1v[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.1a[k] \end{bmatrix} \tag{58}$$

$$\begin{bmatrix} x[k+1] \\ y[k+1] \\ \theta[k+1] \\ v[k+1] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.1a[k] \end{bmatrix} \tag{59}$$

$$\begin{bmatrix} x[k+1] \\ y[k+1] \\ \theta[k+1] \\ v[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.1 & 0 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} a[k] \\ \phi[k] \end{bmatrix} \tag{60}$$

4.c

The motion between the linear and non-linear models are extremely similar in this case. This is probably due to the extremely small turning angle ϕ. This means that according to Equation 53, our $\theta[k+1]$, regardless of the step interval, is effectively 0 and thus the rest of the state vectors do not deviate very much from each other.

4.d

In this case, the trajectories of the linear and nonlinear state vectors are extremely different. This is because 0.5 rad is a relatively large angle that is non-negligible. This means our small angle approximations have massive errors and deviate heavily from the real path.

5 Fountain Codes

5.a

An example in which a is unrecoverable but b, c are is the received vector

$$\mathbf{r} = \begin{bmatrix} \star \\ b \\ c \\ \star \\ b \\ c \end{bmatrix} \tag{61}$$

Since both a entries are corrupted, we must reject them from the system, making it impossible to recover.

5.b

Our goal is to find a matrix $\mathbf{G}_R$ such that

$$\mathbf{G}_R \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ a \\ b \\ c \end{bmatrix} \tag{62}$$

The matrix in question is

$$\mathbf{G}_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{63}$$

which is nothing more than 2 $\mathbf{I}_3$ matrices concatenated vertically.

5.c

The augmented matrix is

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & \star \\ 0 & 0 & 1 & \star \\ 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 4 \\ 0 & 1 & 1 & \star \\ 1 & 1 & 1 & \star \end{array}\right] \tag{64}$$

Since the corrupted rows are unusable, we can simply ignore them, leaving us with the augmented matrix

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 4 \end{array}\right] \tag{65}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & -4 \\ 1 & 0 & 1 & 4 \end{array}\right] \quad r_2 - r_1 \to r_2 \tag{66}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & -3 \end{array}\right] \quad r_3 - r_1 \to r_3 \tag{67}$$

leaving us with the original message

$$\mathbf{m} = \begin{bmatrix} 7 \\ -4 \\ -3 \end{bmatrix} \tag{68}$$

5.d

Bob can recover the message from any 3 *or* 4 symbols he receives using $\mathbf{G}_F$, given certain constraints.

Suppose that 3 symbols are correctly received. This means that 4/7 rows are rejected from $\mathbf{G}_F$. This leaves us with a system of linear equations with 3 unknowns and 3 equations. There are 3 cases. The system can either have no solutions (geometrically, if the 3 planes do not have a common intersection), or infinite (if one of the planes is a scalar multiple of the other), there exists a set of recoverable solutions for $\mathbf{r}$.

Suppose that 4 symbols are correctly received, leaving 3/7 rows rejected. This leaves us with a system with 3 unknowns and 4 equations. While this does not always guarantee a unique solution (e.g. all the equations could be scalar multiples), it is more likely to have a unique solution than 3 given symbols, since there are more possible constraints on $\mathbf{r}$.

5.e

I prefer $\mathbf{G}_F$ because there is 1 more row than $\mathbf{G}_R$, so there is more redundancy when transmitting the message. This means $\mathbf{G}_F$ can handle more corruption.

6 Homework Process and Study Group

I did this homework by myself.

prob2

September 13, 2019

1 EECS16A: Homework 2

1.1 Problem 3: Finding Charges from Potential Measurements

[2]:
```
import numpy as np
a = np.array([[1/np.sqrt(2), 1/np.sqrt(5), 1/2],
              [1, 1/np.sqrt(2), 1],
             [1/2, 1/np.sqrt(5), 1/np.sqrt(2)]])
b = np.array([(4+3*np.sqrt(5)+np.sqrt(10))/(2*np.sqrt(5)), (2+4*np.sqrt(2))/(np.
 ↪sqrt(2)), (4+np.sqrt(5)+3*np.sqrt(10))/(2*np.sqrt(5))])
x = np.linalg.solve(a, b)
print(x)
```

```
[1. 2. 3.]
```

1.2 Problem 4: Kinematic Model for a Simple Car

This script helps to visualize the difference between a nonlinear model and a corresponding linear approximation for a simple car. What you should notice is that the linear model is similar to the nonlinear model when you are close to the point where the approximation is made.

First, run the following block to set up the helper functions needed to simulate the vehicle models and plot the trajectories taken.

[3]:
```
# DO NOT MODIFY THIS BLOCK!
''' Problem/Model Setup'''
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Vehicle Model Constants
L = 1.0  # length of the car, meters
dt = 0.1 # time difference between timestep (k+1) and timestep k, seconds

''' Nonlinear Vehicle Model Update Equation '''
def nonlinear_vehicle_model(initial_state, inputs, num_steps):
    x     = initial_state[0] # x position, meters
    y     = initial_state[1] # y position, meters
```

```
    theta = initial_state[2] # heading (wrt x-axis), radians
    v     = initial_state[3] # speed, meters per second

    a = inputs[0]            # acceleration, meters per second squared
    phi = inputs[1]          # steering angle, radians

    state_history = []       # array to hold state values as the time step k␣
↪advances.
    state_history.append([x,y,theta,v]) # add the initial state (i.e. k = 0) to␣
↪history.

    for i in range(0, num_steps):
        # Find the next state, at time k+1, by applying the nonlinear model to␣
↪the current state, at time k.
        x_next     = x     + v * np.cos(theta) * dt
        y_next     = y     + v * np.sin(theta) * dt
        theta_next = theta + v/L * np.tan(phi) * dt
        v_next     = v     + a * dt

        # Add the next state to the history.
        state_history.append([x_next,y_next,theta_next,v_next])

        # Advance to the next state, at time k+1, to get ready for next loop␣
↪iteration.
        x = x_next
        y = y_next
        theta = theta_next
        v = v_next

    return np.array(state_history)

''' Linear Vehicle Model Update Equation '''
def linear_vehicle_model(A, B, initial_state, inputs, num_steps):
    # Note: A should be a 4x4 matrix, B should be a 4x2 matrix for this linear␣
↪model.

    x     = initial_state[0] # x position, meters
    y     = initial_state[1] # y position, meters
    theta = initial_state[2] # heading (wrt x-axis), radians
    v     = initial_state[3] # speed, meters per second

    a = inputs[0]            # acceleration, meters per second squared
    phi = inputs[1]          # steering angle, radians

    state_history = []       # array to hold state values as the time step k␣
↪advances.
```

```
    state_history.append([x,y,theta,v]) # add the initial state (i.e. k = 0) to␣
 ↪history.

    for i in range(0, num_steps):
        # Find the next state, at time k+1, by applying the nonlinear model to␣
 ↪the current state, at time k.
        state_next = np.dot(A, state_history[-1]) + np.dot(B, inputs)

        # Add the next state to the history.
        state_history.append(state_next)

        # Advance to the next state, at time k+1, to get ready for next loop␣
 ↪iteration.
        state = state_next

    return np.array(state_history)

''' Plotting Setup'''
def make_model_comparison_plot(state_predictions_nonlinear,␣
 ↪state_predictions_linear):
    f = plt.figure()
    plt.plot(state_predictions_nonlinear[0,0],␣
 ↪state_predictions_nonlinear[0,1], 'go', label='Start')
    plt.plot(state_predictions_nonlinear[:,0], state_predictions_nonlinear[:
 ↪,1], 'r', label='Nonlinear')
    plt.plot(state_predictions_linear[:,0], state_predictions_linear[:,1], 'k.
 ↪', label='Linear')
    plt.legend(loc='upper left')
    plt.xlim([4, 8])
    plt.ylim([9, 12])
    plt.show()
```

1.2.1 Part B

Task: Fill in the matrices A and B for the linear system approximating the nonlinear vehicle model under small heading and steering angle approximations.

[4]:
```
# Your code here.
A = np.array([[1, 0, 0, 0.1],
              [0, 1, 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1]])

B = np.array([[ 0, 0],
              [ 0, 0],
              [ 0, 0],
              [ 0.1, 0]])
```

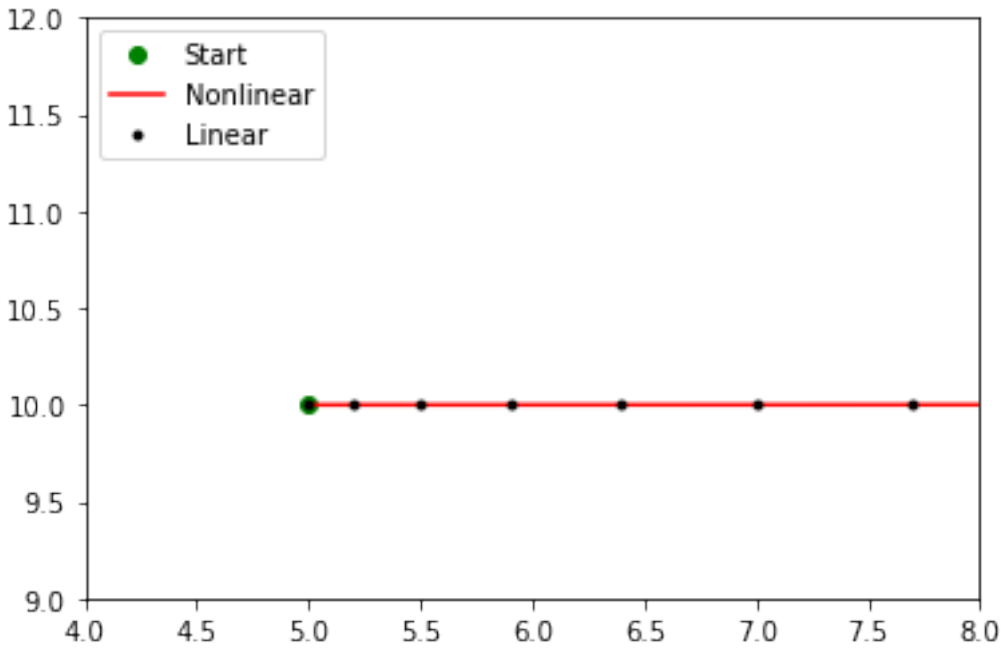
1.2.2 Part C

Task: Fill out the state and input values from Part C and look at the resulting plot. The plot should help you to visualize the difference between using a linear model and a nonlinear model for this specific starting state and input.

[5]:
```
# Your code here.
x_init   = 5.0
y_init   = 10.0
theta_init = 0.0
v_init     = 2.0
a_input    = 10
phi_input  = 0.0001

state_init = [x_init, y_init, theta_init, v_init]
state_predictions_nonlinear = nonlinear_vehicle_model(state_init, [a_input,␣
 ↪phi_input], 10)
state_predictions_linear = linear_vehicle_model(A, B, state_init, [a_input,␣
 ↪phi_input], 10)

make_model_comparison_plot(state_predictions_nonlinear,␣
 ↪state_predictions_linear)
```



1.2.3 Part D

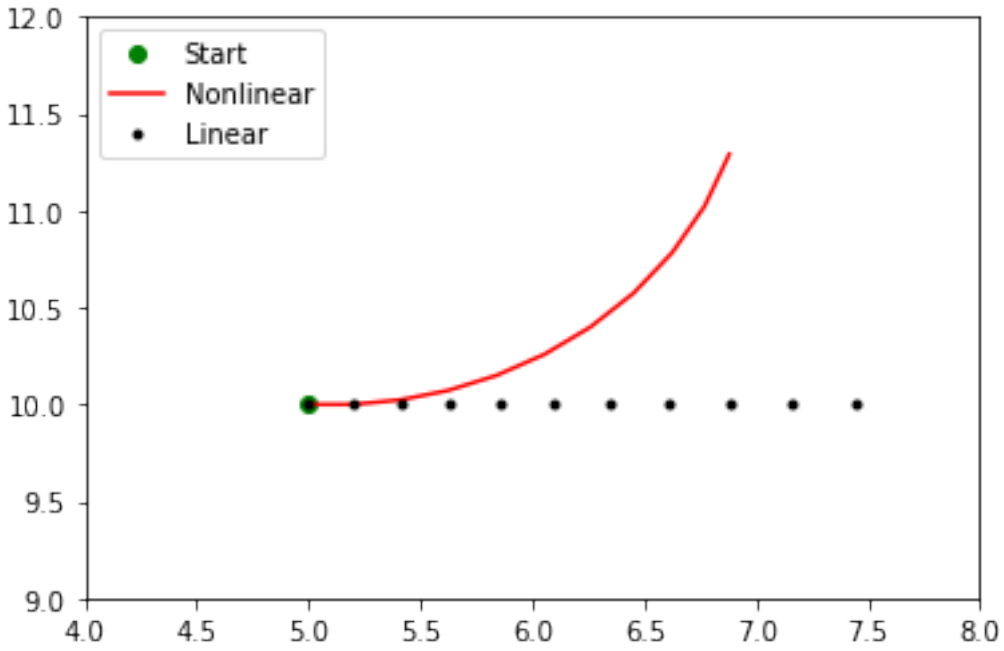
Task: Fill out the state and input values from Problem D and look at the resulting plot. The plot should help you to visualize the difference between using a linear model and a nonlinear model

for this specific starting state and input.

[6]:
```
# Your code here.
x_init  = 5.0
y_init  = 10.0
theta_init = 0.0
v_init     = 2.0
a_input    = 1.0
phi_input  = 0.5

state_init = [x_init, y_init, theta_init, v_init]
state_predictions_nonlinear = nonlinear_vehicle_model(state_init, [a_input,␣
 ↪phi_input], 10)
state_predictions_linear = linear_vehicle_model(A, B, state_init, [a_input,␣
 ↪phi_input], 10)

make_model_comparison_plot(state_predictions_nonlinear,␣
 ↪state_predictions_linear)
print(state_predictions_nonlinear[10])
print(state_predictions_linear[10])
```




```
[ 6.87984693 11.28998941  1.3384411   3.        ]
[ 7.45 10.    0.    3.  ]
```