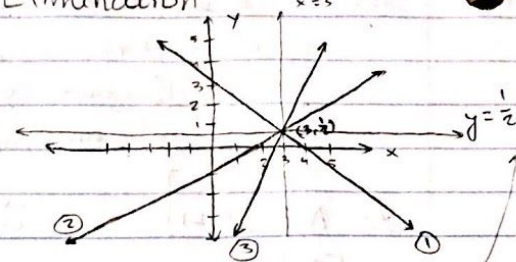
1. Gaussian Elimination

a) i) $\begin{cases} x + 2y = 4 & ① \\ 2x - 4y = 4 & ② \\ 3x - 2y = 8 & ③ \end{cases}$

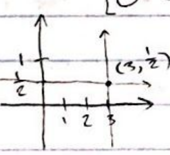


ii) $\left[\begin{array}{cc|c} 1 & 2 & 4 \\ 2 & -4 & 4 \\ 3 & -2 & 8 \end{array}\right] \xrightarrow{-2R_1+R_2} \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & -8 & -4 \\ 3 & -2 & 8 \end{array}\right] \Rightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 3 & -2 & 8 \end{array}\right]$

I notice the 2nd equation now is a horizontal line, but still intersects the intersection point.

iii) $\left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 3 & -2 & 8 \end{array}\right] \xrightarrow{-3R_1+R_3} \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1 & \frac{1}{2} \\ 0 & -8 & -4 \end{array}\right] \xrightarrow{-2R_2+R_1} \left[\begin{array}{cc|c} 1 & 0 & 3 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 \end{array}\right]$

$\begin{cases} x = 3 \\ y = \frac{1}{2} \end{cases}$



I notice the equations intersect at the same point.

b) $\left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 1 & 12 & 6 & 1 \\ 0 & 2 & 1 & 4 \\ 3 & 16 & 16 & 7 \end{array}\right] \xrightarrow[-3R_1+R_4]{-R_1+R_2} \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 1 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 10 & 1 & -2 \end{array}\right] \xrightarrow{R_2-R_4} \left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 10 & 1 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 0 & 0 & -2 \end{array}\right] \xrightarrow{-5R_3+R_2}$

$\left[\begin{array}{ccc|c} 1 & 2 & 5 & 3 \\ 0 & 0 & -4 & -22 \\ 0 & 2 & 1 & 4 \end{array}\right] \xrightarrow{-R_3+R_1} \left[\begin{array}{ccc|c} 1 & 0 & 4 & -1 \\ 0 & 0 & -4 & -22 \\ 0 & 2 & 1 & 4 \end{array}\right] \xrightarrow{R_2+R_1} \left[\begin{array}{ccc|c} 1 & 0 & 0 & -23 \\ 0 & 0 & -4 & -22 \\ 0 & 2 & 1 & 4 \end{array}\right] \Rightarrow$

$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -23 \\ 0 & 2 & 1 & 4 \\ 0 & 0 & 1 & \frac{22}{4} \end{array}\right] \xrightarrow{-R_3+R_2} \left[\begin{array}{ccc|c} 1 & 0 & 0 & -23 \\ 0 & 2 & 0 & -\frac{6}{4} \\ 0 & 0 & 1 & \frac{22}{4} \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & -23 \\ 0 & 1 & 0 & -\frac{3}{4} \\ 0 & 0 & 1 & \frac{22}{4} \end{array}\right]$

Answer: $x = -23$, $y = -\frac{3}{4}$, $z = \frac{22}{4}$ Unique Solution

$$\begin{cases} x + 2y + 5z = 6 \\ 3x + 9y + 6z = 3 \end{cases} \quad \text{and} \quad S = \left\{ \vec{v} \;\middle|\; \vec{v} = \begin{bmatrix} 16 \\ -5 \\ 0 \end{bmatrix} + \begin{bmatrix} -11 \\ 3 \\ 1 \end{bmatrix} t,\ t \in R \right\}$$

c)

$$\left[\begin{array}{ccc|c} 1 & 2 & 5 & 6 \\ 3 & 9 & 6 & 3 \end{array}\right] \xRightarrow{-3R_1+R_2} \left[\begin{array}{ccc|c} 1 & 2 & 5 & 6 \\ 0 & 3 & -9 & -15 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 5 & 6 \\ 0 & 1 & -3 & -5 \end{array}\right] \xRightarrow{-2R_2+R_1}$$

$$\left[\begin{array}{ccc|c} 1 & 0 & 11 & 16 \\ 0 & 1 & -3 & -5 \end{array}\right] \Rightarrow \begin{cases} x + 11z = 16 \\ y - 3z = -5 \end{cases} \Rightarrow \begin{cases} x = 16 - 11z \\ y = -5 + 3z \end{cases}$$

Let $z = t$:

then, $\begin{cases} x = 16 - 11t \\ y = -5 + 3t \\ z = t \end{cases} \Rightarrow \boxed{\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 16 \\ -5 \\ 0 \end{bmatrix} + \begin{bmatrix} -11 \\ 3 \\ 1 \end{bmatrix} t}$

Since $t \in R$, the set of solution is

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

Thus, we proved that any vector in S is a solution.

d)

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

There are 2 free variables.

$$\begin{cases} x + y + v = 16 \\ z - 3v = -17 \\ w + v = 5 \end{cases} \Rightarrow \begin{cases} x = 16 - y - v \\ z = -17 + 3v \\ w = 5 - v \end{cases}$$

Parametric solution:

Let $y = u$.

$$\boxed{(16 - u - v,\ u,\ -17 + 3v,\ 5 - v,\ v)}$$

2. Boser's Optimal Boba

a) Let Black = x , Oolong = y, Green = z, Earl Grey = w

$$\begin{cases} \frac{1}{3}x + \frac{1}{3}y + \frac{1}{3}w = 7 \\ \frac{1}{3}x + \frac{1}{3}y + \frac{1}{3}z = 7 \\ \frac{2}{5}y + \frac{3}{5}z = 7\frac{2}{5} \\ \frac{2}{3}x + \frac{1}{3}y = 6\frac{1}{3} \end{cases}$$

$$\left[\begin{array}{cccc|c} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 7 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 7 \\ 0 & \frac{2}{5} & \frac{3}{5} & 0 & 7\frac{2}{5} \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 & 6\frac{1}{3} \end{array}\right] \xrightarrow[-2R_1+R_4]{R_1-R_2}$$

$$\left[\begin{array}{cccc|c} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 7 \\ 0 & 0 & \frac{1}{3} & -\frac{1}{3} & 0 \\ 0 & \frac{2}{5} & \frac{3}{5} & 0 & 7\frac{2}{5} \\ 0 & -\frac{1}{3} & 0 & -\frac{2}{3} & -\frac{23}{3} \end{array}\right] \xrightarrow[\frac{1}{3}R_3]{R_4+R_1} \left[\begin{array}{cccc|c} \frac{1}{3} & 0 & 0 & -\frac{1}{3} & -\frac{2}{3} \\ 0 & 0 & \frac{1}{3} & -\frac{1}{3} & 0 \\ 0 & \frac{6}{15} & \frac{9}{15} & 0 & \frac{111}{15} \\ 0 & -\frac{1}{3} & 0 & -\frac{2}{3} & -\frac{23}{3} \end{array}\right] \xrightarrow{\frac{6}{5}R_4+R_3}$$

$$\left[\begin{array}{cccc|c} \frac{1}{3} & 0 & 0 & -\frac{1}{3} & -\frac{2}{3} \\ 0 & 0 & \frac{1}{3} & -\frac{1}{3} & 0 \\ 0 & 0 & \frac{9}{15} & -\frac{12}{15} & -\frac{27}{15} \\ 0 & -\frac{1}{3} & 0 & -\frac{2}{3} & -\frac{23}{3} \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} 1 & 0 & 0 & -1 & -2 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 9 & -12 & -27 \\ 0 & -1 & 0 & -2 & -23 \end{array}\right] \xrightarrow[-R_4]{-9R_2+R_3}$$

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & -1 & -2 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -3 & -27 \\ 0 & 1 & 0 & 2 & 23 \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} 1 & 0 & 0 & -1 & -2 \\ 0 & 1 & 0 & 2 & 23 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 9 \end{array}\right] \xrightarrow[\substack{-2R_4-R_2 \\ R_4+R_3}]{R_4+R_1}$$

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 7 \\ 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & 9 \end{array}\right] \qquad \begin{cases} x = 7 \\ y = 5 \\ z = 9 \\ w = 9 \end{cases}$$

Ratings:

Black = 7
Oolong = 5
Green = 9
Earl Grey = 9

b) There are more than one tea combination that will maximize the customer's score. Since the question doesn't ask to find all the possible tea combinations, I will answer with only one possible tea combination.

Since Professor Ranade gave a rating of 9 to Green and Earl Grey teas, any one cup containing ratios of these 2 flavors will receive a maximum score of 9.

One possible combination: Green = $\frac{1}{3}$

Earl Grey = $\frac{2}{3}$

Professor Ranade's score: 9

3. Finding Charges from Potential Measurements

$U = k\frac{Q}{r}$

$$\begin{cases} k\frac{Q_1}{r_1} + k\frac{Q_2}{r_2} + k\frac{Q_3}{r_3} = k\frac{4+3\sqrt{5}+\sqrt{10}}{2\sqrt{5}} \\ k\frac{Q_1}{r_1} + k\frac{Q_2}{r_2} + k\frac{Q_3}{r_3} = k\frac{2+4\sqrt{2}}{\sqrt{2}} \\ k\frac{Q_1}{r_1} + k\frac{Q_2}{r_2} + k\frac{Q_3}{r_3} = k\frac{4+\sqrt{5}+3\sqrt{10}}{2\sqrt{5}} \end{cases}$$

$$\begin{cases} \frac{Q_1}{\sqrt{2}} + \frac{Q_2}{\sqrt{5}} + \frac{Q_3}{2} = \frac{4+3\sqrt{5}+\sqrt{10}}{2\sqrt{5}} \\ Q_1 + \frac{Q_2}{\sqrt{2}} + Q_3 = \frac{2+4\sqrt{2}}{\sqrt{2}} \\ \frac{Q_1}{2} + \frac{Q_2}{\sqrt{5}} + \frac{Q_3}{\sqrt{2}} = \frac{4+\sqrt{5}+3\sqrt{10}}{2\sqrt{5}} \end{cases}$$

$$\left[\begin{array}{ccc|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{5}} & \frac{1}{2} & \frac{4+3\sqrt{5}+\sqrt{10}}{2\sqrt{5}} \\ 1 & \frac{1}{\sqrt{2}} & 1 & \frac{2+4\sqrt{2}}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{2}} & \frac{4+\sqrt{5}+3\sqrt{10}}{2\sqrt{5}} \end{array}\right]$$

After using numpy to solve this system:

$$\boxed{\begin{cases} Q_1 = 1 \\ Q_2 = 2 \\ Q_3 = 3 \end{cases}}$$

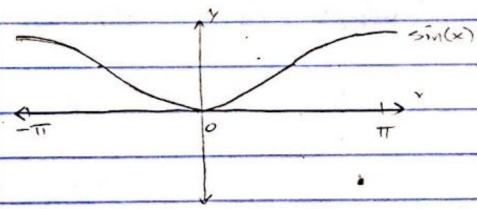
4. Kinematic Model for a Simple Car

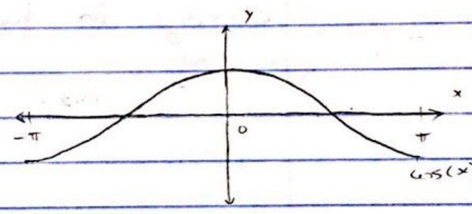
Given the approximation: $\sin(\alpha) \approx 0$, $\cos(\alpha) \approx 1$, $\tan(\alpha) \approx 0$

$$\begin{cases} x[k+1] = x[k] + v[k](1)(0.1) \\ y[k+1] = y[k] + v[k](0)(0.1) \\ \theta[k+1] = \theta[k] + \frac{v[k]}{L}(0)(0.1) \\ v[k+1] = v[k] + a[k](0.1) \end{cases}$$

$$\begin{cases} x[k+1] = x[k] + (0.1)v[k] \\ y[k+1] = y[k] \\ \theta[k+1] = \theta[k] \\ v[k+1] = v[k] + (0.1)a[k] \end{cases}$$

a)





Using these graph, I can see that we can approximate for $\alpha \approx 0$, $\sin(\alpha) \approx 0$ and $\cos(\alpha) \approx 1$. So, I can justify why we assume this way on this problem.

b)

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

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

c) The trajectories are very similar. The reason is because our value for $\varphi[k]$ is 0.0001 (too small). As a result, $\theta[k]$ changes very slowly because of $\tan(\varphi[k]) = \tan(0.0001) = 0.0001$. So, in the first 10 steps, $\theta[10] \approx 0$ and we see similar trajectories.

d) In the case of $\varphi[k] = 0.5$, the trajectories are very different. Since $\varphi[k] = 0.5$, $\theta[k]$ changes faster compared to part (c). By the 10^{th} time step, we can already see a significant change in heading, which is caused by $\varphi[k] = 0.5$

6. Homework Process and Study Group

I worked on this homework alone. I worked on it by first reading all the notes and then did it on my notebook.

5. Fountain Codes

a) Using the repetition scheme, in the case of $\begin{bmatrix} \star \\ b \\ c \\ \star \\ b \\ c \end{bmatrix}$ a cannot be recovered while b and c can.

b) $G_R \vec{m} = \vec{w}$

$$G_R \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ a \\ b \\ c \end{bmatrix}$$

Since we know $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, we can tell that

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{G_R} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{\vec{m}} = \underbrace{\begin{bmatrix} a \\ b \\ c \\ a \\ b \\ c \end{bmatrix}}_{\vec{w}}$$

c) $G_F \vec{m} = \vec{r}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 7 \\ \star \\ \star \\ 3 \\ 4 \\ \star \\ \star \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ 4 \end{bmatrix}$$

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 4 \end{array}\right] \underset{\substack{R_1 - R_2 \\ R_1 - R_3}}{\sim} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & -3 \end{array}\right]$$

$$\vec{m} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \boxed{\begin{bmatrix} 7 \\ -4 \\ -3 \end{bmatrix}}$$

d) Bob <u>cannot</u> recover $\vec{m}$ from <u>any</u> 3 symbols.
Rows of G_F corresponding to the uncorrupted symbols must be <u>linearly independent</u>.
<u>In</u> other words, given 3 uncorrupted symbols, if the corresponding rows in G_F are linearly dependent, then Bob can't recover $\vec{m}$.

For example, if $\vec{r} = \begin{bmatrix} 7 \\ 2 \\ \star \\ 1 \\ \star \\ \star \\ \star \end{bmatrix}$, then Bob can't recover $\vec{m}$.

<u>If</u>, $\vec{r} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \star \\ \star \\ \star \\ \star \end{bmatrix}$, then Bob can easily recover $\vec{m}$.

<u>In</u> the case of any 4 symbols given, Bob can <u>always recover</u> $\vec{m}$. The reason has to do with how G_F is composed.
Choosing any 4 or more row vectors from G_F, the span of it is always R^3.
In other words, let $\vec{r}_1, \vec{r}_2, \vec{r}_3, \vec{r}_4$ be row vectors of G_F, then $\text{span}(\{\vec{r}_1, \vec{r}_2, \vec{r}_3, \vec{r}_4\}) = R^3$

For example, $\vec{r} = \begin{bmatrix} 1 \\ 2 \\ \star \\ \star \\ 3 \\ \star \\ 4 \end{bmatrix}$, Bob can recover $\vec{m}$.

e) I would prefer <u>G_F</u>. This is because G_F can recover $\vec{m}$ given any 4 or more symbols. In case G_R, it cannot always recover $\vec{m}$ if at least 2 symbols are corrupted.

So, G_F is more reliable than G_R.
That is why, I prefer G_F.

EECS16A: Homework 2

Problem 3: Finding Charges from Potential Measurements

In [4]:

```
# Your code here.
import numpy as np
from math import sqrt
a = np.array([
    [1/sqrt(2), 1/sqrt(5), 1/2],
    [1, 1/sqrt(2), 1],
    [1/2, 1/sqrt(5), 1/sqrt(2)]
])
b = np.array([(4 + 3*sqrt(5) + sqrt(10)) / (2*sqrt(5)),
              (2 + 4*sqrt(2)) / sqrt(2),
              (4 + sqrt(5) + 3*sqrt(10)) / (2*sqrt(5))])
x = np.linalg.solve(a, b)
print(x)
```

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

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.

In [1]:

```
# 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, second

''' 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 st
    state_history.append([x,y,theta,v]) # add the initial state (i.e. k

    for i in range(0, num_steps):
        # Find the next state, at time k+1, by applying the nonlinear mo
        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
        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

    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 st
    state_history.append([x,y,theta,v]) # add the initial state (i.e. k

    for i in range(0, num_steps):
        # Find the next state, at time k+1, by applying the nonlinear mo
        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
        state = state_next

    return np.array(state_history)

''' Plotting Setup'''
def make_model_comparison_plot(state_predictions_nonlinear, state_predic
    f = plt.figure()
    plt.plot(state_predictions_nonlinear[0,0], state_predictions_nonline
    plt.plot(state_predictions_nonlinear[:,0], state_predictions_nonline
    plt.plot(state_predictions_linear[:,0], state_predictions_linear[:,1
    plt.legend(loc='upper left')
    plt.xlim([4, 8])
    plt.ylim([9, 12])
    plt.show()
```

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.

In [2]:

```
# 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]])
```

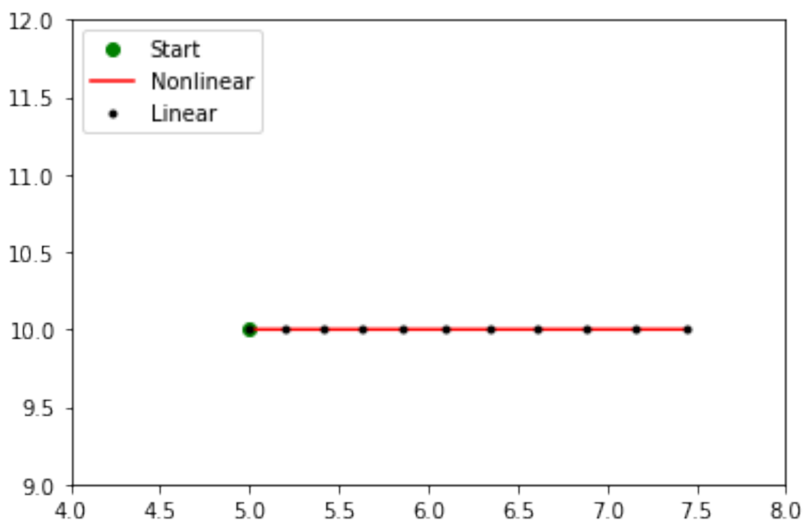
Part C

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.

In [6]:
```
# Your code here.
x_init  = 5
y_init  = 10
theta_init = 0
v_init     = 2
a_input    = 1
phi_input  = 0.0001

state_init = [x_init, y_init, theta_init, v_init]
state_predictions_nonlinear = nonlinear_vehicle_model(state_init, [a_inp
state_predictions_linear = linear_vehicle_model(A, B, state_init, [a_inp

make_model_comparison_plot(state_predictions_nonlinear, state_prediction
```



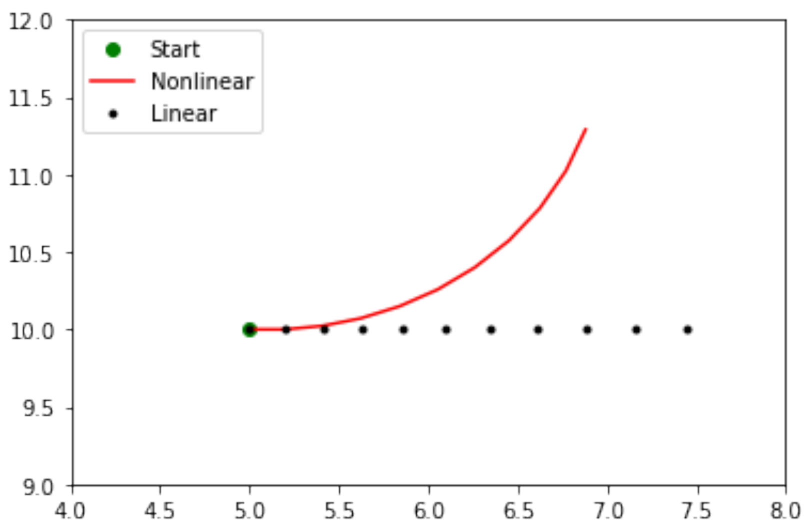
Part D

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.

In [7]:
```
# Your code here.
x_init  = 5
y_init  = 10
theta_init = 0
v_init     = 2
a_input    = 1
phi_input  = 0.5

state_init = [x_init, y_init, theta_init, v_init]
state_predictions_nonlinear = nonlinear_vehicle_model(state_init, [a_inp
state_predictions_linear = linear_vehicle_model(A, B, state_init, [a_inp

make_model_comparison_plot(state_predictions_nonlinear, state_prediction
print(state_predictions_nonlinear[10])
print(state_predictions_linear[10])
```




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

In [ ]: