# Week 6

Tuesday, April 20, 2021 10:26 AM

RoboWeek 6

# Chapter 26

# Robo Week 6: LIDAR and Line Fitting

## Schedule

## Learning Objectives

*Concepts*

- Convert vectors from 2D Cartesian to Polar coordinates (and vice versa).
- Explain how Linear Regression produces a line of best fit to a 2D dataset.
- Explain how PCA produces a line of best fit to a 2D dataset.

*MATLAB skills*

- Eliminate 0-entries from LIDAR data.
- Plot LIDAR data on a polar coordinate system.
- Generate best-fit lines to LIDAR data using linear regression and principal component analysis.

## 26.1 Introduction to LIDAR

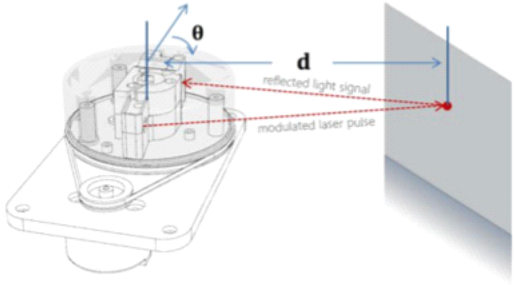


Figure 26.1: A schematic of the Neato's laser scanner. Due to the relative position of the emitter and detector, the position of the reflected light changes predictably as a function of $d$. The laser scanner spins with a rate of 5Hz. The angle of the scanner is denoted as $\theta$.

Laser scanners are becoming an increasingly important sensor in modern robotics. A key accelerator of this trend is the development of self-driving cars, which use laser scanners as their principal source of information about the world around them (e.g., check out this video of a 3D map of a highway in the San Francisco Bay Area).

Your Neato comes with a much more basic laser scanner than the long-range 3D scanners you see in self-driving cars. The Neato's laser scanner provides an estimate of the distance to nearby features in the environment (e.g., obstacles) at a rate of 5 Hz, with an angular resolution of 1 degree, and a maximum usable range of about 3 meters. Under the hood, the sensor works by emitting a laser and then detecting the reflected laser light using a camera. From the known geometry of the camera / laser pair and the position of the reflected laser beam in the camera image, the depth of the obstacle can be recovered (see Figure 26.1). Further, the entire laser / camera assembly spins, providing an estimate of the distance to features in the environment in a full 360-degree sweep around your robot.

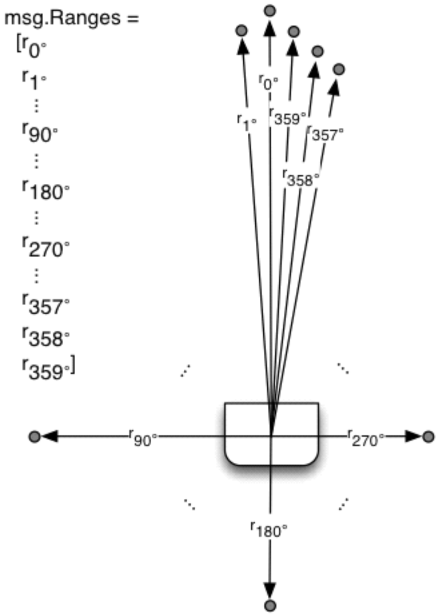


Figure 26.2: A schematic of the laser scan data. In the figure, $r_{\theta^\circ}$ indicates the distance in meters to a detected object at a bearing of $\theta$ degrees. `msg.Ranges` shows how each of these distances maps to a particular index in the array. The array holds the distances, and the angles are implicitly encoded by the position in which they appear in the array. If no detection is made at a particular bearing, the value $0.0$ is used instead. Note: the angles of the measurements around $\theta = 0^\circ$ have been exaggerated for visualization purposes (i.e. the angles shown are larger than a single degree).

Sometimes it is useful to take the raw points detected by the laser scanner and interpret them as geometrical objects (e.g., lines, curves, polygons). For instance, given the laser scan in Figure 26.3 we may wish to automatically interpret the laser scan data (i.e. pick out lines, circles, etc.).

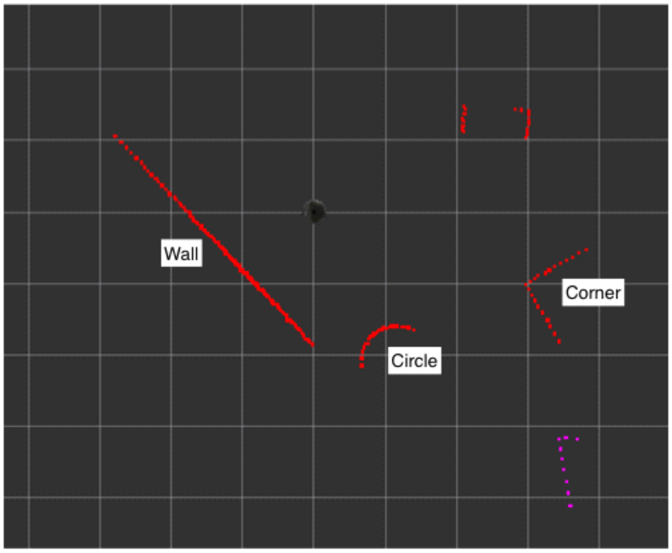


Figure 26.3: A laser scan with high-level features identified.

Conveniently, throughout the class we've built a lot of the machinery needed to do just this (aren't we sneaky?). Here, you'll have a chance to bring it all together.

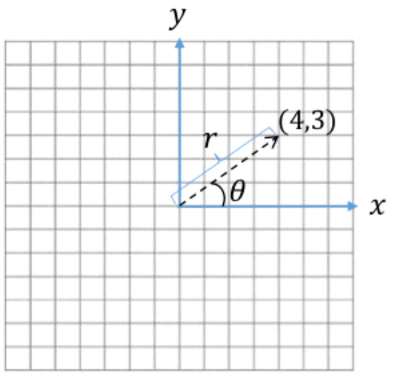


Figure 26.4: Illustration of vector in polar and Cartesian coordinates.

## 26.2 Polar and Cartesian Coordinates

The output data from many sensors, including LIDAR data, is typically presented in polar coordinates. In many situations, however, it is more convenient to represent the data in Cartesian coordinates. In this set of exercises, you will get some practice converting polar to Cartesian coordinates and vice-versa.

In Cartesian coordinates, a point in 2-D space can be described as a sum of unit vectors in the $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ directions, which are unit vectors in the directions of the $x$ and $y$ axes respectively.

Consider the point $(4,3)$ illustrated in Figure 26.4. You can write the vector representing this point as a sum of $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ as follows

$$\begin{bmatrix}4\\3\end{bmatrix} = 4\hat{\mathbf{i}} + 3\hat{\mathbf{j}} \tag{26.1}$$

We can also write this point in terms of polar co-ordinates as $(r,\theta)$, where

$$r = \sqrt{4^2+3^2} = 5 \tag{26.2}$$

$$\theta = \tan^{-1}\left(\frac{3}{4}\right) \tag{26.3}$$

In general a vector $(x,y)$ in Cartesian coordinates can be expressed in polar coordinates as $(r,\theta)$ where

$$r = \sqrt{x^2+y^2} \tag{26.4}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{26.5}$$

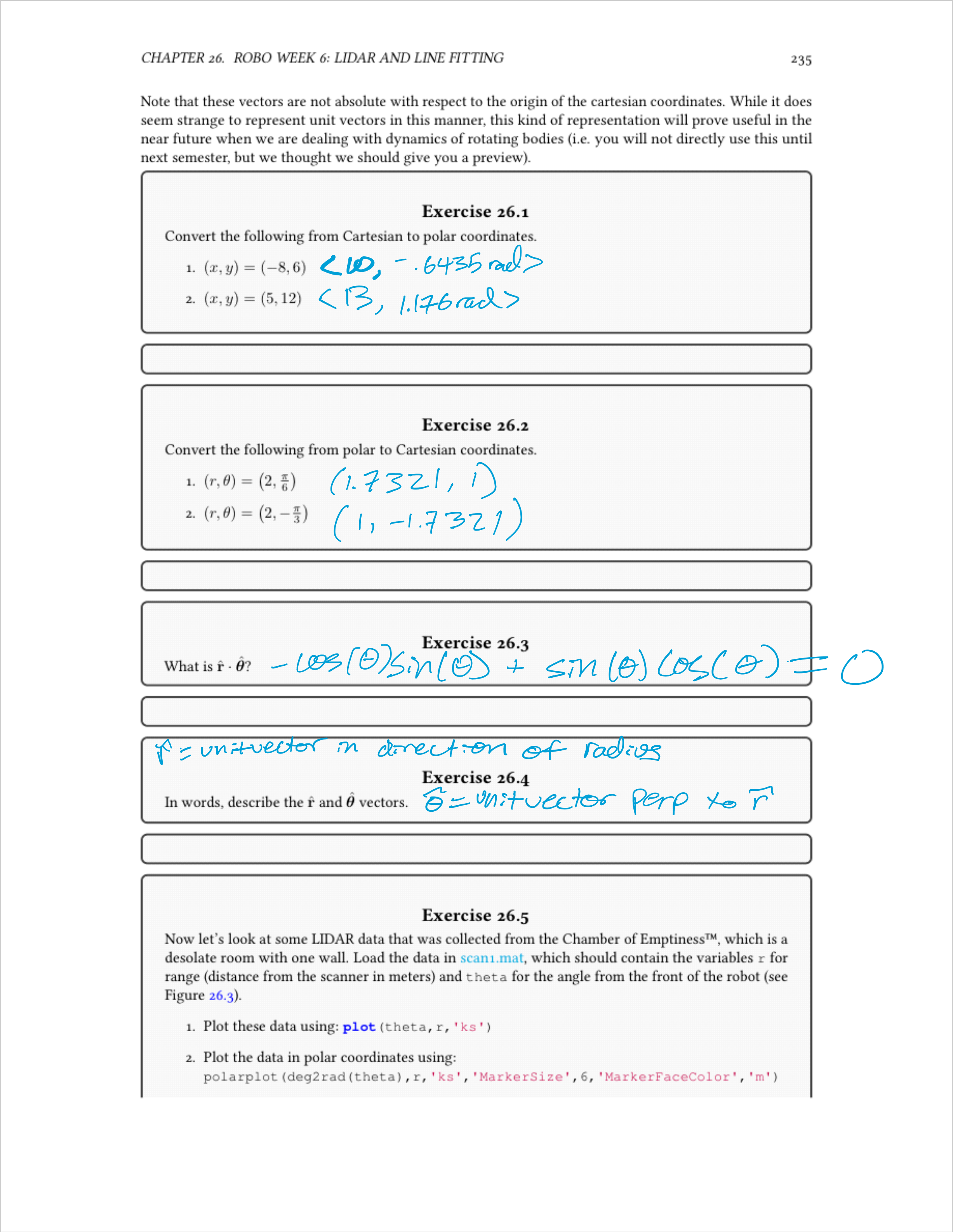
To go from polar to Cartesian coordinates, we can use the following formulas

$$x = r\cos\theta$$
$$y = r\sin\theta$$

Besides expressing a vector as a linear combination of the $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ vectors, we can also define two new unit vectors $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ defined as follows

$$\hat{\mathbf{r}} = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} \tag{26.6}$$

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix}-\sin\theta\\ \cos\theta\end{bmatrix} \tag{26.7}$$



Note that these vectors are not absolute with respect to the origin of the cartesian coordinates. While it does seem strange to represent unit vectors in this manner, this kind of representation will prove useful in the near future when we are dealing with dynamics of rotating bodies (i.e. you will not directly use this until next semester, but we thought we should give you a preview).

### Exercise 26.1

Convert the following from Cartesian to polar coordinates.

1. $(x, y) = (-8, 6)$ $<10, -.6435 \text{ rad}>$
2. $(x, y) = (5, 12)$ $<13, 1.176 \text{ rad}>$

### Exercise 26.2

Convert the following from polar to Cartesian coordinates.

1. $(r, \theta) = (2, \frac{\pi}{6})$ $(1.7321, 1)$
2. $(r, \theta) = (2, -\frac{\pi}{3})$ $(1, -1.7321)$

### Exercise 26.3

What is $\hat{\mathbf{r}} \cdot \hat{\boldsymbol{\theta}}$? $-\cos(\theta)\sin(\theta) + \sin(\theta)\cos(\theta) = 0$

$\hat{r}$ = unitvector in direction of radius

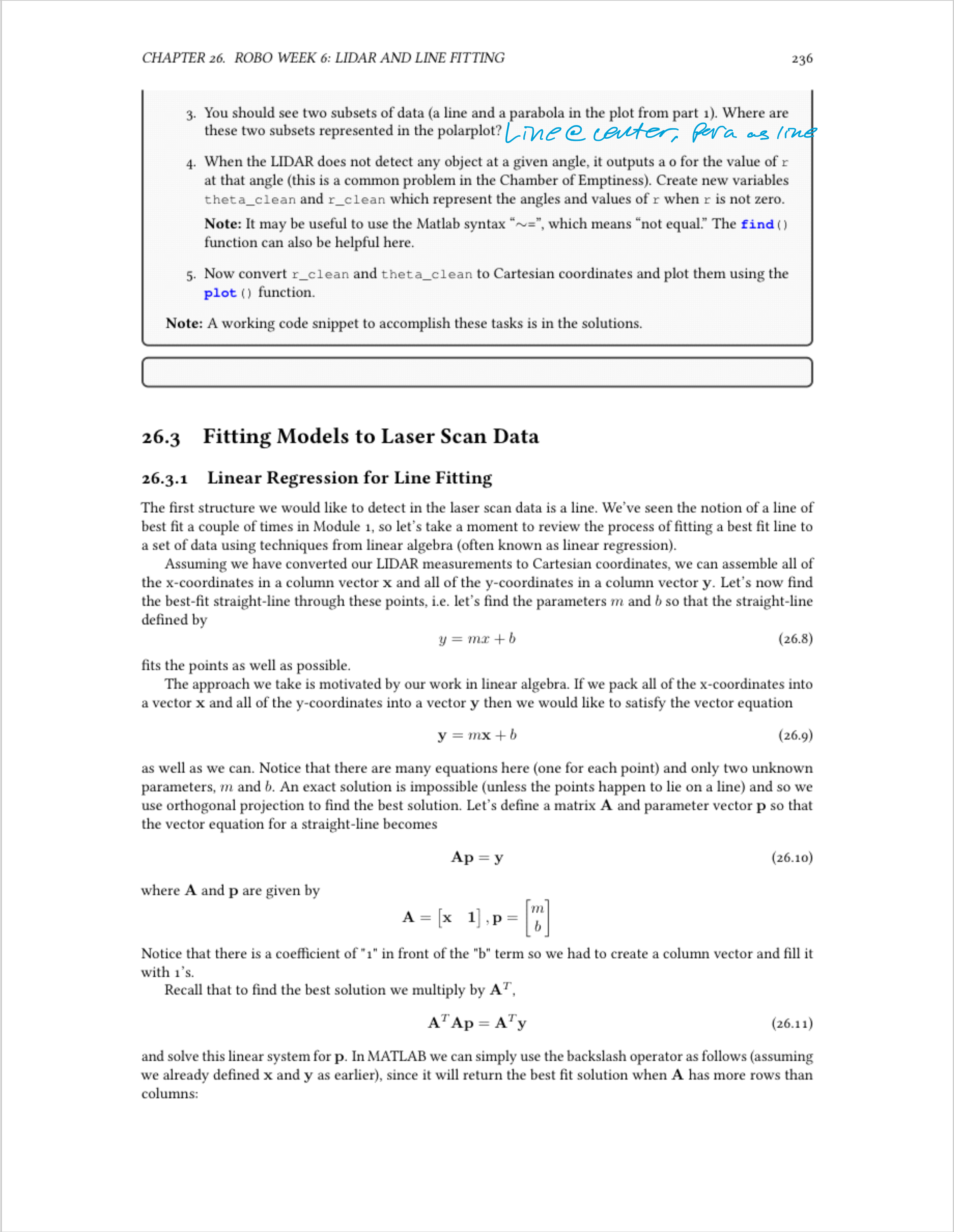
### Exercise 26.4

In words, describe the $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ vectors. $\hat{\theta}$ = unitvector perp to $\vec{r}$

### Exercise 26.5

Now let's look at some LIDAR data that was collected from the Chamber of Emptiness™, which is a desolate room with one wall. Load the data in scan1.mat, which should contain the variables `r` for range (distance from the scanner in meters) and `theta` for the angle from the front of the robot (see Figure 26.3).

1. Plot these data using: `plot(theta,r,'ks')`
2. Plot the data in polar coordinates using: `polarplot(deg2rad(theta),r,'ks','MarkerSize',6,'MarkerFaceColor','m')`



3. You should see two subsets of data (a line and a parabola in the plot from part 1). Where are these two subsets represented in the polarplot? Line @ center, Para as line

4. When the LIDAR does not detect any object at a given angle, it outputs a 0 for the value of `r` at that angle (this is a common problem in the Chamber of Emptiness). Create new variables `theta_clean` and `r_clean` which represent the angles and values of `r` when `r` is not zero.

   **Note:** It may be useful to use the Matlab syntax "∼=", which means "not equal." The `find()` function can also be helpful here.

5. Now convert `r_clean` and `theta_clean` to Cartesian coordinates and plot them using the `plot()` function.

**Note:** A working code snippet to accomplish these tasks is in the solutions.

## 26.3 Fitting Models to Laser Scan Data

### 26.3.1 Linear Regression for Line Fitting

The first structure we would like to detect in the laser scan data is a line. We've seen the notion of a line of best fit a couple of times in Module 1, so let's take a moment to review the process of fitting a best fit line to a set of data using techniques from linear algebra (often known as linear regression).

Assuming we have converted our LIDAR measurements to Cartesian coordinates, we can assemble all of the x-coordinates in a column vector $\mathbf{x}$ and all of the y-coordinates in a column vector $\mathbf{y}$. Let's now find the best-fit straight-line through these points, i.e. let's find the parameters $m$ and $b$ so that the straight-line defined by

$$y = mx + b \tag{26.8}$$

fits the points as well as possible.

The approach we take is motivated by our work in linear algebra. If we pack all of the x-coordinates into a vector $\mathbf{x}$ and all of the y-coordinates into a vector $\mathbf{y}$ then we would like to satisfy the vector equation

$$\mathbf{y} = m\mathbf{x} + b \tag{26.9}$$

as well as we can. Notice that there are many equations here (one for each point) and only two unknown parameters, $m$ and $b$. An exact solution is impossible (unless the points happen to lie on a line) and so we use orthogonal projection to find the best solution. Let's define a matrix $\mathbf{A}$ and parameter vector $\mathbf{p}$ so that the vector equation for a straight-line becomes

$$\mathbf{A}\mathbf{p} = \mathbf{y} \tag{26.10}$$

where $\mathbf{A}$ and $\mathbf{p}$ are given by

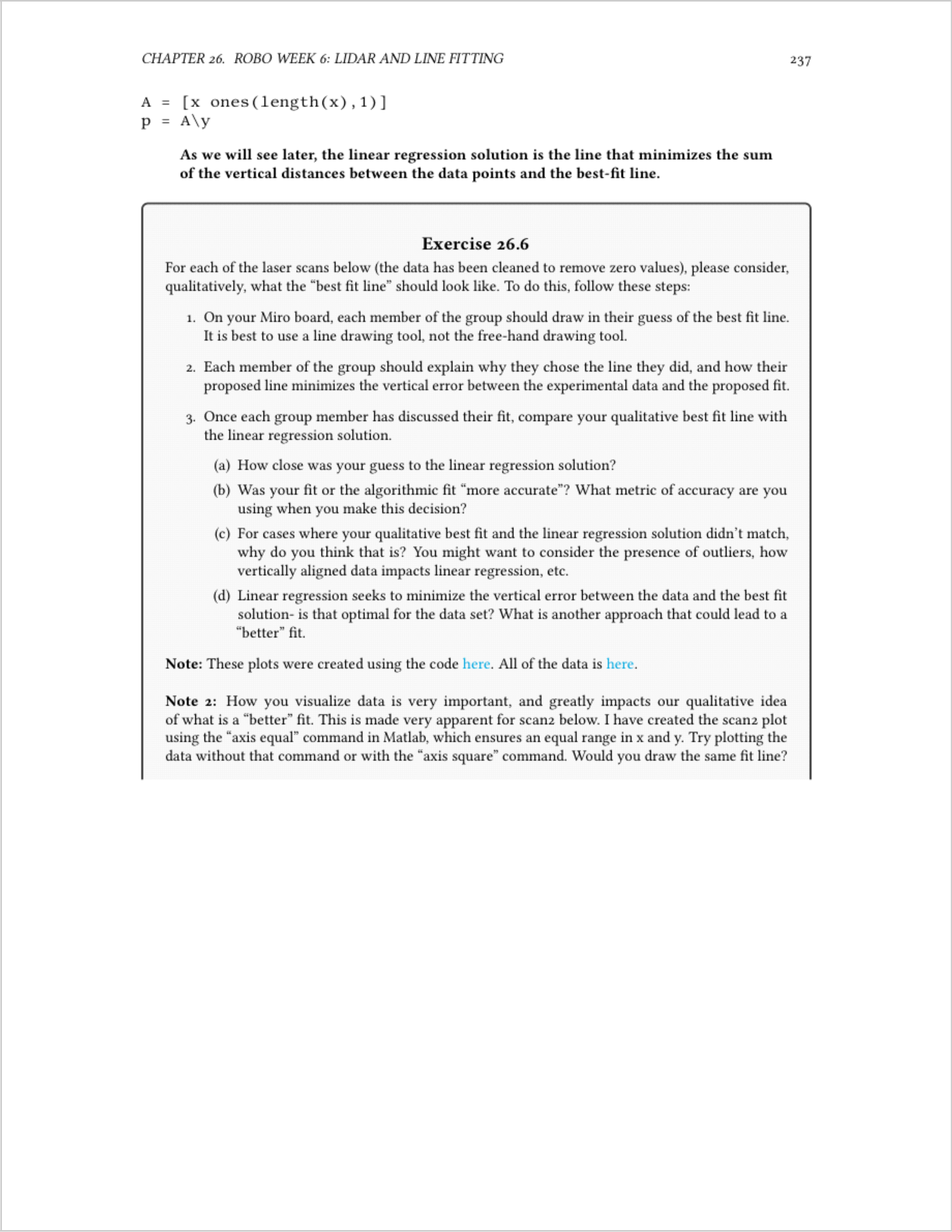
$$\mathbf{A} = \begin{bmatrix} \mathbf{x} & \mathbf{1} \end{bmatrix}, \mathbf{p} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Notice that there is a coefficient of "1" in front of the "b" term so we had to create a column vector and fill it with 1's.

Recall that to find the best solution we multiply by $\mathbf{A}^T$,

$$\mathbf{A}^T\mathbf{A}\mathbf{p} = \mathbf{A}^T\mathbf{y} \tag{26.11}$$

and solve this linear system for $\mathbf{p}$. In MATLAB we can simply use the backslash operator as follows (assuming we already defined $\mathbf{x}$ and $\mathbf{y}$ as earlier), since it will return the best fit solution when $\mathbf{A}$ has more rows than columns:



```
A = [x ones(length(x),1)]
p = A\y
```

**As we will see later, the linear regression solution is the line that minimizes the sum of the vertical distances between the data points and the best-fit line.**

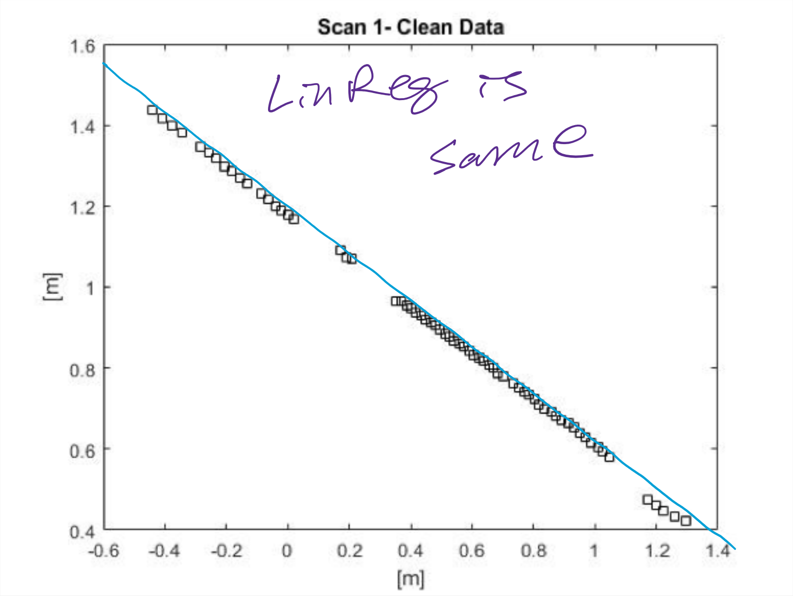
### Exercise 26.6

For each of the laser scans below (the data has been cleaned to remove zero values), please consider, qualitatively, what the "best fit line" should look like. To do this, follow these steps:

1. On your Miro board, each member of the group should draw in their guess of the best fit line. It is best to use a line drawing tool, not the free-hand drawing tool.
2. Each member of the group should explain why they chose the line they did, and how their proposed line minimizes the vertical error between the experimental data and the proposed fit.
3. Once each group member has discussed their fit, compare your qualitative best fit line with the linear regression solution.
   (a) How close was your guess to the linear regression solution?
   (b) Was your fit or the algorithmic fit "more accurate"? What metric of accuracy are you using when you make this decision?
   (c) For cases where your qualitative best fit and the linear regression solution didn't match, why do you think that is? You might want to consider the presence of outliers, how vertically aligned data impacts linear regression, etc.
   (d) Linear regression seeks to minimize the vertical error between the data and the best fit solution- is that optimal for the data set? What is another approach that could lead to a "better" fit.

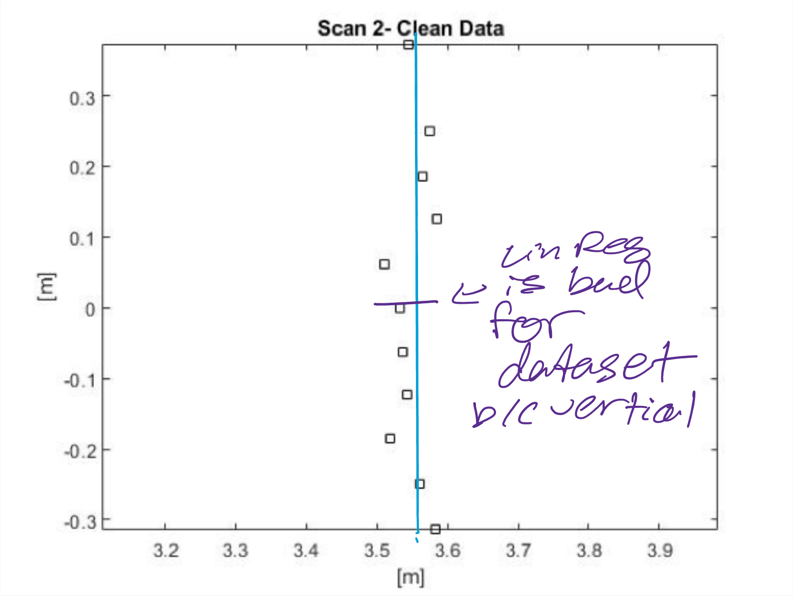
**Note:** These plots were created using the code here. All of the data is here.

**Note 2:** How you visualize data is very important, and greatly impacts our qualitative idea of what is a "better" fit. This is made very apparent for scan2 below. I have created the scan2 plot using the "axis equal" command in Matlab, which ensures an equal range in x and y. Try plotting the data without that command or with the "axis square" command. Would you draw the same fit line?



Lin Reg is same

Figure 26.5: Clean data from Scan1. Please add your fit lines using Miro.



← Lin Reg is bad for dataset b/c vertical

Figure 26.6: Clean data from Scan2. Please add your fit lines using Miro.

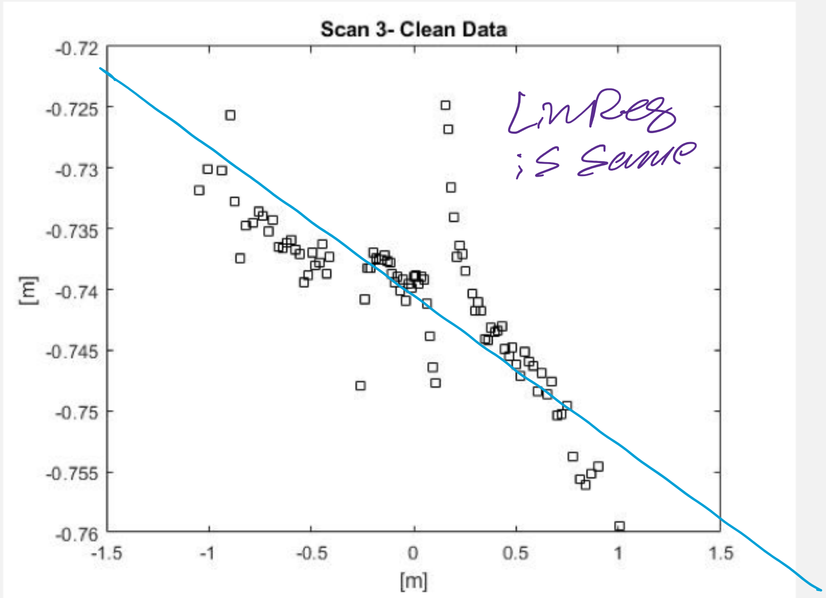


Figure 26.7: Clean data from Scan3. Please add your fit lines using Miro.

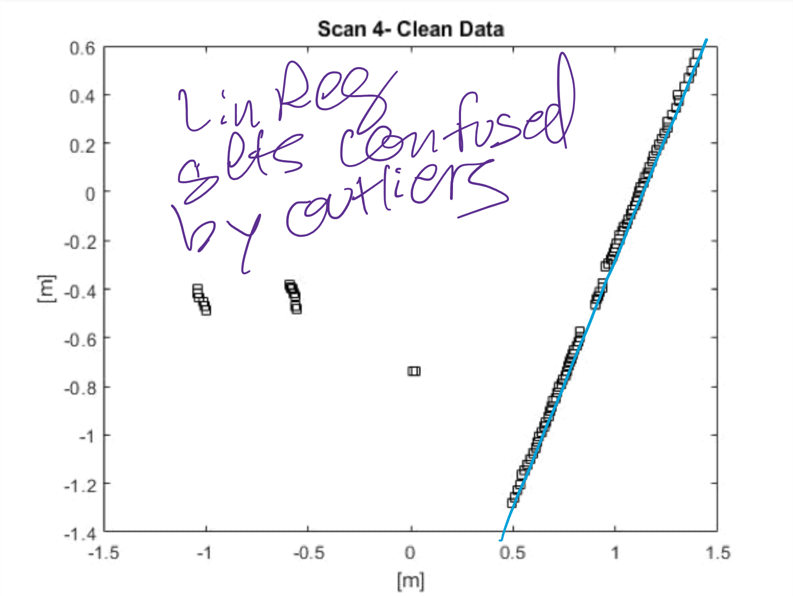


Figure 26.8: Clean data from Scan4. Please add your fit lines using Miro.

### 26.3.2 Principal Component Analysis for Line Fitting

Linear regression defines the line of best fit in terms of the sum of squared distances, measured vertically, between the data points and the line (see Figure 26.13). However, when determining the best fitting line for a laser scan, there's nothing special about the vertical direction. In fact, it would make as much sense to minimize the sum of the squared horizontal distances. In this context, a much more natural way to think about the line of best fit is to minimize the perpendicular distances between the line and the data points (see Figure 26.13).

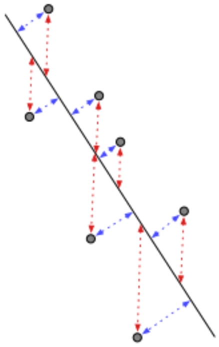


Figure 26.13: Laser scan points (gray dots), a proposed fitted line, and two different notions of error. The red dashed lines provide the vertical distance (which is used in linear regression). The blue dashed lines provide the perpendicular distance which is a more natural way to define line of best fit in the context of fitting lines to laser scan data.

### Exercise 26.7

Think about how to use principal component analysis to determine the line of best fit that minimizes the sum of squared perpendicular distances as shown by the blue dashed lines in Figure 26.13. You might recall that in PCA, we found the directions of greatest variation in an $(x, y)$ dataset by first creating a matrix of mean-centered data,

$$\mathbf{A} = \frac{1}{\sqrt{N-1}} \begin{pmatrix} x_1 - \mu_x & y_1 - \mu_y \\ x_2 - \mu_x & y_2 - \mu_y \\ x_3 - \mu_x & y_3 - \mu_y \\ \vdots & \vdots \\ x_N - \mu_x & y_N - \mu_y \end{pmatrix} \tag{26.12}$$

and then finding the eigenvectors of the co-variance matrix, $\mathbf{R} = \mathbf{A}^T\mathbf{A}$.

→ PCA is the natural axis(es) of a dataset, the most significant axis is the LOBF

1. Describe conceptually why you think this algorithm will achieve the desired goal.
2. Write pseudocode to describe the steps you will need to take. You should consider what an eigenvector represents. (Hint: if you translate your data, be sure to translate it back. )

→ $R = A^TA$

$V = eigs(R, 1)$

$V(:) + M$

### Exercise 26.8

In MATLAB, do the following for the Cartesian data for scans scan2.mat and scan3.mat.
**Note:** Make use of the code snippets above to remove zeroes and convert the data from polar to Cartesian coordinates. To find the best fit line using PCA, the `pca` function in Matlab will be useful. Remember you can always use `help pca` or `doc pca` to learn the usage details of the `pca` function. Once you have attempted the Matlab solution as a group, a working solution is here.

1. Compute the line of best fit using PCA.

2. Plot the data points, line of best fit from PCA, and line of best fit from linear regression. Be sure to note which line is which.

3. How does the fit using PCA differ from your qualitative fit from above, and from the fit using linear regression? Why do you think this is? Discuss how the difference between linear regressions minimizing vertical error, and PCA identifying the direction of greatest (and least) variation impacts the fit lines.

## 26.4 Least-Squares (Optional)

In least squares fitting, we want to find the line, $y = mx + b$, which lies 'closest' to all our data points. To express this quantitatively, we want to find the line which minimizes the mean of the squared error of our data set from the line. Recalling our definition of the mean, we can rewrite the mean-square error as:

$$MSE(m,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2 \tag{26.13}$$

The key insight to make here is that the mean-squared error is a function of two variables, $m$ and $b$.

The process of finding the best-fit straight line therefore amounts to finding the values of $m$ and $b$ that minimize the mean-square error function. Since the mean-square error is a function of two variables we need to compute the partial derivatives, and then set of them equal to zero in order to find the critical points of the function:

3. Show that the partial derivative of the MSE function are:

$$\frac{\partial MSE}{\partial m} = -2\frac{1}{N}\sum_{i=1}^{N}x_i(y_i - (mx_i + b)) \tag{26.14}$$

$$\frac{\partial MSE}{\partial b} = -2\frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b)) \tag{26.15}$$

4. Show that setting each partial derivative equal to zero results in the following equations (after some simplifying):

$$m\sum_{i=1}^{N}x_i^2 + b\sum_{i=1}^{N}x_i = \sum_{i=1}^{N}x_iy_i \tag{26.16}$$

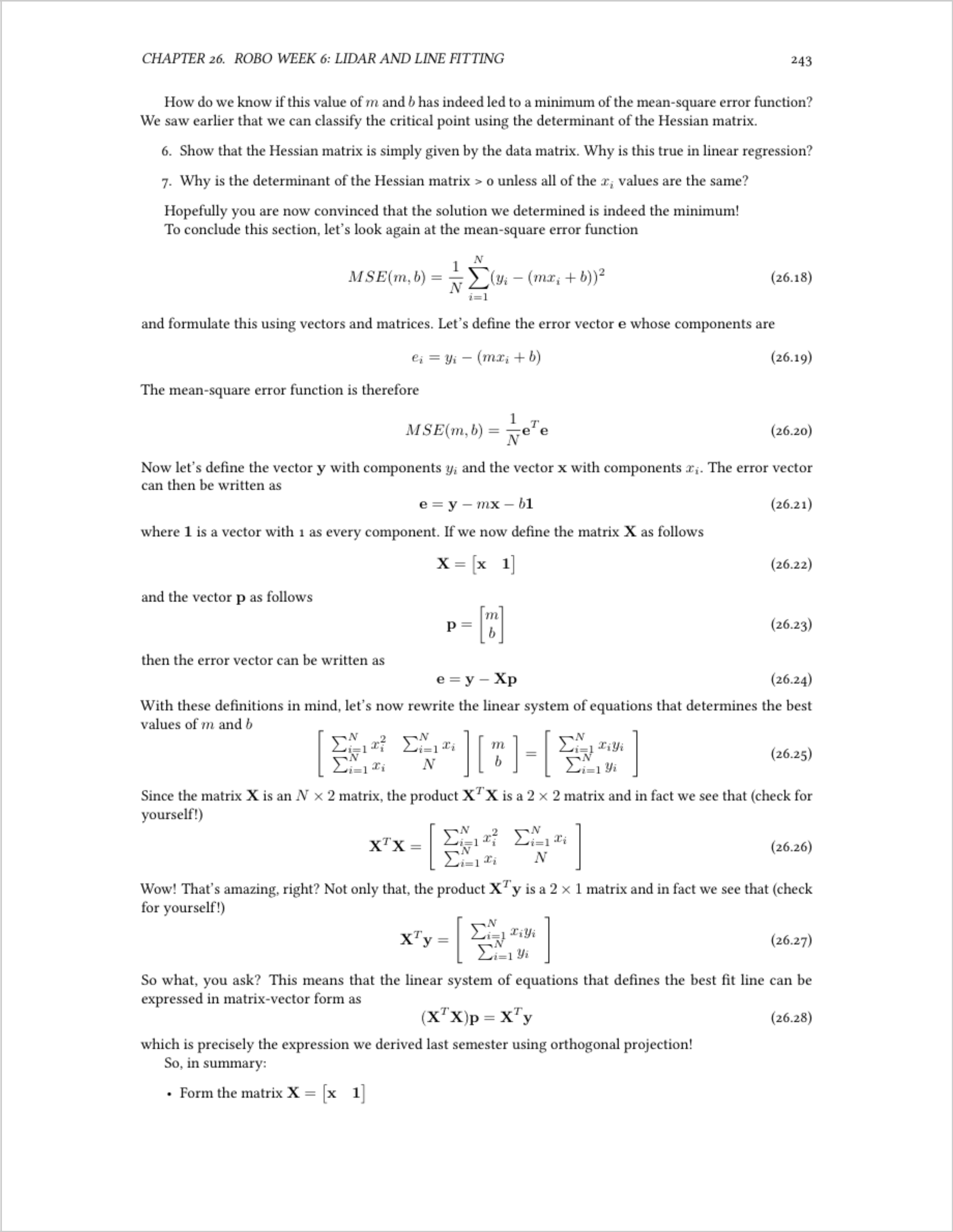
$$m\sum_{i=1}^{N}x_i + Nb = \sum_{i=1}^{N}y_i \tag{26.17}$$

Notice that both of these equations are `linear` in $m$ and $b$, and we can therefore write this system of linear algebraic equations in matrix-vector form,

$$\begin{bmatrix} \sum_{i=1}^{N}x_i^2 & \sum_{i=1}^{N}x_i \\ \sum_{i=1}^{N}x_i & N \end{bmatrix}\begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N}x_iy_i \\ \sum_{i=1}^{N}y_i \end{bmatrix}$$

and find the solution for $m$ and $b$ by solving this linear system.

5. Under what data conditions does a solution not exist to this problem, i.e. when is the determinant of the data matrix zero?



How do we know if this value of $m$ and $b$ has indeed led to a minimum of the mean-square error function? We saw earlier that we can classify the critical point using the determinant of the Hessian matrix.

6. Show that the Hessian matrix is simply given by the data matrix. Why is this true in linear regression?
7. Why is the determinant of the Hessian matrix > 0 unless all of the $x_i$ values are the same?

Hopefully you are now convinced that the solution we determined is indeed the minimum!

To conclude this section, let's look again at the mean-square error function

$$MSE(m,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2 \tag{26.18}$$

and formulate this using vectors and matrices. Let's define the error vector $\mathbf{e}$ whose components are

$$e_i = y_i - (mx_i + b) \tag{26.19}$$

The mean-square error function is therefore

$$MSE(m,b) = \frac{1}{N}\mathbf{e}^T\mathbf{e} \tag{26.20}$$

Now let's define the vector $\mathbf{y}$ with components $y_i$ and the vector $\mathbf{x}$ with components $x_i$. The error vector can then be written as

$$\mathbf{e} = \mathbf{y} - m\mathbf{x} - b\mathbf{1} \tag{26.21}$$

where $\mathbf{1}$ is a vector with 1 as every component. If we now define the matrix $\mathbf{X}$ as follows

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{1} \end{bmatrix} \tag{26.22}$$

and the vector $\mathbf{p}$ as follows

$$\mathbf{p} = \begin{bmatrix} m \\ b \end{bmatrix} \tag{26.23}$$

then the error vector can be written as

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{p} \tag{26.24}$$

With these definitions in mind, let's now rewrite the linear system of equations that determines the best values of $m$ and $b$

$$\begin{bmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & N \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} y_i \end{bmatrix} \tag{26.25}$$

Since the matrix $\mathbf{X}$ is an $N \times 2$ matrix, the product $\mathbf{X}^T\mathbf{X}$ is a $2 \times 2$ matrix and in fact we see that (check for yourself!)

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & N \end{bmatrix} \tag{26.26}$$

Wow! That's amazing, right? Not only that, the product $\mathbf{X}^T\mathbf{y}$ is a $2 \times 1$ matrix and in fact we see that (check for yourself!)

$$\mathbf{X}^T\mathbf{y} = \begin{bmatrix} \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} y_i \end{bmatrix} \tag{26.27}$$

So what, you ask? This means that the linear system of equations that defines the best fit line can be expressed in matrix-vector form as

$$(\mathbf{X}^T\mathbf{X})\mathbf{p} = \mathbf{X}^T\mathbf{y} \tag{26.28}$$

which is precisely the expression we derived last semester using orthogonal projection!

So, in summary:

- Form the matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x} & \mathbf{1} \end{bmatrix}$

- Compute the matrix product $\mathbf{X}^T\mathbf{X}$
- Compute the vector $\mathbf{X}^T\mathbf{y}$
- Solve the linear system of equations defined by $(\mathbf{X}^T\mathbf{X})\mathbf{p} = \mathbf{X}^T\mathbf{y}$ for the vector $\mathbf{p}$.
- Compute the error vector $\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{p}$ using this solution for $\mathbf{p}$.
- Evaluate the mean square error function, MSE = $\frac{1}{N}\mathbf{e}^T\mathbf{e}$

### Solution 26.1

1. $\theta$=2.5 radians, R=10
2. $\theta$=1.18 radians, R=13

### Solution 26.2

1. x=1.73, y=1.00
2. x=1, y=-1.73

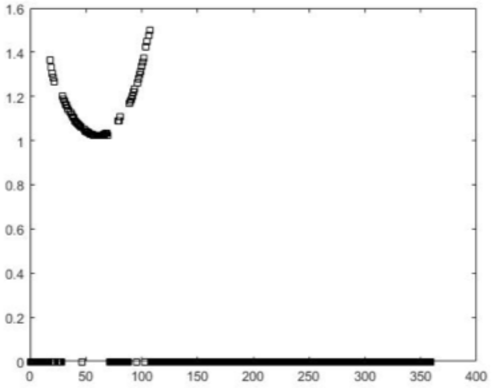
### Solution 26.3

$\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ are orthogonal, so their dot product is zero.

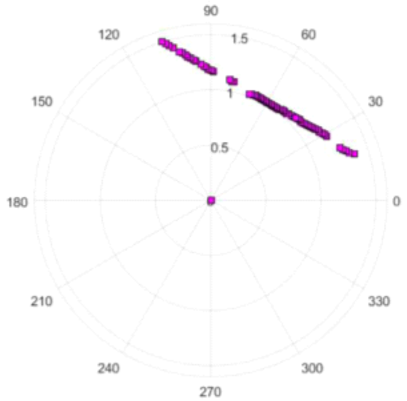
### Solution 26.4

The $\hat{\mathbf{r}}$ vector has length 1 and points in the direction of increasing $r$, and $\hat{\boldsymbol{\theta}}$ has length 1 and points in the direction of increasing $\theta$, orthogonal to $\hat{\mathbf{r}}$.

### Solution 26.5



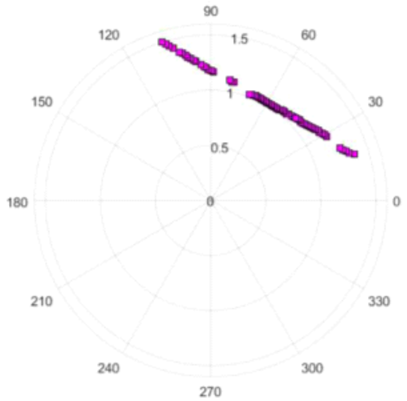
1.

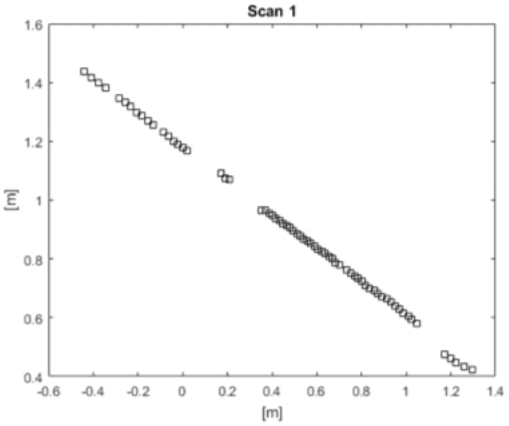


2.

3. The samples making up the line in part a are all at zero distance, and appear as a cluster and the center of the polar plot. The parabola from part a becomes the line in part b.

4. one implementation:
   index=find(r∼ =0);
   r_clean=r(index);
   theta_clean=theta(index);





5.

An example code section that could do this whole process is here.

### Solution 26.6

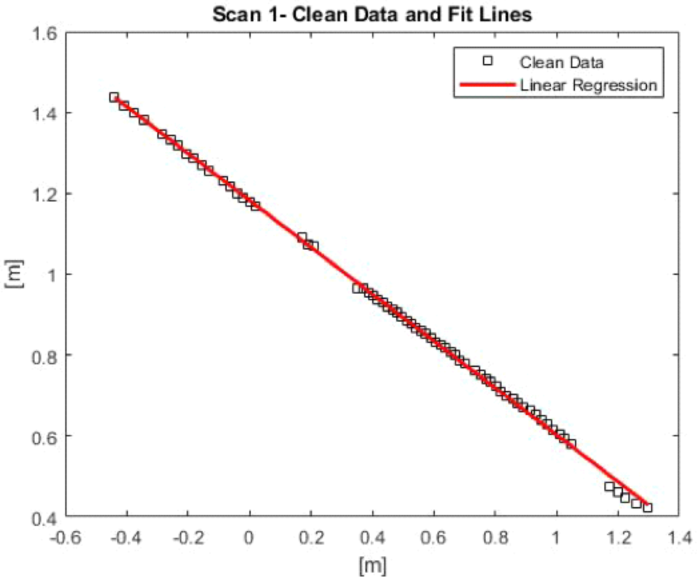


Figure 26.9: Best fit using linear regression for scan1.

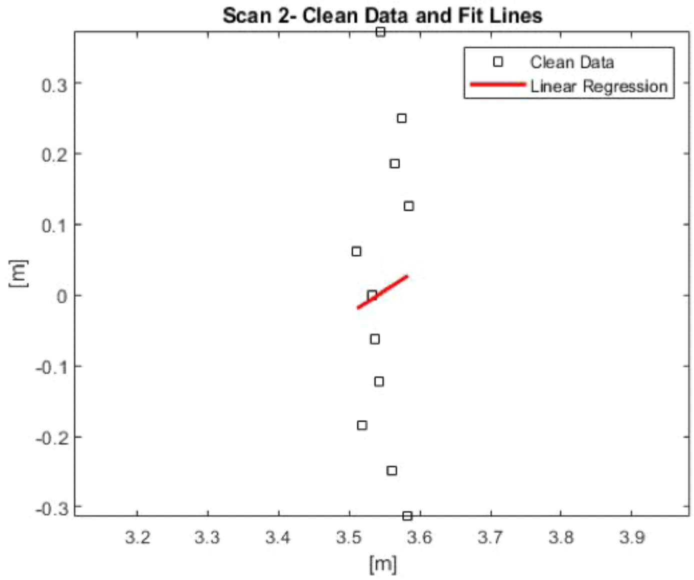


Figure 26.10: Best fit using linear regression for scan2.

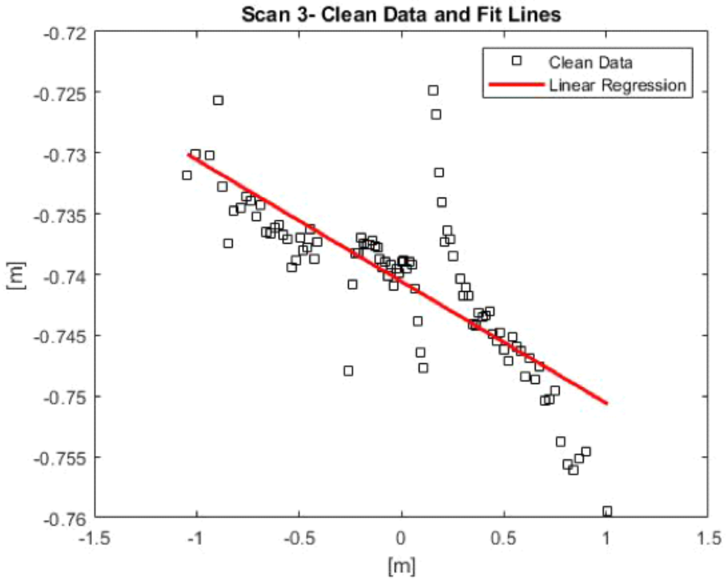


Figure 26.11: Best fit using linear regression for scan3.

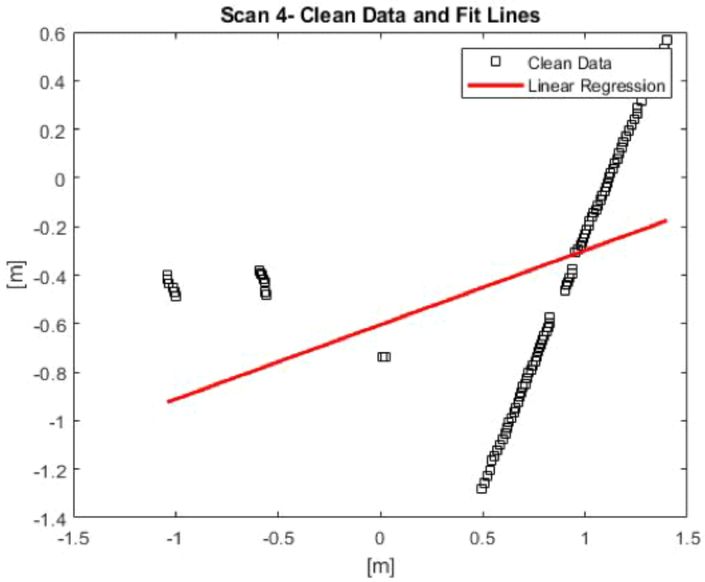


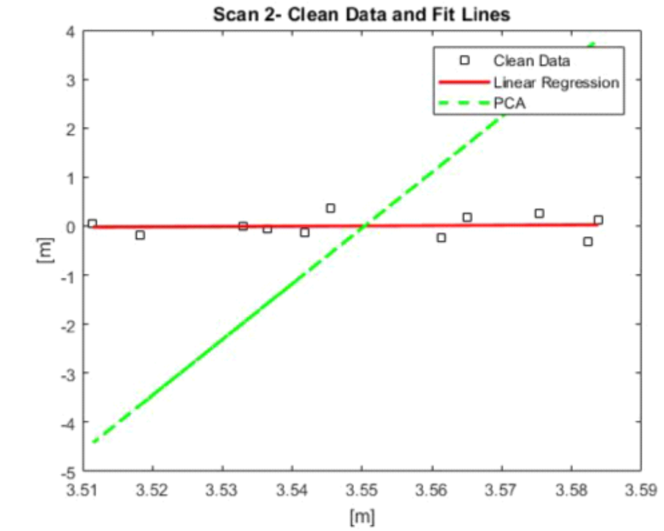
Figure 26.12: Best fit using linear regression for scan4.

### Solution 26.7

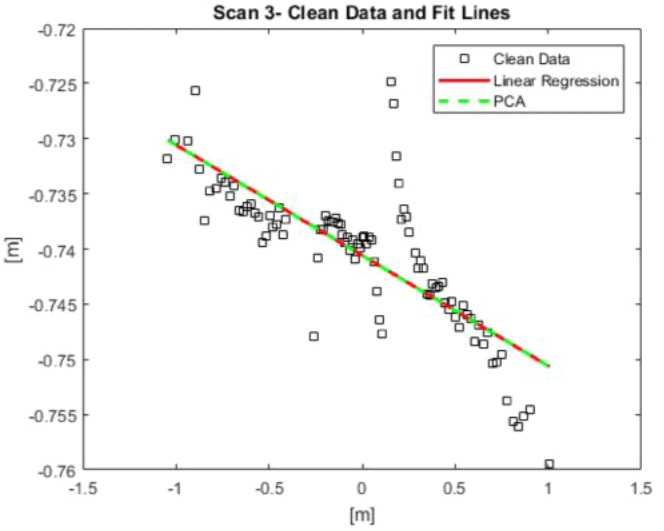
1. We can recall from Module 1 that PCA identifies the direction of greatest variation in a data set (this direction is given by the eigenvector associated with the largest eigenvalue). Conceptually, the direction of *greatest* variation should be orthogonal to the direction of *least* variation. So, the direction perpendicular, or orthogonal, to the line found using PCA should minimize variation, and therefore minimize the sum of squared perpendicular distances.

2. (a) Convert data from polar to cartesian coordinates.
   (b) Calculate the x and y means for the dataset.
   (c) Use the x and y means to center the data.
   (d) Find the singular value decomposition (SVD).
   (e) Identify the direction of greatest variation in the dataset as the eigenvector associated with the largest eigenvalue.
   (f) Translate the data back to its original positions.

### Solution 26.8



Clean data and best fit lines using linear regression and PCA for Scan 2.



Clean data and best fit lines using linear regression and PCA for Scan 3.

Chapter 27

# Homework 6: Frames of Reference and LIDAR

## Schedule

## Learning Objectives

*Concepts*

- Create a matrix that rotates a vector from one 2D coordinate system to another that shares its origin but is rotated an angle, $\theta$, clockwise, relative to the original coordinate system.
- Create a matrix that translates the origin of a 2D coordinate system to another known point.

*MATLAB skills*

- Compute the vector coordinates of a vector in a global matrix to a rotated coordinate system with a translated origin.
- Determine the location of the NEATO in the global frame in response to LIDAR data.

## Overview

In this overnight activity you are going to think about transforming points between different frames of reference, and how to apply this data collected by the LIDAR on the NEATO in order to create a "map" of the room. Rather than translating and rotating points, we will be using translation and rotation matrices to express points in different frames of reference. For example, imagine we want to refer to a point (a,b,c) in a global frame (G), from within a different frame (M), as shownn in Figure 27.1. Our task would be to translate the origin and transform the G-frame to the M-frame. How would we do that? Let's first remember how we might transform the frames when they share an origin.

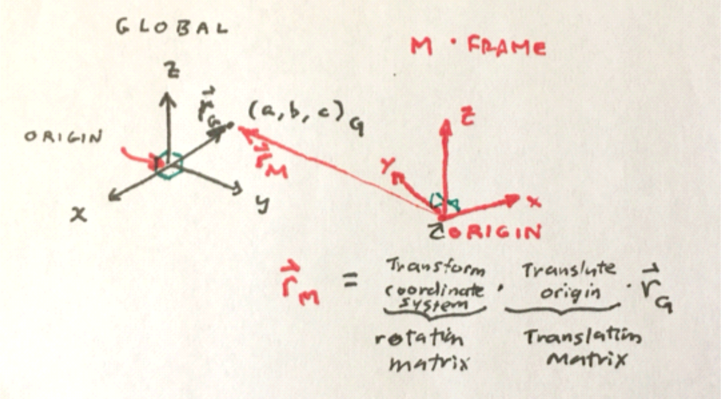

GLOBAL
M · FRAME
Z
$\vec{r}_G$
$(a,b,c)_G$
ORIGIN
$\vec{r}_M$
X
y
$Y_R$
Z
x
Z ORIGIN
$\vec{r}_M = \underbrace{\text{Transform coordinate system}}_{\text{rotatin matrix}} \cdot \underbrace{\text{Translute origin}}_{\text{Translatin Matrix}} \cdot \vec{r}_G$

Figure 27.1: The global frame of reference (G) differs in origin and shape from the M-frame.

## 27.1 Frames of Reference

In robotics and indeed many other applications, it is often useful to express points using different coordinate systems. You have already done this to a certain extent in Module 1 when you expressed vectorized images as linear combinations of Eigenfaces. In robotics applications, it is beneficial to express points relative to a fixed origin (e.g. a point designated as the origin in a room) and orthogonal basis vectors. In other cases, it is convenient to express points relative to an origin on the robot itself, with one basis vector in the direction of motion, and the others in orthogonal directions. Here, we will first develop some tools to translate points from one coordinate system to another, and apply these tools in the context of the NEATO. You will then be able to take points expressed in a coordinate system centered on the robot (e.g. from sensor data), and represent them in terms of a fixed coordinate system (e.g. with a corner of a room as the origin). We will work in 2D, but we can just as easily extend this discussion to higher dimensions.

### 27.1.1 Coordinate Systems with the Same Origin

A coordinate system consists of an **origin** and a set of **basis vectors**. Recall that a set of vectors form a basis if they are linearly independent, and if they are mutually orthogonal then we have an orthogonal coordinate system.

The standard basis vectors for 2D are usually labelled $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$, but they are sometimes written as $\mathbf{e}_1$ and $\mathbf{e}_2$, or $\mathbf{e}_x$ and $\mathbf{e}_y$, or $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$. From now on, we will assume that the global reference frame (frame G) is defined by the standard basis vectors, but just to be very clear we will use a subscript and write $\hat{\mathbf{i}}_G$ and $\hat{\mathbf{j}}_G$ as the basis vectors of the global frame G.

Points in 2D are expressed in terms of the basis vectors, and we refer to the component of the vector as the **coordinates** of the point. For example, the point A in Figure 27.2 has coordinates $(4, 3)$ when expressed in terms of the standard basis vectors. We might also write the position vector of $A$ as

$$\mathbf{r}_G = 4\hat{\mathbf{i}}_G + 3\hat{\mathbf{j}}_G$$

or express it as a row or column vector

$$\mathbf{r}_G = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

All of these refer to the same point, but the first and last representations imply the basis vectors, while the second representation makes it explicit. We use the notation $\mathbf{r}_G$ to be clear that this is the position vector of

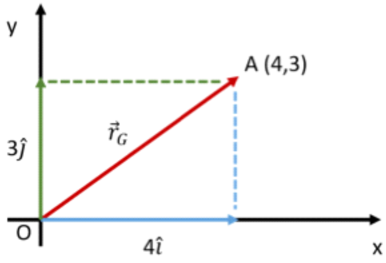


Figure 27.2: The coordinates of the point $(4, 3)$ are the projections of the position vector onto the relevant basis vectors.

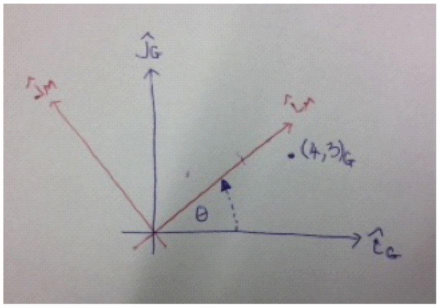


Figure 27.3: The frame M has the same origin, but the basis vectors are rotated by an angle of $\theta$. The coordinates of the point $(4, 3)_G$ can be expressed in terms of the new frame M.

point $A$ when expressed in the frame G. Notice that the coordinates $(x_G, y_G)$ of the point A are simply the projection of the position vector onto the relevant basis vectors,

$$x_G = \mathbf{r}_G \cdot \hat{\mathbf{i}}_G, \; y_G = \mathbf{r}_G \cdot \hat{\mathbf{j}}_G$$

Since these basis vectors are mutually orthogonal the coordinates are simply $(4, 3)$ as expected, but again for clarity we will write $(4, 3)_G$ to mean that these are the coordinates in the frame G. You may recall that we met this concept in detail back in module 1 when we learned about *decomposition*.

How do we express the same point in terms of a new set of basis vectors, $\hat{\mathbf{i}}_M$ and $\hat{\mathbf{j}}_M$? Mathematically, we are trying to express the vector $\mathbf{r}_M$ as a linear combination of these vectors

$$\mathbf{r}_M = x_M \, \hat{\mathbf{i}}_M + y_M \, \hat{\mathbf{j}}_M$$

where the coordinates of the point are now $(x_M, y_M)$, i.e. the x and y coordinates of the point in the frame of reference M. See Figure 27.3.

The components of the point A expressed in this coordinate system is again the projection of the position vector in the frame G onto each basic vector of frame M in turn,

$$x_M = \mathbf{r}_G \cdot \hat{\mathbf{i}}_M, \; y_M = \mathbf{r}_G \cdot \hat{\mathbf{j}}_M$$

Since $\mathbf{r}_G = x_G \, \hat{\mathbf{i}}_G + y_G \, \hat{\mathbf{j}}_G$ we see that

$$x_M = x_G \, \hat{\mathbf{i}}_G \cdot \hat{\mathbf{i}}_M + y_G \, \hat{\mathbf{j}}_G \cdot \hat{\mathbf{i}}_M, \; y_M = x_G \, \hat{\mathbf{i}}_G \cdot \hat{\mathbf{j}}_M + y_G \, \hat{\mathbf{j}}_G \cdot \hat{\mathbf{j}}_M$$

Whilst this looks cumbersome, it becomes a lot clearer when we use the following matrix-vector formulation

$$\begin{bmatrix} x_M \\ y_M \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{i}}_G \cdot \hat{\mathbf{i}}_M & \hat{\mathbf{j}}_G \cdot \hat{\mathbf{i}}_M \\ \hat{\mathbf{i}}_G \cdot \hat{\mathbf{j}}_M & \hat{\mathbf{j}}_G \cdot \hat{\mathbf{j}}_M \end{bmatrix} \begin{bmatrix} x_G \\ y_G \end{bmatrix}$$

This matrix is the transformation matrix from the global reference frame (frame G) to the new reference frame (frame M), and we will often use the notation $\mathbf{R}_{MG}$ when referring to this transformation matrix

$$\mathbf{r}_M = \mathbf{R}_{MG}\mathbf{r}_G$$

For example, consider the frame M shown in Figure 2, which is simply the global frame rotated counter-clockwise by an angle $\theta$. The basis vectors in this frame are

$$\hat{\mathbf{i}}_M = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \hat{\mathbf{j}}_M = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

which means that the transformation matrix from frame G to frame M is

$$\mathbf{R}_{MG} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

which is almost identical to the rotation matrix we met in Module 1. Rather than rotating the point through $\theta$ degrees, it rotates the coordinate system through $\theta$ degrees and expresses the point in terms of this new coordinate system. If $\theta = \pi/4$, the point $(4,3)_G$ is expressed as

$$\begin{aligned} \mathbf{r}_M &= \begin{bmatrix} 1/\sqrt{2} & 1\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \\ \Rightarrow \mathbf{r}_M &= \begin{bmatrix} 7/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix} \end{aligned}$$

or equivalently $(7/\sqrt{2}, -1/\sqrt{2})_M$.

What if we have the coordinates of a point in frame M, and we wish to express them in frame G? Referring to the transformation matrix we developed earlier, we can express this using the matrix inverse

$$\mathbf{r}_G = \mathbf{R}_{MG}^{-1}\mathbf{r}_M$$

Since this inverse must be the transformation that takes points from frame M to frame G it must be true that

$$\mathbf{R}_{GM} = \mathbf{R}_{MG}^{-1}$$

Transforming from frame M to frame G corresponds to a clockwise rotation of $\theta$ so that $\mathbf{R}_{GM}$ must be the transpose of $\mathbf{R}_{MG}$,

$$\mathbf{R}_{GM} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

For example, consider the point $(1,2)_M$ in frame M. The coordinates of this point in frame G must be

$$\begin{aligned} \mathbf{r}_G &= \begin{bmatrix} 1/\sqrt{2} & -1\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\ \Rightarrow \mathbf{r}_G &= \begin{bmatrix} -1/\sqrt{2} \\ 3/\sqrt{2} \end{bmatrix} \end{aligned}$$

or simply $(-1/\sqrt{2}, 3/\sqrt{2})_G$.

### Exercise 27.1

The frame M is a counterclockwise rotation of the global frame G by $\pi/3$ radians.

1. Draw the basis vectors for frame G and frame M.
2. Plot the frame G coordinates $(2, -1)_G$. Now express the frame G coordinates $(2, -1)_G$ in the frame M, and confirm that this is the same point by plotting it using the frame M. $P_M = (0.134, -2.232)$
3. Plot the frame M coordinates $(3, -2)_M$. Now express the frame M coordinates $(3, -2)_M$ in the frame G, and confirm that this is the same point by plotting it using the frame G.

$P_G = (3.232, 1.598)$

### 27.1.2 Coordinate Systems with a Different Origin

In addition to defining new basis vectors, we often encounter situations in which we use a new origin. In Figure 27.4 we define a global frame with basis vectors $\hat{\mathbf{i}}_G$ and $\hat{\mathbf{j}}_G$, and origin $O_G$. We also define a frame M with basis vectors $\hat{\mathbf{i}}_M$ and $\hat{\mathbf{j}}_M$, and origin $O_M$. How do we transform points from frame G to frame M, and vice versa?

Fortunately, we already met the concept of translating points in Module 1, and here we will utilize these ideas to translate the origin before rotating the basis vectors. Recall from Module 1 that in order to translate a point we can use the translation matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where $t_x$ and $t_y$ are the components of the translation, and the translation matrix acts on the vector

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For example, let's consider the case when the origin of frame M is located at $(a, b)_G$. The coordinates of this point in frame M must be, by definition, $(0, 0)_M$. The components of the translation must therefore be $-a$ and $-b$ and the translation matrix that moves the origin of frame G to frame M is then

$$\mathbf{T}_{MG} = \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix}$$

and the translation matrix that moves the origin of frame M to frame G

$$\mathbf{T}_{GM} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$$

In order to be consistent, we should adapt our rotation matrix so that it acts on a vector with 1 in the third slot

$$\mathbf{R}_{MG} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{R}_{GM} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

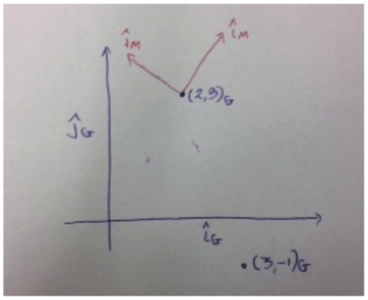


Figure 27.4: The frame M has an origin at $(2,3)_G$, and the basis vectors are rotated by an angle of $\theta$. The coordinates of the point $(3,-1)_G$ can be expressed in terms of the new frame M.

We are now ready to transform a point from frame G to frame M, by first translating the origin of frame G to the origin of frame M, and then rotating the basis vectors from frame G to frame M. The position vector of an arbitrary point is then

$$\mathbf{r}_M = \mathbf{R}_{MG}\mathbf{T}_{MG}\mathbf{r}_G$$

We can, if we choose, combine the translation with the rotation into a general transformation, but we don't have to, and there are some advantages to keeping the distinction clear.

Transforming back from frame M to frame G would be accomplished with

$$\begin{aligned} \mathbf{r}_G &= (\mathbf{R}_{MG}\mathbf{T}_{MG})^{-1}\mathbf{r}_M \\ \Rightarrow \mathbf{r}_G &= \mathbf{T}_{MG}^{-1}\mathbf{R}_{MG}^{-1}\mathbf{r}_M \end{aligned}$$

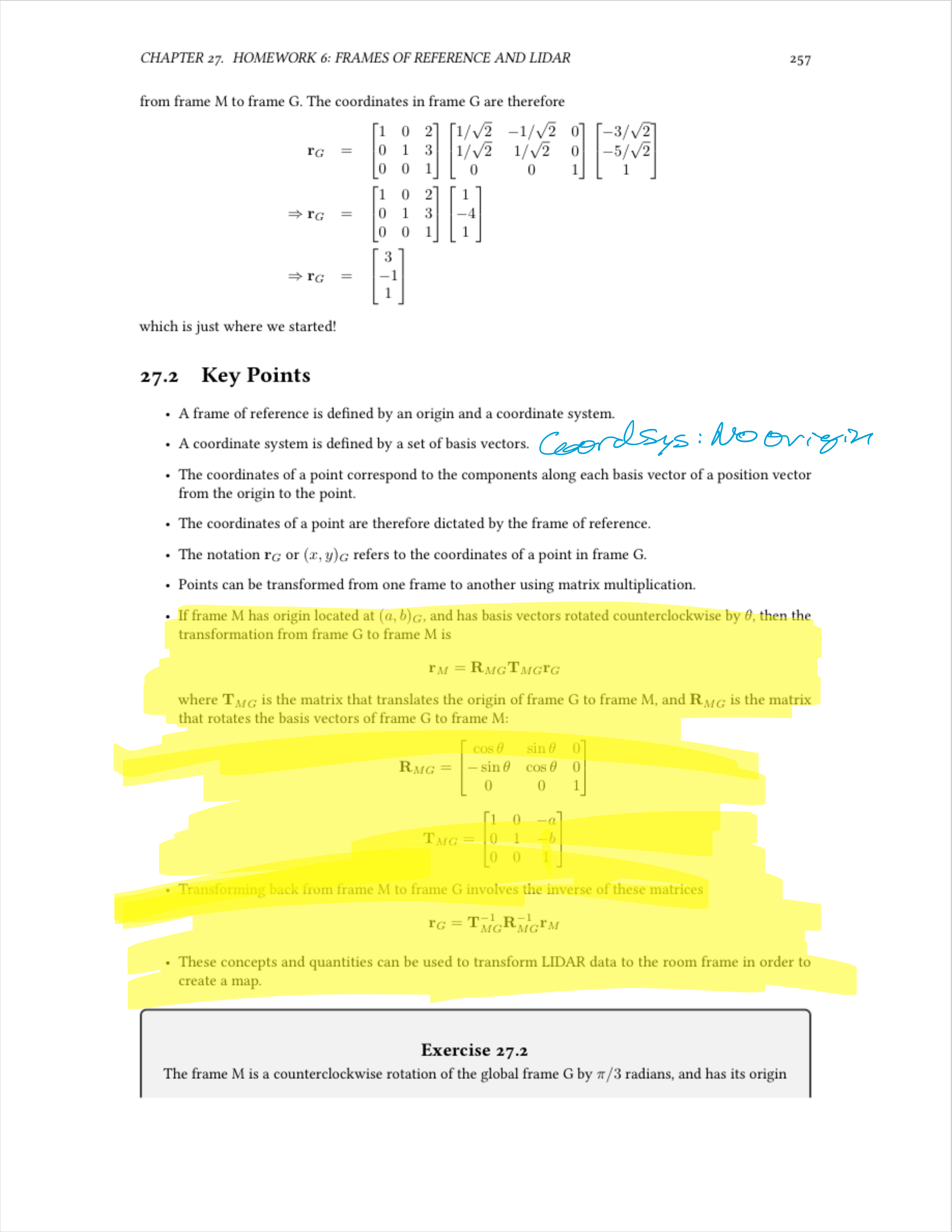
We've already seen that the inverse of the rotation matrix is just the transpose of the original and thus $(\mathbf{R}_{MG})^{-1} = \mathbf{R}_{GM}$. This makes sense. To rotate back from frame M to frame G we use the $\mathbf{R}_{GM}$. Furthermore, the inverse of the translation matrix $\mathbf{T}_{MG}$ is just the translation matrix $\mathbf{T}_{GM}$. Notice, however, that we first apply the inverse rotation and then the inverse translation,

$$\mathbf{r}_G = \mathbf{T}_{GM}\mathbf{R}_{GM}\mathbf{r}_M$$

For example, let's express the point $(3,-1)_G$ in frame M, which has its origin at $(2,3)_G$, with basis vectors rotated counterclockwise by $\pi/4$. The coordinates in frame M are therefore

$$\begin{aligned} \mathbf{r}_M &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix} \\ \Rightarrow \mathbf{r}_M &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -4 \\ 1 \end{bmatrix} \\ \Rightarrow \mathbf{r}_M &= \begin{bmatrix} -3/\sqrt{2} \\ -5/\sqrt{2} \\ 1 \end{bmatrix} \end{aligned}$$

which means the coordinates are $(-3/\sqrt{2}, -5/\sqrt{2})_M$. Let's check to see if we can transform this point back



from frame M to frame G. The coordinates in frame G are therefore

$$\begin{aligned}
\mathbf{r}_G &= \begin{bmatrix}1 & 0 & 2\\0 & 1 & 3\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1/\sqrt{2} & -1/\sqrt{2} & 0\\1/\sqrt{2} & 1/\sqrt{2} & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}-3/\sqrt{2}\\-5/\sqrt{2}\\1\end{bmatrix}\\
\Rightarrow \mathbf{r}_G &= \begin{bmatrix}1 & 0 & 2\\0 & 1 & 3\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\-4\\1\end{bmatrix}\\
\Rightarrow \mathbf{r}_G &= \begin{bmatrix}3\\-1\\1\end{bmatrix}
\end{aligned}$$

which is just where we started!

## 27.2 Key Points

- A frame of reference is defined by an origin and a coordinate system.
- A coordinate system is defined by a set of basis vectors. Coordsys: No origin
- The coordinates of a point correspond to the components along each basis vector of a position vector from the origin to the point.
- The coordinates of a point are therefore dictated by the frame of reference.
- The notation $\mathbf{r}_G$ or $(x, y)_G$ refers to the coordinates of a point in frame G.
- Points can be transformed from one frame to another using matrix multiplication.
- If frame M has origin located at $(a, b)_G$, and has basis vectors rotated counterclockwise by $\theta$, then the transformation from frame G to frame M is

$$\mathbf{r}_M = \mathbf{R}_{MG}\mathbf{T}_{MG}\mathbf{r}_G$$

where $\mathbf{T}_{MG}$ is the matrix that translates the origin of frame G to frame M, and $\mathbf{R}_{MG}$ is the matrix that rotates the basis vectors of frame G to frame M:

$$\mathbf{R}_{MG} = \begin{bmatrix}\cos\theta & \sin\theta & 0\\-\sin\theta & \cos\theta & 0\\0 & 0 & 1\end{bmatrix}$$

$$\mathbf{T}_{MG} = \begin{bmatrix}1 & 0 & -a\\0 & 1 & -b\\0 & 0 & 1\end{bmatrix}$$

- Transforming back from frame M to frame G involves the inverse of these matrices

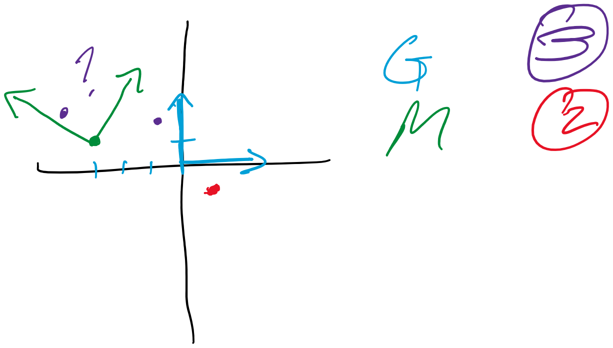
$$\mathbf{r}_G = \mathbf{T}_{MG}^{-1}\mathbf{R}_{MG}^{-1}\mathbf{r}_M$$

- These concepts and quantities can be used to transform LIDAR data to the room frame in order to create a map.

**Exercise 27.2**

The frame M is a counterclockwise rotation of the global frame G by $\pi/3$ radians, and has its origin at $(-3, 1)_G$.

1. Draw the origin and basis vectors for frame G and frame M.
2. Plot the frame G coordinates $(2, -1)_G$. Now express the frame G coordinates $(2, -1)_G$ in the frame M, and confirm that this is the same point by plotting it using the frame M.
3. Plot the frame M coordinates $(3, -2)_M$. Now express the frame M coordinates $(3, -2)_M$ in the frame G, and confirm that this is the same point by plotting it using the frame G.



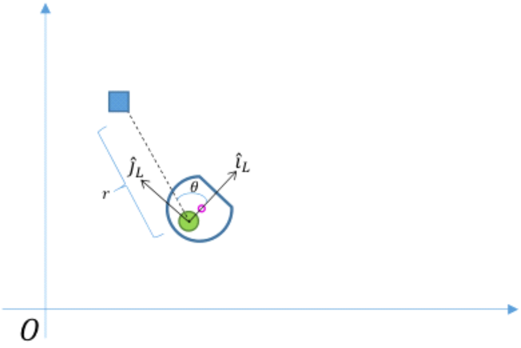


Figure 27.5: Illustration of NEATO with different coordinate systems with origin at the center of rotation of the LIDAR, and in the origin of a fixed frame of reference (e.g. origin of the room).

## 27.3 Application to the NEATO

The LIDAR reading provides a range and angle with respect to the center of rotation of the LIDAR sensor on the NEATO, with the angle measured relative to the front of the NEATO as illustrated by $\theta$ in Figure 27.5. The square object is located at a distance $r$ and angle $\theta$ as measured by the LIDAR on the Neato.

For further reference, consider Figure 27.6 which indicates the location of the LIDAR relative to the center of rotation of the Neato. The center of rotation of the Neato is indicated by the magenta circle and is the origin of two orthogonal unit vectors $\hat{\mathbf{i}}_N$ and $\hat{\mathbf{j}}_N$ (thicker, light blue arrows). For the physical Neato, you could measure the distance $\mathrm{d}$ between the origin of the reference frame based on the LIDAR and the origin of the reference frame based on the Neato's center of rotation. For the simulated Neato, the origin of the frame based on the LIDAR is located at position $-0.084m\,\hat{\mathbf{i}}_N + 0m\,\hat{\mathbf{j}}_N$ relative to the origin of the Neato coordinate system.

The orientation of the Neato relative to the absolute horizontal axis of the room is indicated by the angle $\phi$. Depending on your application, you may wish to express the position of the object (the box, in this case) in terms of a coordinate system relative to the center of rotation of the LIDAR sensor with unit vectors $\hat{\mathbf{i}}_L$ and $\hat{\mathbf{j}}_L$, center of rotation of the Neato with the unit vectors $\hat{\mathbf{i}}_N$ and $\hat{\mathbf{j}}_N$, or the global frame of reference indicated by the origin marked "O" and the blue arrows for which the unit vectors are $\hat{\mathbf{i}}_G$ and $\hat{\mathbf{j}}_G$.

**Exercise 27.3**

1. Suppose that the LIDAR returns a value of $(r, \theta)$ when scanning an object. With reference to Figure 27.5, please express the location of the object with respect to the LIDAR frame L.
2. With reference to Figures 27.5 and 27.6, please express the location of the object with respect to the NEATO frame N
3. Please express the location of the square object in the global frame G. Assume that the center of rotation of the NEATO is located at $(x_N, y_N)_G$

1) $(r\cos\theta, r\sin\theta)_L$

2) $r_N = T_{NL} R_{NL} \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 1 \end{bmatrix}$ $R_{NL} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $T_{NL} = \begin{bmatrix} 1, & 0, & 0.084 \\ 0, & 1, & 0 \\ 0, & 0, & 1 \end{bmatrix}$

3) $R_G = T_{GN} R_{GN} r_N$ $R_{GN} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$T_{GN} = \begin{bmatrix} 1 & 0 & -x_N \\ 0 & 1 & -y_N \\ 0 & 0 & 1 \end{bmatrix}$

$R_G = T_{GN} R_{GN} T_{NL} R_{NL} \begin{bmatrix} r\cos\theta \end{bmatrix}$

$0 \quad 0 \quad 1$

$$R_G = T_{GN} R_{GN} T_{NL} R_{NL} \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 1 \end{bmatrix}$$

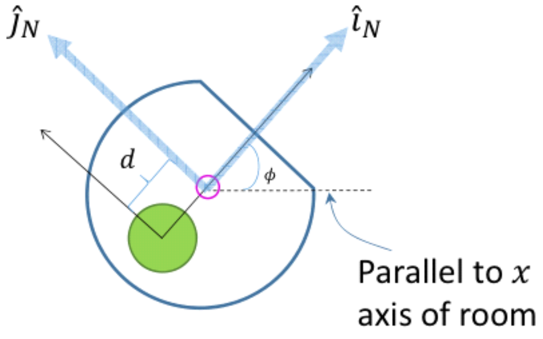


Figure 27.6: Illustration of NEATO with origin at center of rotation.

### Exercise 27.4

**Advice for this exercise:**

- Before doing any coding, make sure you have a good sense of the chain of transformations needed to express the points in the global frame (this was the focus of the preceding exercise).
- Write out some pseudocode for how you are going to enact these transformations.
- Consider writing your code using a proper MATLAB script (.m). You may want to create a function that takes as input scan data and information about the location where the scan came from and returns the points in the global frame.
- Create intermediate visualizations (e.g., scatter plots) as you move from the polar representation in the LIDAR frame all the way to the global frame. Scrutinize these intermediate plots to make sure they make sense.

Now, we will use these techniques to take LIDAR data and build a map with respect to a fixed co-ordinate frame. In the simulator, we have defined an origin, $x$, and $y$ axes, as well as placed objects on the floor at fixed locations in the Gauntlet. **Your job is to build a map of the Gauntlet when the NEATO is placed at different positions and orientations.** The map will be built using the global coordinate frame and unit vectors in the $\hat{\mathbf{i}}_G$ and $\hat{\mathbf{j}}_G$ directions. Recall that the LIDAR data is provided to you in polar co-ordinates using the coordinate frame with the origin at the center of the LIDAR sensor and the basis vectors pointing in the forward direction of the LIDAR, and $90^0$ counter-clockwise from it.
To start the Gauntlet world, use the following Matlab command.

```
>> qeasim start gauntlet_final
```

1. Read through the parts (3)-(7) below. You will be producing a total of **5 plots** for this exercise.
2. For the sake of logistics, you can collect all of your data (four different placements of the Neato) first, and then construct the plots. To do this, edit collectScans.m by modifying the coordinates and angles in the placeNeato.m function calls. **Note: both of these scripts are already in the simulated environment, so you do not need to download them.**

3. By default the NEATO starts at the origin of the global frame facing in the direction of the $\hat{\mathbf{i}}_G$ axis. Note that if you needed to reposition the Neato at this location, you can run the MATLAB command `placeNeato(0,0,1,0)`. In this position and orientation, collect data from the LIDAR. Express the LIDAR data in the fixed co-ordinate frame. Plot the data in MATLAB using the fixed reference frame and compare it to the locations of the objects in the Gauntlet.

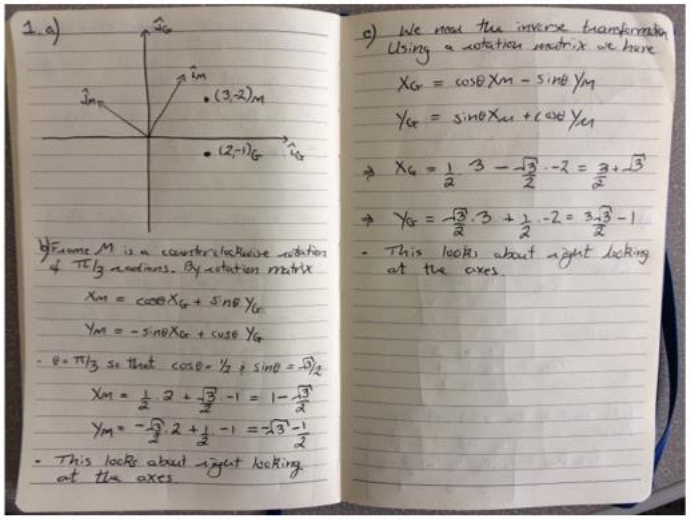
4. With the NEATO at the origin, rotate it through some angle $\phi$ (you can just pick your favourite angle here) and repeat what you did for the previous part (i.e., collect LIDAR data, translate to the fixed co-ordinate frame, plot in MATLAB, compare to the locations of the physical objects in the Gauntlet). In order to rotate the Neato by an angle $\phi$ you can run the MATLAB command `placeNeato(0, 0,cos(phi),sin(phi))`.

5. Now, move the NEATO to a different location (pick your favourite location in the Gauntlet), but keep it pointing in the same direction as the $\hat{\mathbf{i}}_G$ axis, and repeat what you did for the previous part. If you want to move the Neato to position $(a, b)_G$ but still have it $\hat{\mathbf{i}}_N$ oriented along $\hat{\mathbf{i}}_G$, you can run the MATLAB command `placeNeato(a,b,1,0)`. Make sure your Neato doesn't crash into anything when you move it. Note that it is a bit difficult to see where the axes of the Gauntlet coordinate system are located, but the grid shown in the visualizer is $1m$ by $1m$.

6. Now, move the NEATO to a different location and pointing in some arbitrary direction, and repeat what you did for the previous part.

7. Plot the locations of all four scans in the fixed (global reference frame) on the same plot (this is a great way to see if your code is correct). If all went well, the scans should mesh together well.

## Solution 27.1


1. a)
$\hat{j}_G$
$\hat{i}_M$
$\hat{j}_M$
$(3,-2)_M$
$(2,-1)_G$
$\hat{i}_G$

b) Frame M is a counterclockwise rotation of $\pi/3$ radians. By rotation matrix

$$X_M = \cos\theta X_G + \sin\theta Y_G$$

$$Y_M = -\sin\theta X_G + \cos\theta Y_G$$

- $\theta = \pi/3$ so that $\cos\theta = 1/2$ & $\sin\theta = \sqrt{3}/2$

$$X_M = \frac{1}{2} 2 + \frac{\sqrt{3}}{2} \cdot -1 = 1 - \frac{\sqrt{3}}{2}$$

$$Y_M = -\frac{\sqrt{3}}{2} \cdot 2 + \frac{1}{2} \cdot -1 = -\sqrt{3} - \frac{1}{2}$$

- This looks about right looking at the axes

c) We need the inverse transformation Using a rotation matrix we have

$$X_G = \cos\theta X_M - \sin\theta Y_M$$

$$Y_G = \sin\theta X_M + \cos\theta Y_M$$

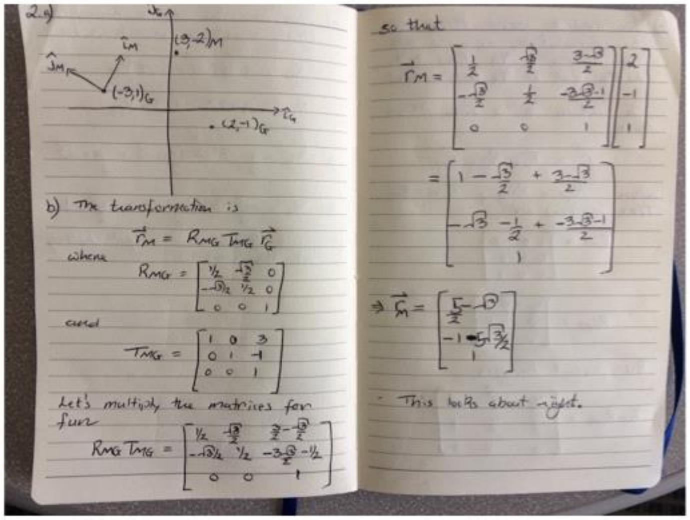
$$\Rightarrow X_G = \frac{1}{2} 3 - \frac{\sqrt{3}}{2} \cdot -2 = \frac{3}{2} + \sqrt{3}$$

$$\Rightarrow Y_G = \frac{\sqrt{3}}{2} \cdot 3 + \frac{1}{2} \cdot -2 = \frac{3\sqrt{3}}{2} - 1$$

- This looks about right looking at the axes.

## Solution 27.2

1. Parts 1 and 2:


2. a)
$\hat{j}_G$
$\hat{i}_M$
$\hat{j}_M$
$(3,-2)_M$
$(-3,1)_G$
$(2,-1)_G$
$\hat{i}_G$

b) The transformation is

$$\vec{r}_M = R_{MG} T_{MG} \vec{r}_G$$

where

$$R_{MG} = \begin{bmatrix} 1/2 & \frac{\sqrt{3}}{2} & 0 \\ -\sqrt{3}/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$T_{MG} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Let's multiply the matrices for fun

$$R_{MG} T_{MG} = \begin{bmatrix} 1/2 & \frac{\sqrt{3}}{2} & \frac{3}{2} - \frac{\sqrt{3}}{2} \\ -\sqrt{3}/2 & 1/2 & -3\frac{\sqrt{3}}{2} - 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

so that

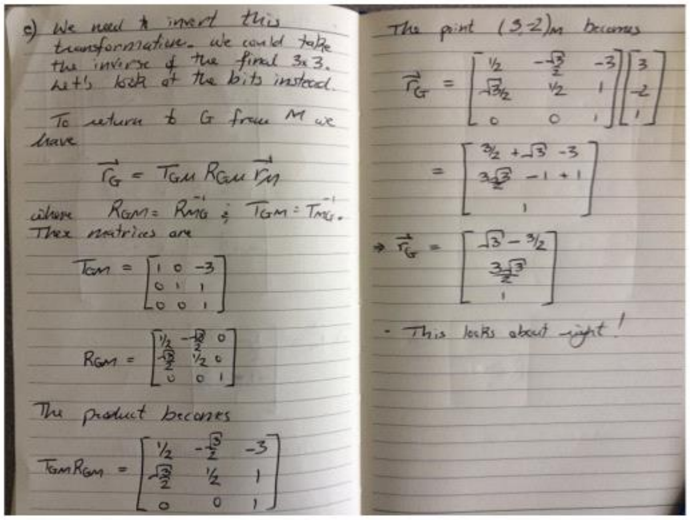
$$\vec{r}_M = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{3-\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{-3\sqrt{3}-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - \frac{\sqrt{3}}{2} + \frac{3-\sqrt{3}}{2} \\ -\sqrt{3} - \frac{1}{2} + \frac{-3\sqrt{3}-1}{2} \\ 1 \end{bmatrix}$$

$$\Rightarrow \vec{r}_M = \begin{bmatrix} \frac{5}{2} - \sqrt{3} \\ -1 - 5\sqrt{3}/2 \\ 1 \end{bmatrix}$$

- This looks about right.

2. Part 3:



c) We need to invert this transformation. We could take the inverse of the final 3x3. Let's look at the bits instead.

To return to G from M we have

$$\vec{r}_G = T_{GM} R_{GM} \vec{r}_M$$

where $R_{GM} = R_{MG}^{-1}$ ; $T_{GM} = T_{MG}^{-1}$. These matrices are

$$T_{GM} = \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{GM} = \begin{bmatrix} 1/2 & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The product becomes

$$T_{GM} R_{GM} = \begin{bmatrix} 1/2 & -\frac{\sqrt{3}}{2} & -3 \\ \frac{\sqrt{3}}{2} & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

The point $(3,-2)_M$ becomes

$$\vec{r}_G = \begin{bmatrix} 1/2 & -\frac{\sqrt{3}}{2} & -3 \\ \sqrt{3}/2 & 1/2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix}$$

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

$$\Rightarrow \vec{r}_G = \begin{bmatrix} \sqrt{3} - 3/2 \\ \frac{3\sqrt{3}}{2} \\ 1 \end{bmatrix}$$

- This looks about right!

## Solution 27.3

1. We will denote the location of the object in the LIDAR frame L as $\mathbf{r}_L$. If we measure the polar coordinates of the object then its location is $\mathbf{r}_L = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}$.
2. We will denote the location of the object in the NEATO frame as $\mathbf{r}_N$. It is a translation with respect to the LIDAR origin. The location of the object is now $\mathbf{r}_N = \begin{bmatrix} r\cos\theta - d \\ r\sin\theta \end{bmatrix}$.
3. The NEATO is translated and rotated with respect to the room. The transformation matrices are

$$\mathbf{T}_{GN} = \begin{bmatrix} 1 & 0 & x_N \\ 0 & 1 & y_N \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{GN} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying these out and transforming the coordinates of the object in the NEATO frame leads to

$$\mathbf{r}_G = \begin{bmatrix} r\cos(\theta+\phi) - d\cos\phi + x_N \\ r\sin(\theta+\phi) - d\sin\phi + y_N \end{bmatrix}$$

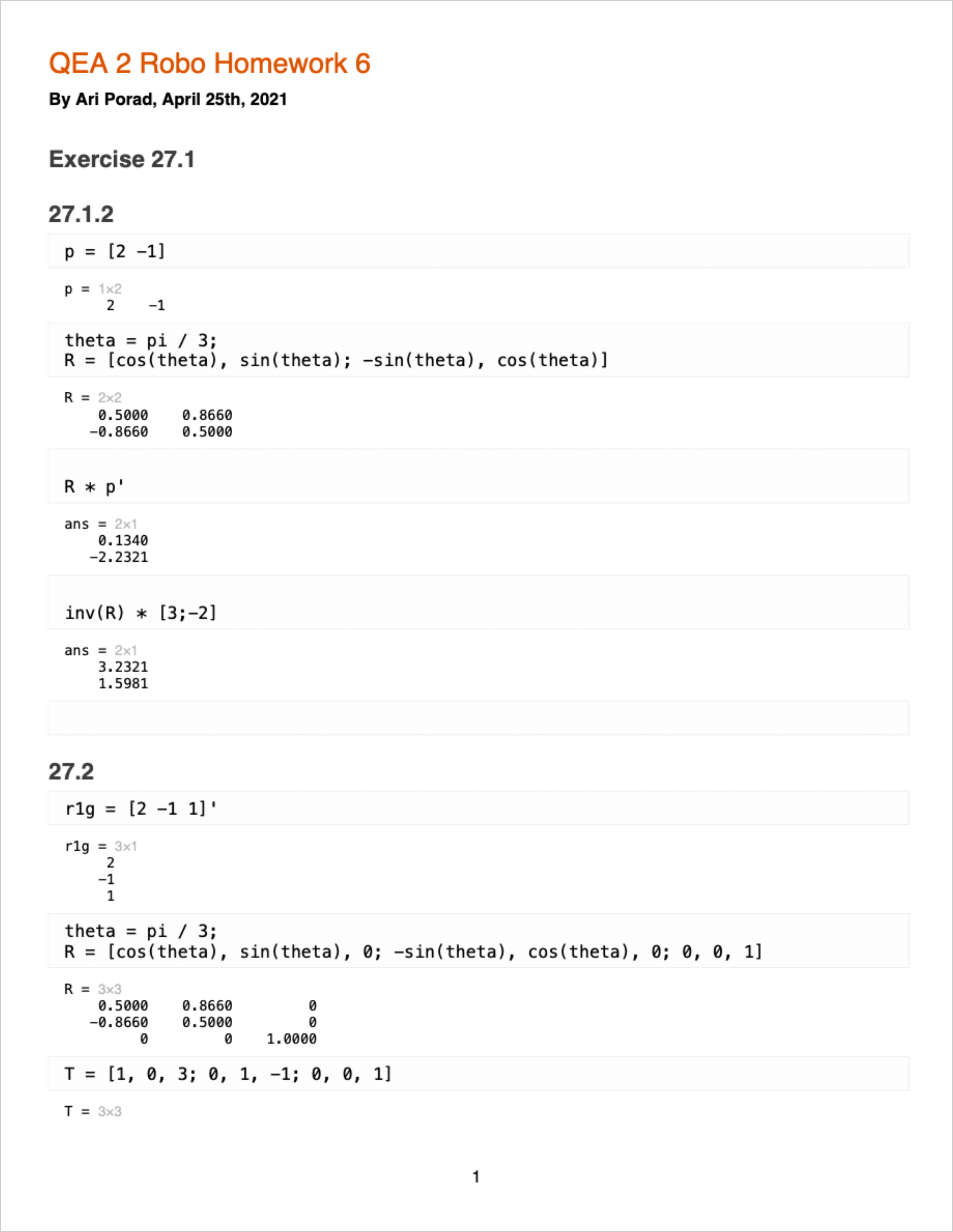
where we have used a trig formula.

## Solution 27.4

Our solutions are available in the following two MATLAB files.

- collectScansSolution.m: after connecting to the simulator, run this function to position the robot and collect the scan data. The data is stored in a `.mat` file for subsequent process.
- makeGauntletMapSolution.m: after running `collectScansSolution`, you can run this function transform each scan to the global frame and plot them on a single plot.

hw6



# QEA 2 Robo Homework 6

**By Ari Porad, April 25th, 2021**

## Exercise 27.1

### 27.1.2

```
p = [2 -1]
```

```
p = 1x2
     2    -1
```

```
theta = pi / 3;
R = [cos(theta), sin(theta); -sin(theta), cos(theta)]
```

```
R = 2x2
    0.5000    0.8660
   -0.8660    0.5000
```

```
R * p'
```

```
ans = 2x1
    0.1340
   -2.2321
```

```
inv(R) * [3;-2]
```

```
ans = 2x1
    3.2321
    1.5981
```

### 27.2

```
r1g = [2 -1 1]'
```

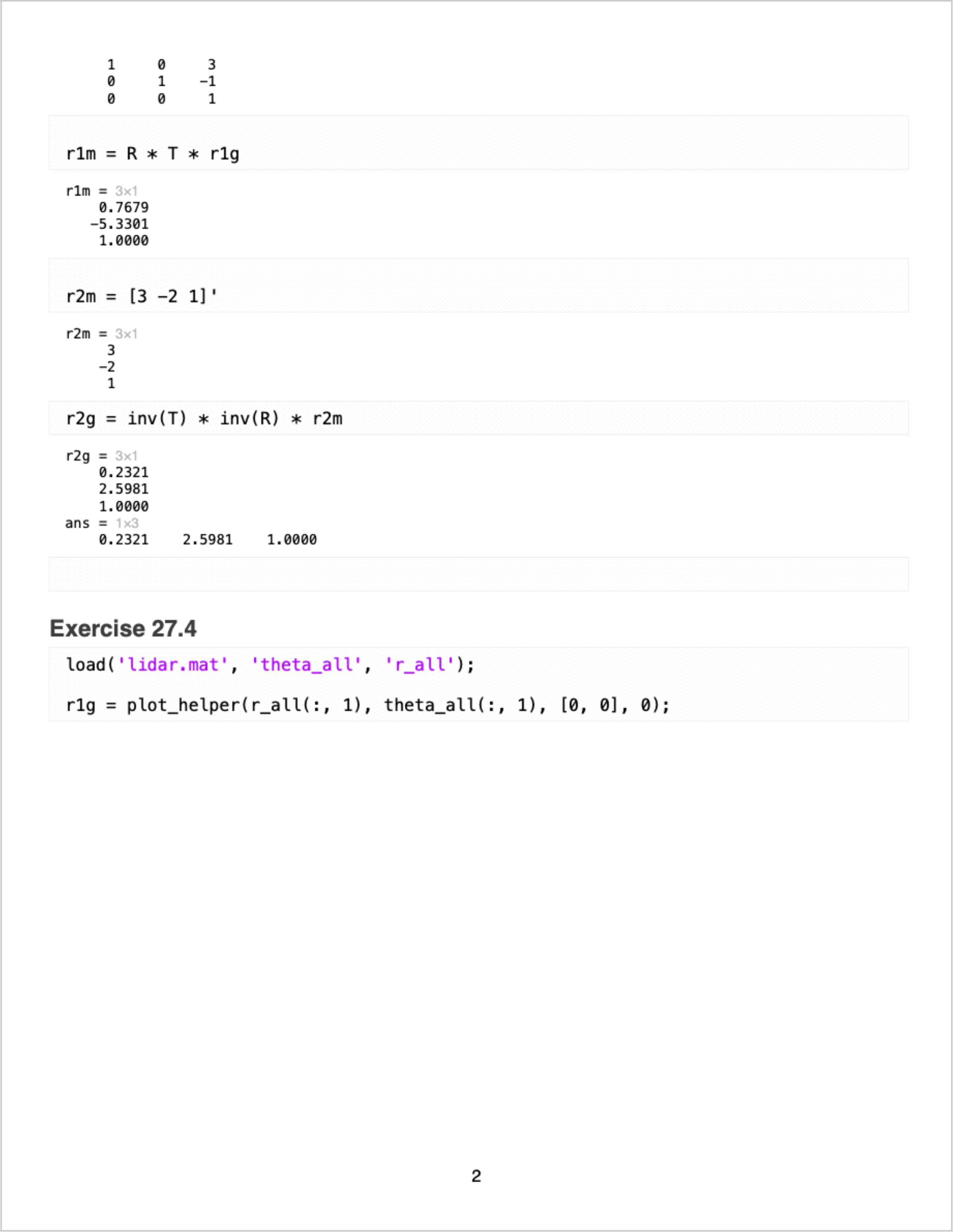
```
r1g = 3x1
     2
    -1
     1
```

```
theta = pi / 3;
R = [cos(theta), sin(theta), 0; -sin(theta), cos(theta), 0; 0, 0, 1]
```

```
R = 3x3
    0.5000    0.8660         0
   -0.8660    0.5000         0
         0         0    1.0000
```

```
T = [1, 0, 3; 0, 1, -1; 0, 0, 1]
```

```
T = 3x3
```



```
     1     0     3
     0     1    -1
     0     0     1
```

```
r1m = R * T * r1g
```

```
r1m = 3×1
    0.7679
   -5.3301
    1.0000
```

```
r2m = [3 -2 1]'
```

```
r2m = 3×1
     3
    -2
     1
```

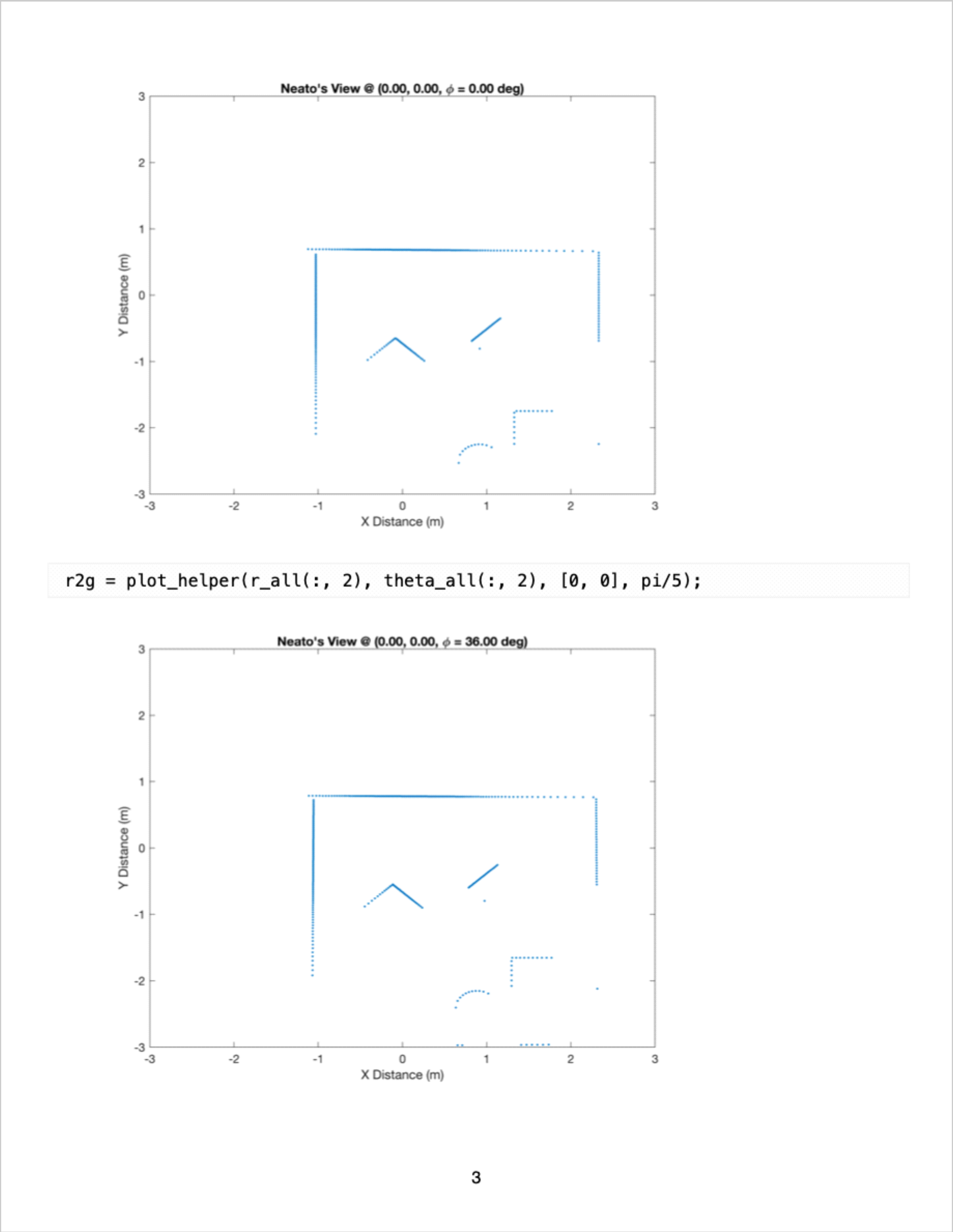
```
r2g = inv(T) * inv(R) * r2m
```

```
r2g = 3×1
    0.2321
    2.5981
    1.0000
ans = 1×3
    0.2321    2.5981    1.0000
```

### Exercise 27.4

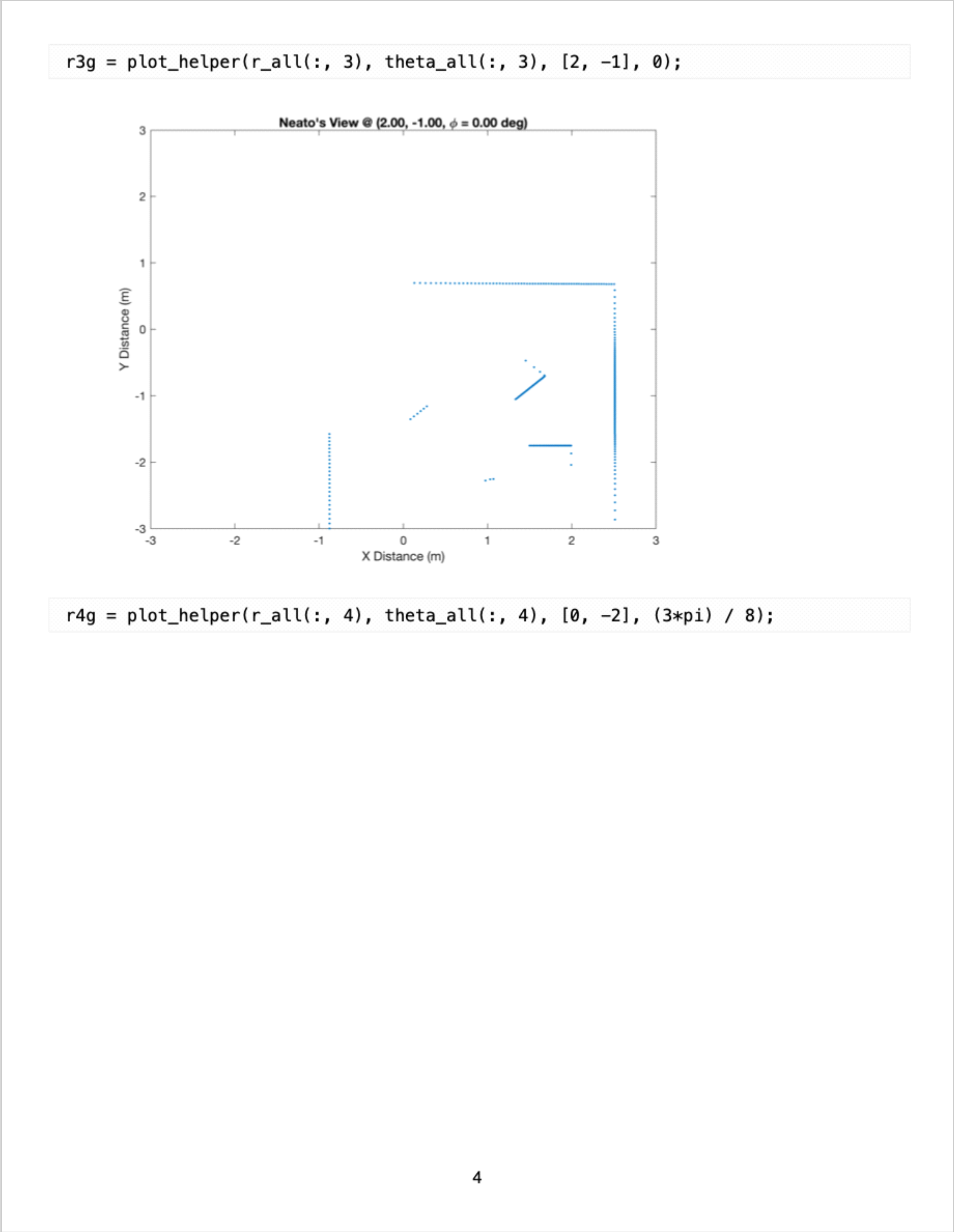
```
load('lidar.mat', 'theta_all', 'r_all');

r1g = plot_helper(r_all(:, 1), theta_all(:, 1), [0, 0], 0);
```



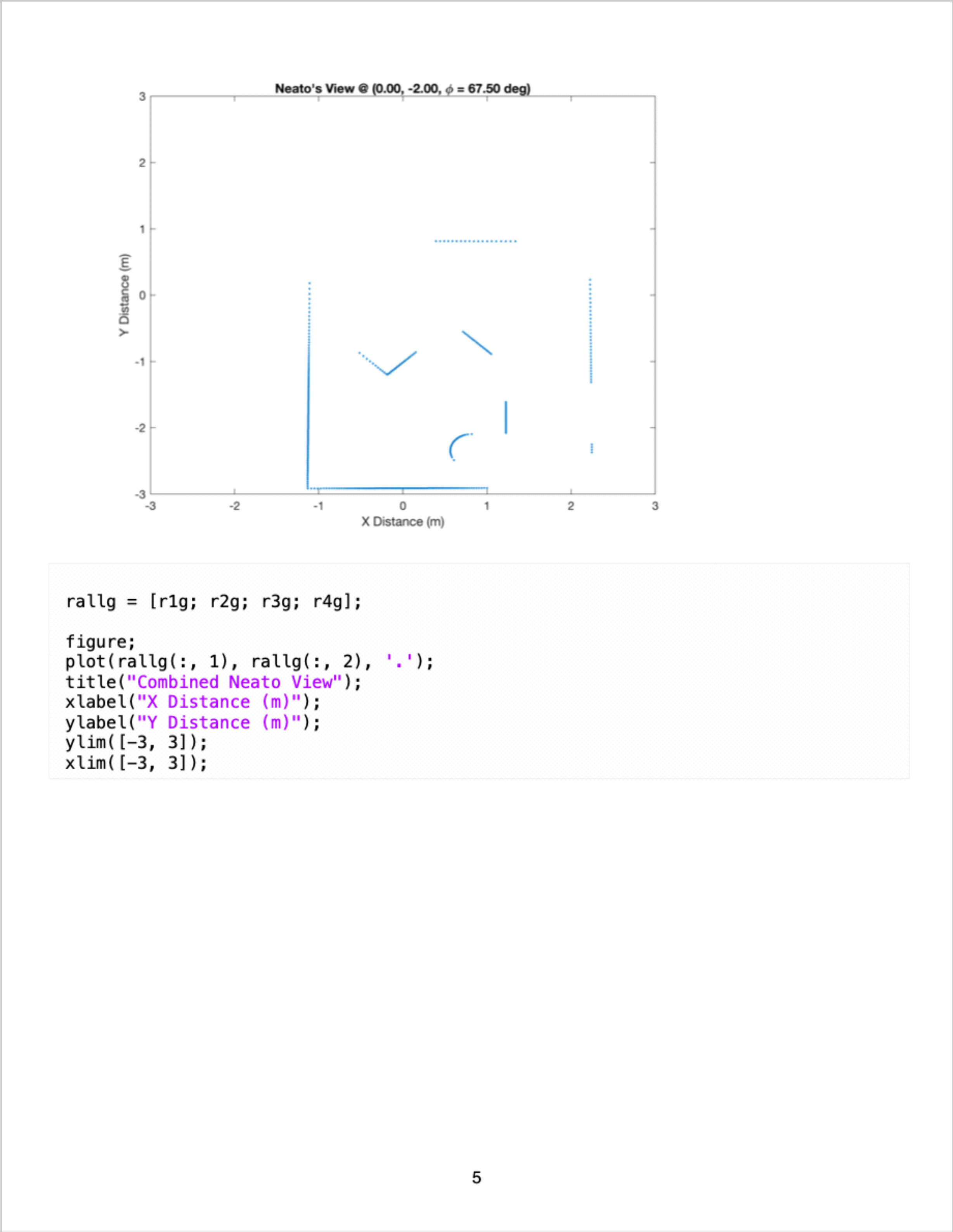
```
r2g = plot_helper(r_all(:, 2), theta_all(:, 2), [0, 0], pi/5);
```

3



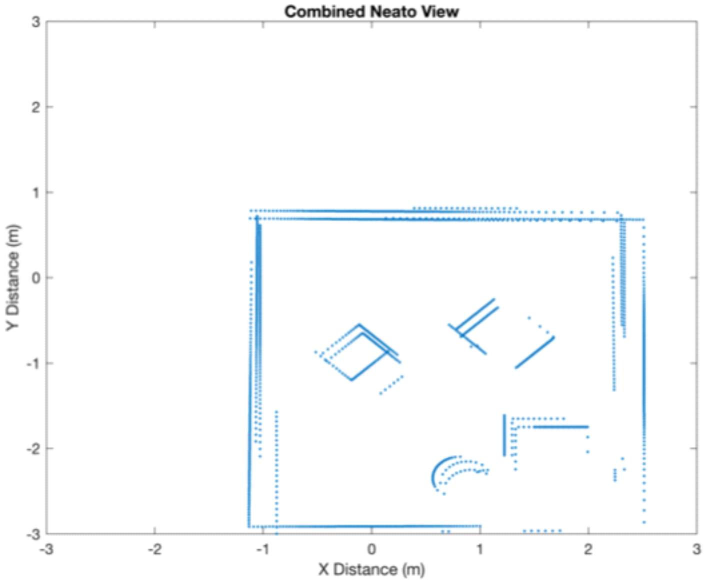
```
r3g = plot_helper(r_all(:, 3), theta_all(:, 3), [2, -1], 0);
```

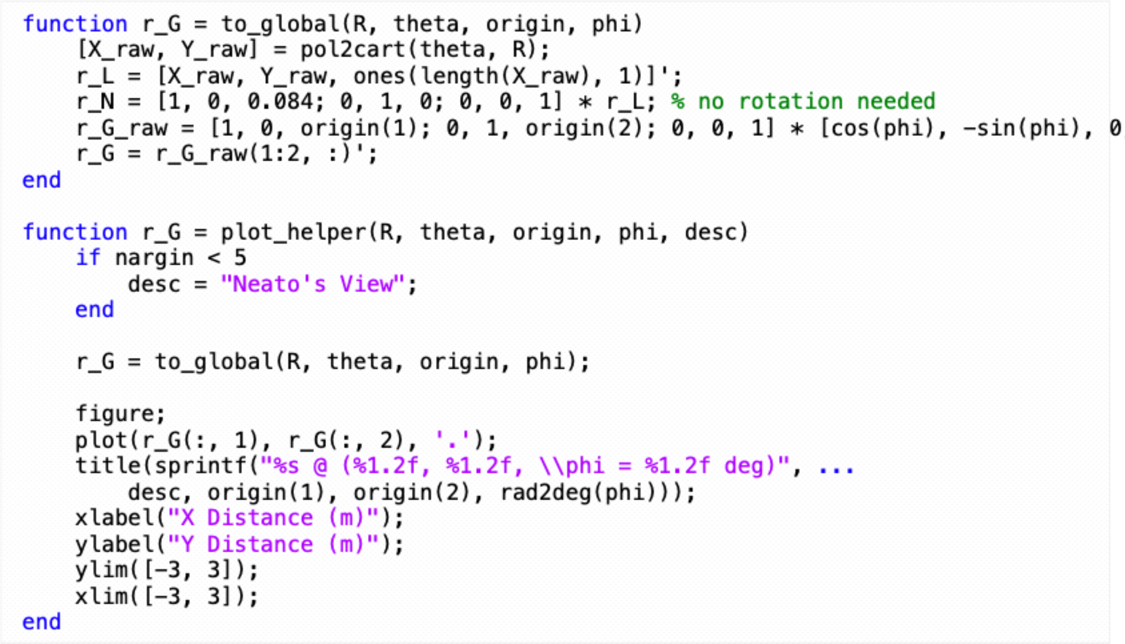
```
r4g = plot_helper(r_all(:, 4), theta_all(:, 4), [0, -2], (3*pi) / 8);
```



```
rallg = [r1g; r2g; r3g; r4g];

figure;
plot(rallg(:, 1), rallg(:, 2), '.');
title("Combined Neato View");
xlabel("X Distance (m)");
ylabel("Y Distance (m)");
ylim([-3, 3]);
xlim([-3, 3]);
```





```
function r_G = to_global(R, theta, origin, phi)
    [X_raw, Y_raw] = pol2cart(theta, R);
    r_L = [X_raw, Y_raw, ones(length(X_raw), 1)]';
    r_N = [1, 0, 0.084; 0, 1, 0; 0, 0, 1] * r_L; % no rotation needed
    r_G_raw = [1, 0, origin(1); 0, 1, origin(2); 0, 0, 1] * [cos(phi), -sin(phi), 0; s
    r_G = r_G_raw(1:2, :)';
end

function r_G = plot_helper(R, theta, origin, phi, desc)
    if nargin < 5
        desc = "Neato's View";
    end

    r_G = to_global(R, theta, origin, phi);

    figure;
    plot(r_G(:, 1), r_G(:, 2), '.');
    title(sprintf("%s @ (%1.2f, %1.2f, \\phi = %1.2f deg)", ...
        desc, origin(1), origin(2), rad2deg(phi)));
    xlabel("X Distance (m)");
    ylabel("Y Distance (m)");
    ylim([-3, 3]);
    xlim([-3, 3]);
end
```