SDM5008 Fall 2024 · Advanced Control for Robotics

Homework 1

董骏博 / 12432995

September 25, 2024

Liner Algebra

(a) What are the dimensions of the null space and column space (i.e. range space) of A?
Answer: The dimension of the null space of A is equal to $dim(A) - rank(A)$, and the dimension of the column space of A is rank(A). That is, the number of independent column vectors of A
Since rank(A) is 2 and dim(A) is 1, the **dimension of the null space is 1** and the **column vector space is 2**.

(b) Find a set of basis vectors for null(A).
Answer: Since the dimension of the null space of this matrix A is 1, the basis vectors of null(A) can be $\begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$.

The basis vectors of col(A) are the linearly independent vectors in matrix A, i.e. $\begin{bmatrix} 1 & -1 \\ 1 & 2 \\ -1 & 0 \\ 0 & 1 \\ 1 & 2 \end{bmatrix}$.

(c) Find a set of basis vectors for col(A).
Answer: In terms of dimensions, the rank of both matrix A and matrix C is 2. Therefore, the dimensions of Col(A) and Col(C) are the same. It can be determined that the basis vectors of Col(C) can be $\begin{bmatrix} -2 & -1 \\ 1 & 5 \\ 1 & -1 \\ 1 & 2 \\ 1 & 5 \end{bmatrix}$.

(d) Is $col(C) = col(A)$? Justify your answer.
Answer: If $Col(A) = Col(B)$, then it can be said Col(A) and Col(B)are linearly dependence. We can assume a matrix $Sample = [Col(A), Col(B)] = \begin{bmatrix} 1 & -1 & -2 & -1 \\ 1 & 2 & 1 & 5 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 2 \\ 1 & 2 & 1 & 5 \end{bmatrix}$. If the dimension of the null space of the matrix is equal to 2 or the dimension of the Column space is equal to 2, then it means $Col(A) = Col(B)$, otherwise $Col(A) \neq Col(B)$. Solving the rank of the matrix Sample, we find that its rank is equal to 2, so the dimension of the null space of the matrix Sample is 2, which can be proved $Col(A) = Col(B)$。

(e) Find a matrix B of appropriate dimension such that $C = AB$. (You should be able to find B just by inspection).
Answer: Because for $C = AB$, the C matrix is a linear combination of the column vectors of the A matrix. Since$c_1 = -a_1 + a_2, \quad c_2 = a_1 + 2a_2, \quad c_3 = 2a_1 + a_2, \quad c_4 = a_1 + a_2$, we can coonclude that the B matrix is $\begin{bmatrix} -1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

Speak the Matrix Language

(a) For each i, row i of Z is a linear combination of rows $i, ..., n$ of Y .
Answer: Every row of I is a Linear combination of the columns of Y^T. It can be expressed as "$I(i) = Y^T F$, for some matrix F".

(b) W is obtained from V by permuting adjacent odd and even columns (i.e., 1 and 2, 3 and 4,...).
Answer:

$$W = VF, F = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} (V \in \mathbb{R}^{m*2n}) \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & \\ 1 & 0 & 0 & 0 & \dots & 0 & \\ 0 & 0 & 0 & 1 & \dots & 0 & \\ 0 & 0 & 1 & 0 & \dots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & \end{bmatrix} (V \in \mathbb{R}^{m*(2n+1)})$$

Or this can be written as : $W = VP$ where P is a permutation matrix that swaps adjacent odd and even columns.

(c) Each column of P makes an acute angle with the corresponding column of Q.
Answer: Due to the property of an acute angle, the angle between the two matrices satisfies $0 < \theta < \frac{\pi}{2}, so 0 < \cos(\theta) < 1$. So we can get $P^T Q > 0$.

(d) The first k columns of A are orthogonal to the remaining columns of A.
Answer: Asssume A_1 irepresents the first k columns of A, and A_2 represents the remaining columns. We then have $A_1^T A_2 = 0$

Matrix Rank

(a) Let $a \in \mathbf{R}^n$ be an n-dim vector. Show that the $n \times n$ matrix $A \triangleq aa^T$ is of rank 1.
Answer:

$$A = aa^T = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

Matrix A can be seen as a linear combination of the column vectors of a, with coefficients given by $a^T x$. Therefore, all columns of matrix A are linearly dependent, and the rank of A is 1.

(b) Given two nonzero square matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$, argue that if $AB = 0$, then neither A nor B can be full rank.
Answer: If matrix A is full rank, the null space of A is trivial, i.e., it only contains the zero vector. For matrix A, in order for $Ax = 0$, $x$must be the zero vector. However, since matrix B is nonzero, it is impossible for $AB = 0$. Therefore, the statement "if $AB = 0$, then neither A nor B can be full rank" does not hold. The proof for matrix B follows similarly to that of A.

(c) Explain why the system $Ax = b$ has a solution if and only if rank(A) = rank ([Ab]).
Answer: When rank(A) = rank([Ab]), it indicates that the vector b is linearly dependent on the columns of matrix A, meaning that b can be expressed as a linear combination of the columns of A. Since Ax is a linear combination of the columns of A, there exists a solution x such that $Ax = b$.

Ellipsoids

(a) **Answer:** Given that substituting (A, b) into $E_2(A, x_c)$ directly yields $b = x_c$. And using the representation $E_2(A, x_c)$ as $x = x_c + Au : \|u\|^2 \leq 1$, we have$(x - x_c) = Au$. Substituting$Au$ into $E_1(P, x_c)$, we can get $(Au)^T P^{-1} Au \leq 1$. By eliminating u^2 on the both sides, we arrive at $A^T P^{-1} A = I$, which implies $A = P^{-\frac{1}{2}}$.

(b) The hand-drawn figure is shown below:

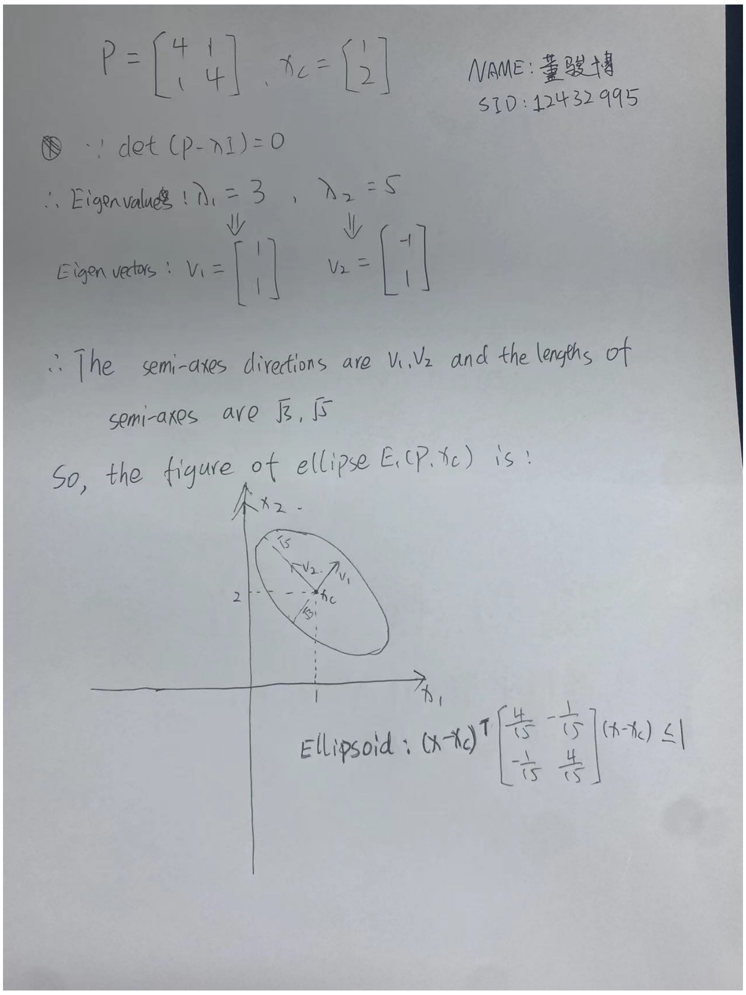


Figure 1: figure of ellipse $E_1(P, x_c)$

(c) The python-drawn figure is shown below:

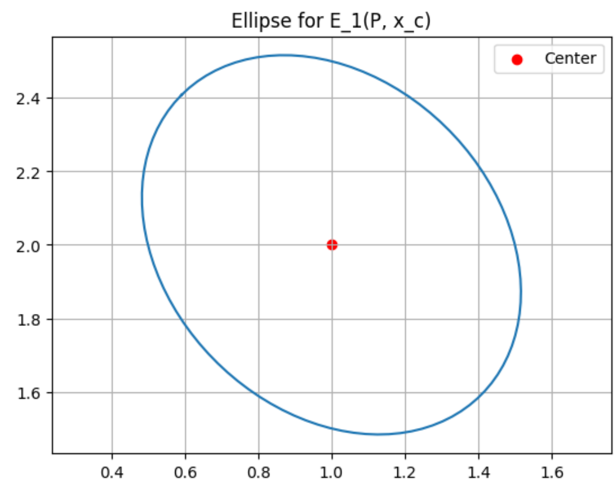


Figure 2: figure of ellipse $E_1(P, x_c)$

Polyhedron

(a) **Answer:** We can combine these two polyhedra into a new polyhedron P, expressed as follows $P = P_1 \cap P_2 = \{x \in \mathbb{R}^n : Ax \leq b\}$ where

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

(b) **Answer:** The question of whether P_1 intersects with the halfspace $a^T x \leq 3$ can be viewed as a linear programming problem. Given $A_1 x \leq b_1$ and $a^T x \leq 3$ we can derive the following.

$$\begin{cases} x_2 \leq 7 \\ 5x_1 - 2x_2 \leq 36 \\ -x_1 - 2x_2 \leq -14 \\ -4x_1 - 2x_2 \leq -26 \end{cases} \quad \begin{cases} x_1 + x_2 \leq 3 \end{cases}$$

Using linear programming in Python, we can solve this problem, and the result indicates that P_1 does not intersect with the halfspace $a^T x \leq 3$.

HOMEWORK1

Python Basics

(a) Write a program to display the current date and time.

In [1]:
```
from datetime import datetime
# get time
current_time = datetime.now()
#print it
print("Current date and time: ", current_time)
```

```
Current date and time:  2024-09-19 14:43:49.533915
```

(b) Write a program to print a specified list after removing the 0th, 4th and 5th elements.

In [13]:
```
# create a list
SampleList = ['one', 'two', 'three', 'four', 'five', 'six']
# remove the 0th, 4th and 5th elements
RemoveIndex = [0, 4, 5]
for item in sorted(RemoveIndex, reverse=True):
    SampleList.pop(item)
# print the modified list
print(SampleList)
```

```
['two', 'three', 'four']
```

(c) Define a class called Student that includes the student’s name and age information.

In addition, you should provide a method to display these information.

In [18]:
```
# define class of student
class student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
# print student name and age
```

```
    def StudentInfo(self):
        print("student name:",self.name,"\nstudent age:",self.ag
# test
st1 = student('junbo',22)
st1.StudentInfo()
```

```
student name: junbo
student age: 22
```

Linear Algebra

In this class, it is important to use Python to complete the linear algebra task. Let's get familiar with it now.

(a) Print the two matrices A and B.

$$A = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} B = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} b = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

In [20]:

```
# import numpy
import numpy as np
# define matrix
A = np.array([[1, -2, 4], [1, -1, 1], [1, 0, 0], [1, 1, 1]])
print(A)
B = np.array([[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]])
print(B)
```

```
[[ 1 -2  4]
 [ 1 -1  1]
 [ 1  0  0]
 [ 1  1  1]]
[[1 2 3]
 [1 2 3]
 [1 2 3]
 [1 2 3]]
```

(b) Print the second row of A and the third column of B

In [35]:

```
print("the second row of A:", A[1, ])
print("the third column of B:", B[1:,2])
```

```
the second row of A: [ 1 -1  1]
the third column of B: [3 3 3]
```

(c) Print the results of A + B and A − B.

In [37]:
```python
print("the results of A + B:\n",A+B)
print("the results of A - B:\n",A-B)
```

```
the results of A + B:
 [[2 0 7]
 [2 1 4]
 [2 2 3]
 [2 3 4]]
the results of A - B:
 [[ 0 -4  1]
 [ 0 -3 -2]
 [ 0 -2 -3]
 [ 0 -1 -2]]
```

(d) Construct a new 4 x 6 matrix [A, B] by appending B to the right of matrix A.

In [40]:
```python
# define a NewMatrix by np.hstack
NewMatrix = np.hstack((A, B))
# print it
print(NewMatrix)
```

```
[[ 1 -2  4  1  2  3]
 [ 1 -1  1  1  2  3]
 [ 1  0  0  1  2  3]
 [ 1  1  1  1  2  3]]
```

(d) Compute $A^T B$

In [41]:
```python
# use the python @ operator as matrix multiplication
print(A.T @ B)
```

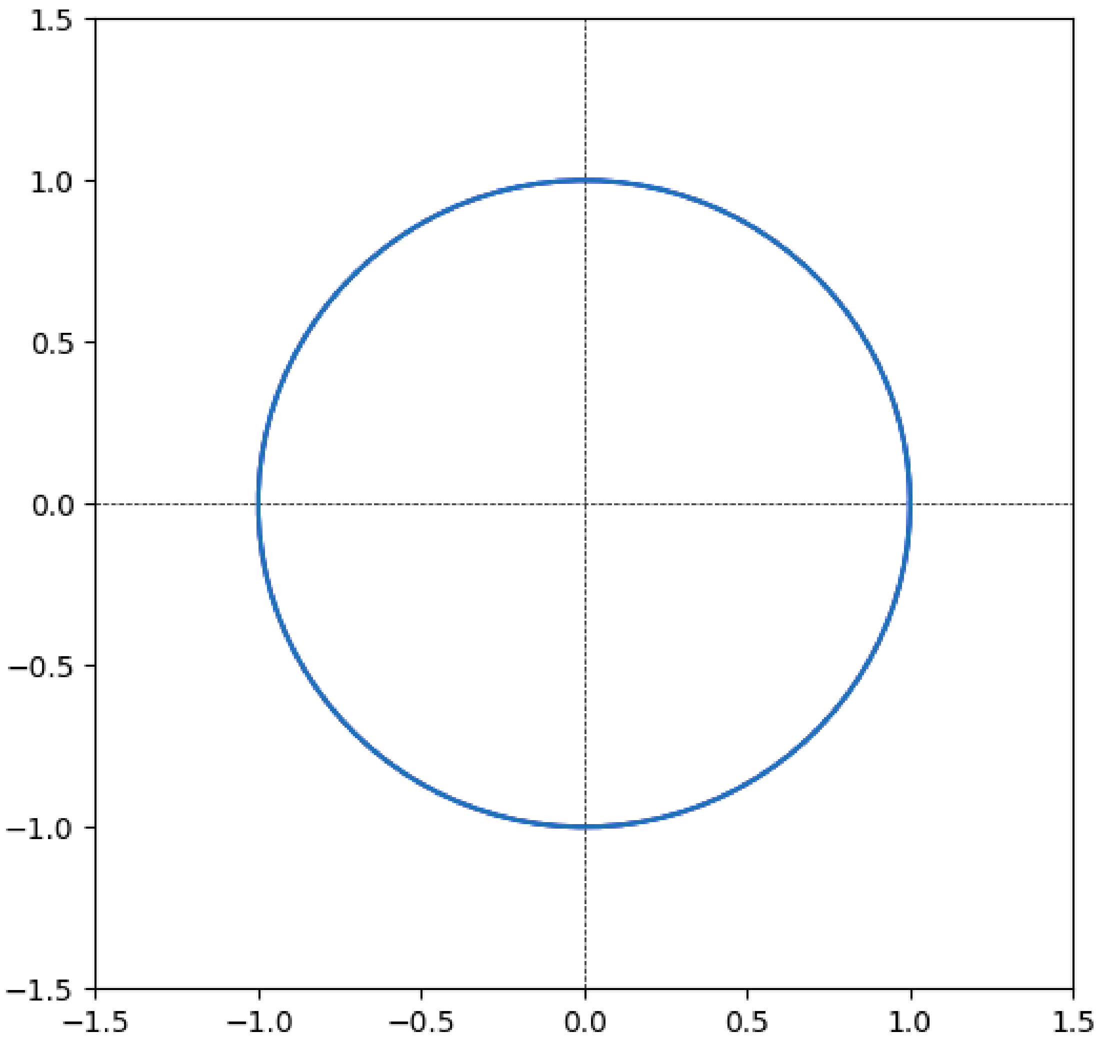
```
[[ 4  8 12]
 [-2 -4 -6]
 [ 6 12 18]]
```

3. Matplotlib

(a) Plot a unit circle

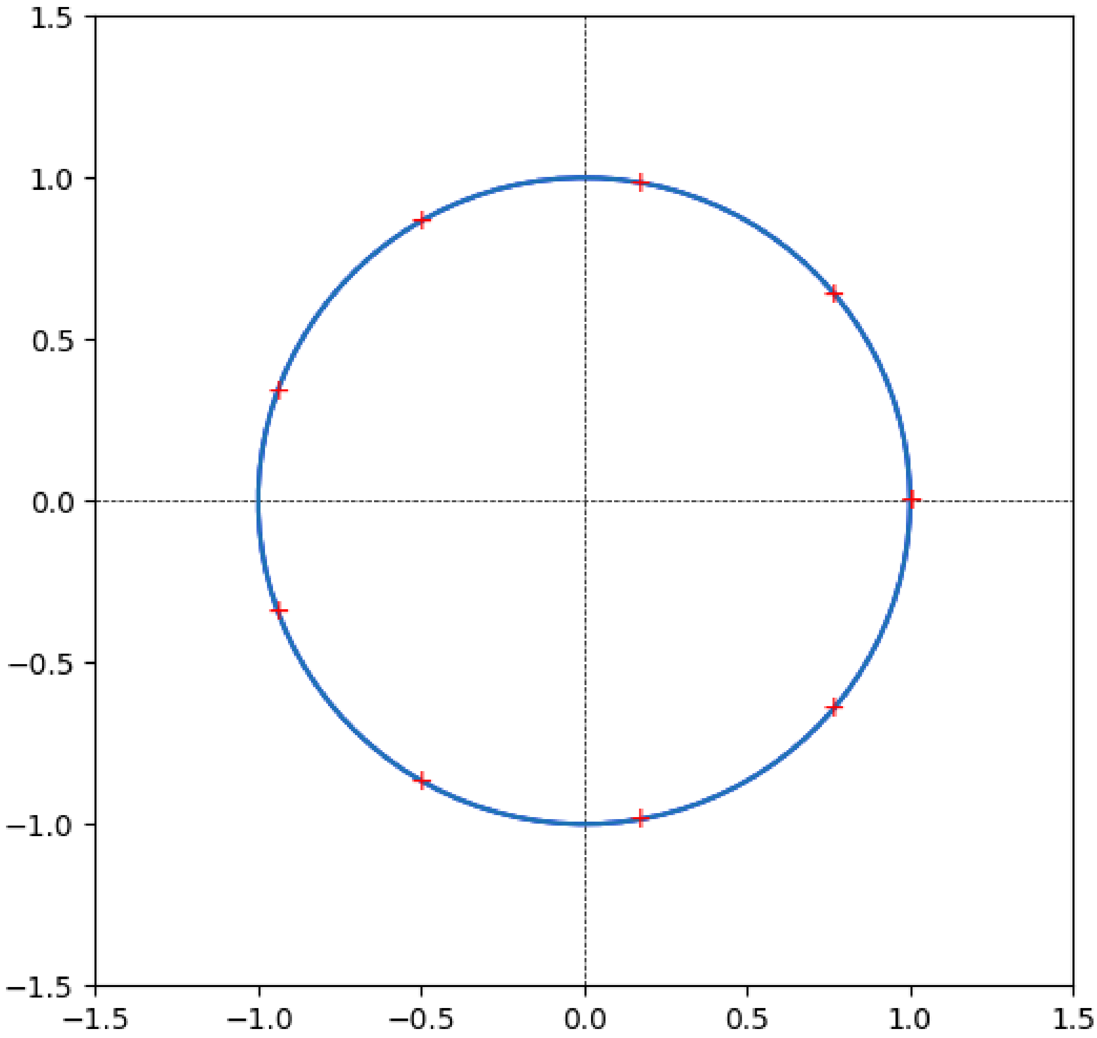
In [85]:
```
import matplotlib.pyplot as plt
import numpy as np

# generate theta values from 0 to 2π
theta = np.linspace(0, 2 * np.pi, 100)

# cos^2 + sin^2 = 1 the cricle equation
x = np.cos(theta)
y = np.sin(theta)

# set the plot to beauty the figure
plt.figure(figsize=(6, 6))
plt.plot(x, y, label='Unit Circle', color='blue')
plt.xlim(-1.5, 1.5)
plt.ylim(-1.5, 1.5)
plt.axhline(0, color='black',linewidth=0.5, ls='--')
plt.axvline(0, color='black',linewidth=0.5, ls='--')

# plot the cricle
plt.plot(x,y)
```

Out[85]:
```
[<matplotlib.lines.Line2D at 0x25db5639d30>]
```



(b) Plot 10 plus signs "+" uniformly distributed on the unit circle.

In [95]:
```
import matplotlib.pyplot as plt
import numpy as np

# generate theta values from 0 to 2π
theta = np.linspace(0, 2 * np.pi, 100)

# cos^2 + sin^2 = 1 the cricle equation
x = np.cos(theta)
y = np.sin(theta)

# set the plot to beauty the figure
plt.figure(figsize=(6, 6))
plt.plot(x, y, label='Unit Circle', color='blue')
plt.xlim(-1.5, 1.5)
plt.ylim(-1.5, 1.5)
plt.axhline(0, color='black',linewidth=0.5, ls='--')
plt.axvline(0, color='black',linewidth=0.5, ls='--')

# plot the cricle
plt.plot(x,y)

# Plot plus signs at each point
theta1 = np.linspace(0, 2 * np.pi, 10)

# cos^2 + sin^2 = 1 the cricle equation
x = np.cos(theta1)
y = np.sin(theta1)
for i in range(10):
    plt.text(x[i], y[i], '+', ha = 'center', va = 'center', colo
```



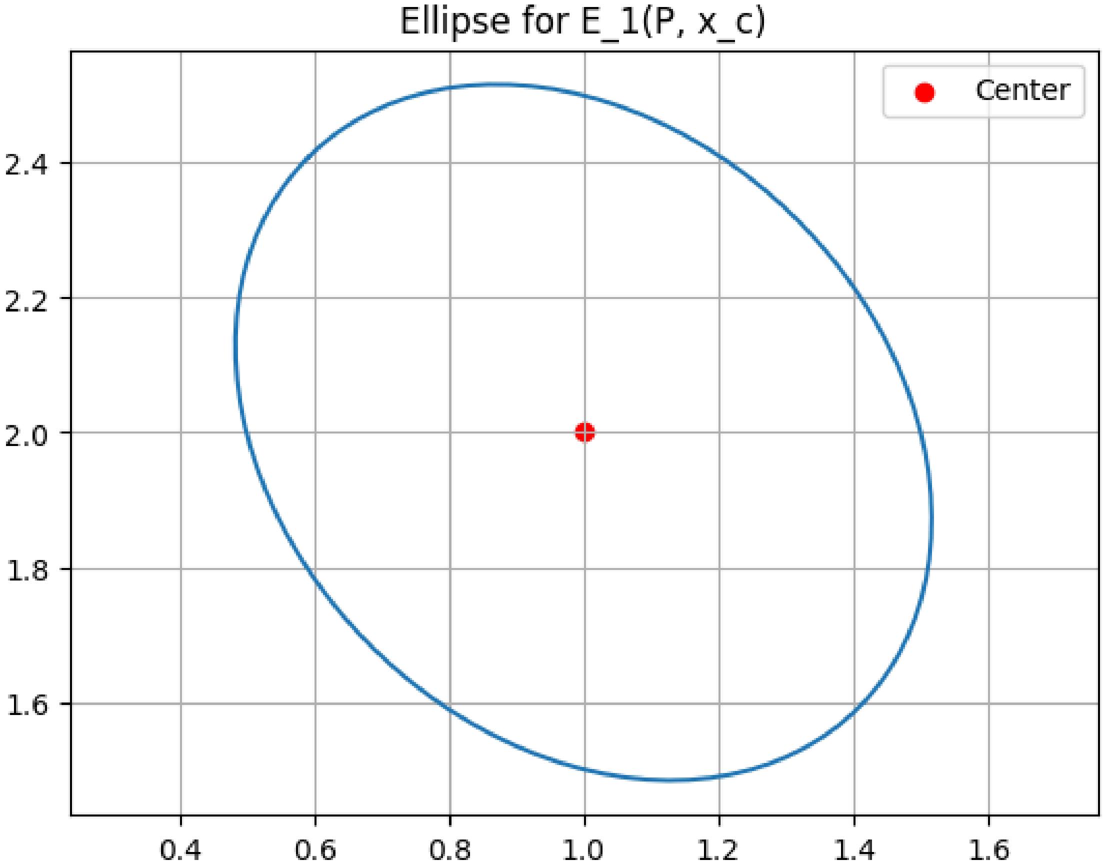
7.(c) Draw the ellipse in part (b) using Python

In [1]:

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

# define P,xc
P = np.array([[4, 1], [1, 4]])
xc = np.array([1, 2])

# eigenvalue & eigenvector
eigvals, eigvecs = np.linalg.eigh(P)
print(eigvals,eigvecs)
axes_lengths = 1 / np.sqrt(eigvals)
theta = np.linspace(0, 2 * np.pi, 100)
ellipse = np.array([axes_lengths[0] * np.cos(theta), axes_length
rotation_matrix = eigvecs
rotated_ellipse = rotation_matrix @ ellipse
ellipse_x = rotated_ellipse[0, :] + xc[0]
ellipse_y = rotated_ellipse[1, :] + xc[1]
```

```
# plot
plt.plot(ellipse_x, ellipse_y)
plt.scatter(xc[0], xc[1], color='red', label='Center')
plt.axis('equal')
plt.title('Ellipse for E_1(P, x_c)')
plt.grid(True)
plt.legend()
plt.show()
```

```
[3. 5.] [[-0.70710678  0.70710678]
 [ 0.70710678  0.70710678]]
```



8.(2) Check whether P_1 intersects with the halfspace $a^T x \leq 3$ using Python or by hand

In [4]:
```
import numpy as np
from scipy.optimize import linprog

# define A1,b1,a^T
A1 = np.array([[0, 1], [5, -2], [-1, -2], [-4, -2]])
b1 = np.array([7, 36, -14, -26])
c = np.array([1, 1])

# solute
```

```python
res = linprog(c, A_ub=A1, b_ub=b1, method='highs')

# print info
if res.success:
    print(f"Optimal solution{res.x}")
    print(f"The minimum value of a^T x: {res.fun}")
else:
    print("No Solution")
if res.fun <= 3:
    print("intersect")
else:
    print("no intersect")
```

```
Optimal solution[4. 5.]
The minimum value of a^T x: 9.0
no intersect
```