CSC398: Introduction to Autonomous Robots
Linear algebra for robotics

Ubbo Visser

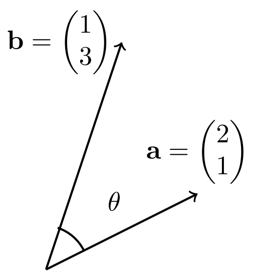
August 22, 2024

This document serves as an introductory guide to the application of linear algebra in the area of robotics, particularly focusing on essential mathematical concepts such as vectors, dot products, matrix multiplication, determinants, and eigenvalues. These concepts are foundational for understanding the complex dynamics and control systems in autonomous robotic systems.

1. Vectors and Dot Product

Vectors and the dot product are essential concepts in robotics for understanding direction, magnitude, and interactions between forces, velocities, and other quantities.

- Vectors represent quantities that have both magnitude and direction. In robotics, vectors can represent positions, velocities, forces, and more. For example, the movement of a robot in space can be expressed as a vector in 2D or 3D space.
- Dot Product is an operation that takes two vectors and **returns a scalar**. Mathematically, it is the **product of the magnitudes of the two vectors and the cosine of the angle between them**. In robotics, the dot product is used to:
  - Measure the similarity between two vectors (if the dot product is zero, the vectors are orthogonal).
  - Project one vector onto another.
  - Compute the angle between two vectors, which is important for tasks like trajectory planning, collision avoidance, and calculating work done by a force in a specific direction.


$\mathbf{b} = \begin{pmatrix}1\\3\end{pmatrix}$
$\mathbf{a} = \begin{pmatrix}2\\1\end{pmatrix}$
θ

$$\mathbf{a} \cdot \mathbf{b} = 2 \times 1 + 1 \times 3 = 5$$

Exercise:

Compute the angle between two vectors. Given vectors $\mathbf{a} = \begin{pmatrix} 3 \\ 4 \\ -1 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}$, compute the following:

a) The dot product $\mathbf{a} \cdot \mathbf{b}$.

b) The angle between $\mathbf{a}$ and $\mathbf{b}$ (in degrees).

Solution

a) **The dot product $\mathbf{a} \cdot \mathbf{b}$**:

The formula for the dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

Substituting the components of $\mathbf{a}$ and $\mathbf{b}$:

$$\mathbf{a} \cdot \mathbf{b} = (3 \times 1) + (4 \times -2) + (-1 \times 2)$$

$$= 3 + (-8) + (-2) = 3 - 8 - 2 = -7$$

Thus, the dot product $\mathbf{a} \cdot \mathbf{b} = -7$.

b) **The angle between a and b**:

The dot product is related to the cosine of the angle θ between the vectors by the formula:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos(\theta)$$

where $|\mathbf{a}|$ and $|\mathbf{b}|$ are the magnitudes of the vectors.

First, calculate the magnitude of $\mathbf{a}$:

$$|\mathbf{a}| = \sqrt{3^2 + 4^2 + (-1)^2} = \sqrt{9 + 16 + 1} = \sqrt{26}$$

Now, calculate the magnitude of $\mathbf{b}$:

$$|\mathbf{b}| = \sqrt{1^2 + (-2)^2 + 2^2} = \sqrt{1 + 4 + 4} = \sqrt{9} = 3$$

Using the dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos(\theta)$$

Substituting known values:

$$-7 = \sqrt{26} \times 3 \times \cos(\theta)$$

$$\cos(\theta) = \frac{-7}{3\sqrt{26}} \approx \frac{-7}{15.297} \approx -0.4576$$

Finally, solve for the angle θ by taking the inverse cosine:

$$\theta \approx \arccos(-0.4576) \approx 117.26^\circ$$

Thus, the angle between the vectors is approximately 117.26°.

2. Matrix Multiplication

Matrix multiplication is a fundamental operation in robotics for transforming coordinates, rotations, and combining multiple linear transformations. We can use a robot arm for a simple example.

- **Understanding the Problem:** Imagine a robot arm with two segments. The first segment can rotate horizontally, and the second segment can rotate both horizontally and vertically. We want to determine the final position of the end effector (the tip of the arm) based on the angles of rotation for each segment.
- **Representing the Rotations:**
  - **Rotation Matrix:** A 3x3 matrix can represent a 2D rotation around the origin.
  - **Segment 1:** Rotate horizontally. We'll use a rotation matrix R1.
  - **Segment 2:** Rotate horizontally and vertically. We'll use rotation matrices R2h and R2v.
- **Calculating the Final Position:**
  - **Initial Position:** Assume the end effector starts at (1, 0) relative to the base of the arm.
  - **Segment 1 Rotation:** Multiply the initial position by R1 to get the new position after the first segment rotates.
  - **Segment 2 Horizontal Rotation:** Multiply the result from step 2 by R2h.
  - **Segment 2 Vertical Rotation:** Multiply the result from step 3 by R2v.
- **Matrix Multiplication Example:** Let's say R1 represents a 45-degree horizontal rotation and R2h represents a 30-degree horizontal rotation.
  - R1: [[cos(45), -sin(45)], [sin(45), cos(45)]] [[0.707, -0.707], [0.707, 0.707]]
  - R2h: [[cos(30), -sin(30)], [sin(30), cos(30)]] [[0.866, -0.5], [0.5, 0.866]]
  - Initial position: [[1], [0]]
  - Final position: R2v * R2h * R1 * [[1], [0]]

Note: The R2v matrix would represent a vertical rotation, but for simplicity, we'll omit it in this example.

By performing the matrix multiplications, we can calculate the final x and y coordinates of the end effector.

Matrix multiplication between two matrices A and B can be represented as follows:

Let A be a matrix of size $m \times n$, and B be a matrix of size $n \times p$. The resulting matrix C from the multiplication of A and B will be of size $m \times p$.

$$C = A \times B$$

Each element C_{ij} of the resulting matrix C is computed as:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$$

For example, if:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Then the product $C = A \times B$ would be:

$$C = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Exercise:

Given the matrices:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

a) Compute the matrix product AB.

b) Is matrix multiplication commutative? Verify by computing BA and comparing with AB.

Solution:

a) The matrix product AB is:

$$AB = \begin{pmatrix} 1 \cdot (-1) + 2 \cdot 0 & 1 \cdot 0 + 2 \cdot 1 \\ 3 \cdot (-1) + 4 \cdot 0 & 3 \cdot 0 + 4 \cdot 1 \end{pmatrix} = \begin{pmatrix} -1 & 2 \\ -3 & 4 \end{pmatrix}.$$

b) The matrix product BA is:

$$BA = \begin{pmatrix} -1 \cdot 1 + 0 \cdot 3 & -1 \cdot 2 + 0 \cdot 4 \\ 0 \cdot 1 + 1 \cdot 3 & 0 \cdot 2 + 1 \cdot 4 \end{pmatrix} = \begin{pmatrix} -1 & -2 \\ 3 & 4 \end{pmatrix}.$$

Since $AB \neq BA$, matrix multiplication is not commutative.

3. Determinants and Inverses

Given a square matrix A of size $n \times n$, the determinant of A, denoted as $\det(A)$ or $|A|$, is a scalar value that provides important properties of the matrix, such as whether the matrix is **invertible**.

For example, for a 2x2 matrix A:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

The determinant of A is calculated as:

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}$$

The determinant helps to determine whether a matrix is invertible. If $\det(A) = 0$, the matrix is singular and does not have an inverse.

The inverse of a square matrix A, denoted as A^{-1}, is the matrix that satisfies the following equation:

$$AA^{-1} = A^{-1}A = I$$

where I is the identity matrix of the same size as A.
For a 2x2 matrix A, the inverse exists if $\det(A) \neq 0$, and the inverse is given by:

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

This formula shows that the inverse of a matrix can be computed by using the determinant and modifying the original matrix.

Exercise

Given the matrix:

$$C = \begin{pmatrix} 4 & 7 \\ 2 & 6 \end{pmatrix}$$

a) Compute the determinant of matrix C.

b) If the determinant is non-zero, find the inverse of C.

Solution:

a) The determinant of C is:

$$\det(C) = (4 \cdot 6) - (7 \cdot 2) = 24 - 14 = 10.$$

b) Since the determinant is non-zero, the inverse of C is:

$$C^{-1} = \frac{1}{\det(C)} \begin{pmatrix} 6 & -7 \\ -2 & 4 \end{pmatrix} = \frac{1}{10} \begin{pmatrix} 6 & -7 \\ -2 & 4 \end{pmatrix}.$$

The inverse of a matrix is crucial in robotics due to its role in solving systems of linear equations, which are fundamental to various robotic tasks such as control, motion planning, and kinematics. Specifically:

1. Solving Linear Systems

The inverse matrix is used to solve equations of the form $A\mathbf{x} = \mathbf{b}$, where A represents the system dynamics, $\mathbf{x}$ is the unknown vector (e.g., joint angles or velocities), and $\mathbf{b}$ is a vector representing the desired state or forces. The solution for $\mathbf{x}$ can be found using:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

2. Kinematics

In inverse kinematics (IK) problems, the goal is to compute the joint angles of a robot to achieve a desired end-effector position. This often involves finding the inverse of the Jacobian matrix, which maps joint velocities to end-effector velocities. The inverse Jacobian enables the calculation of joint velocities necessary to achieve a desired end-effector velocity.

3. Control

In robotic control algorithms, such as feedback control, matrix inverses are used to calculate the necessary input forces or torques to achieve the desired behavior. For example, the inverse of a control matrix can help determine the appropriate control inputs to drive the system toward a target state. Thus, matrix inverses are essential for efficient and accurate robotic control, motion, and behavior.

4. Eigenvalues and Eigenvectors

Eigenvalues of matrices provide insights into the stability, control, and dynamic behavior of systems. In control theory, they help determine the stability of a system by analyzing the characteristics of the system's matrices, such as those representing a robot's dynamics or control laws. Eigenvalues indicate how the system will respond over time to changes in state, which is critical for tasks like motion planning, manipulation, and feedback control. **If the eigenvalues have negative real parts, the system is stable**; if positive, it may be unstable, which is vital for designing robust robotic systems.

Exercise:

Given the matrix:

$$D = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

a) Find the eigenvalues of D.

b) For each eigenvalue, find the corresponding eigenvector.

Solution:

a) To find the eigenvalues, solve the characteristic equation:

$$\det(D - \lambda I) = \det \begin{pmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{pmatrix} = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = 0.$$

Solving this quadratic equation gives the eigenvalues:

$$\lambda_1 = 3, \quad \lambda_2 = 1.$$

b) For $\lambda_1 = 3$, solve $(D - 3I)\mathbf{v} = 0$:

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \implies x_1 = x_2.$$

Thus, an eigenvector for $\lambda_1 = 3$ is $\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

For $\lambda_2 = 1$, solve $(D - I)\mathbf{v} = 0$:

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \implies x_1 = -x_2.$$

Thus, an eigenvector for $\lambda_2 = 1$ is $\mathbf{v}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

5. Linear Transformations

A linear transformation is a mapping $T : \mathbb{R}^n \to \mathbb{R}^m$ that preserves vector addition and scalar multiplication. In mathematical terms, for vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and scalar c, the transformation satisfies the following properties:

$$T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$$
$$T(c\mathbf{u}) = cT(\mathbf{u})$$

A linear transformation can be represented by a matrix A, such that for a vector $\mathbf{x} \in \mathbb{R}^n$, the transformation is defined as:

$$T(\mathbf{x}) = A\mathbf{x}$$

The matrix A encodes the effects of the transformation, such as rotations, scaling, or shear, depending on the specific entries of A.

Examples in Robotics

In robotics, linear transformations are used to:

- Transform coordinates between different reference frames (e.g., from world coordinates to robot coordinates).
- Apply rotations and translations to vectors representing positions or velocities.
- Perform scaling or shear operations in tasks such as robotic manipulation or computer vision.

For example, a 2D rotation transformation can be represented by the following matrix:

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

This matrix rotates a vector $\mathbf{x}$ by an angle θ.

In the context of linear transformations, shearing refers to a transformation that slants the shape of an object. Unlike rotation or scaling, shearing distorts the shape of objects by shifting one axis, while keeping the other axis fixed. This transformation skews the object, changing its angles but keeping its area (in 2D) or volume (in 3D) constant.

In robotics, shearing can be used to adjust or distort coordinate frames, such as when handling visual perspectives or when objects need to be reshaped geometrically within certain tasks, like image processing or sensor data interpretation.

Exercise:

Consider the linear transformation T represented by the matrix:

$$T = \begin{pmatrix} 1 & 3 \\ 2 & 5 \end{pmatrix}$$

a) Apply the transformation to the vector $\mathbf{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

b) What is the geometric interpretation of the transformation T in terms of scaling, rotation, or shearing?

Solution:

a) Apply the transformation to $\mathbf{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$:

$$T\mathbf{v} = \begin{pmatrix} 1 \cdot 1 + 3 \cdot 1 \\ 2 \cdot 1 + 5 \cdot 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 7 \end{pmatrix}.$$

b) The geometric interpretation of T is a combination of scaling and shearing.

6. Systems of Linear Equations

Solving linear equations using matrix methods, such as Gaussian elimination or matrix inversion, is important in robotics for several key reasons:

1. Kinematics and Dynamics

- **Forward and Inverse Kinematics**: Robotics often involves calculating the position and orientation of a robot's end-effector given certain joint parameters (forward kinematics) or computing the joint parameters required to reach a target position (inverse kinematics). These problems are typically expressed as systems of linear equations, and solving them efficiently ensures accurate robot positioning.
- **Dynamics**: The equations of motion in robotics, which relate forces, torques, and accelerations, are often linear or linearized and can be solved using matrix methods. Solving these linear equations is critical for modeling how a robot will move in response to control inputs.

2. Control Systems

- **Feedback Control**: In control systems, robotic controllers often involve solving systems of linear equations to compute control inputs (such as forces or torques) that will achieve the desired behavior. For example, in Linear Quadratic Regulators (LQR), the control law is derived by solving linear equations.
- **State Estimation**: Techniques such as the Kalman filter, which are used for sensor fusion and state estimation, rely on solving linear systems to combine noisy measurements into accurate state predictions.

3. Optimization and Path Planning

Many optimization problems in robotics, such as finding the shortest path or optimal trajectory, involve solving linear systems either directly or as part of iterative methods (e.g., in linear programming or least squares optimization).

4. Real-Time Computation

In robotics, many computations must be done in real-time for reactive control, sensor integration, and path adjustments. Efficient matrix methods like Gaussian elimination and matrix inversion allow robots to compute necessary transformations, adjustments, and solutions rapidly, ensuring smooth operation and responsiveness.

Thus, matrix methods provide a structured and efficient way to solve complex linear systems that are fundamental to robotics, enabling precise control, accurate motion, and real-time adaptability.

Exercise:

Solve the following system of linear equations using matrix methods (e.g., Gaussian elimination or matrix inversion):

$$\begin{aligned} 2x + 3y &= 5 \\ 4x - y &= 1 \end{aligned}$$

Solution:

Solving the System of Equations using Matrix Method.

We are given the system of linear equations:

$$\begin{aligned} 2x + 3y &= 5 \\ 4x - y &= 1 \end{aligned}$$

This can be written in matrix form as:

$$\begin{pmatrix} 2 & 3 \\ 4 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

We denote the matrix equation as:

$$A\mathbf{x} = \mathbf{b}$$

where:

$$A = \begin{pmatrix} 2 & 3 \\ 4 & -1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

The solution to the system is given by:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Step 1: Compute the Determinant of A

The determinant of A is:

$$\det(A) = (2)(-1) - (3)(4) = -2 - 12 = -14$$

Step 2: Compute the Inverse of Matrix A

The inverse of a 2×2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is given by:

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

So, for our matrix A:

$$A^{-1} = \frac{1}{-14} \begin{pmatrix} -1 & -3 \\ -4 & 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{14} & \frac{3}{14} \\ \frac{2}{7} & -\frac{1}{7} \end{pmatrix}$$

Step 3: Multiply A^{-1} by b

Now, multiply the inverse of A by the vector $\mathbf{b}$:

$$\mathbf{x} = A^{-1}\mathbf{b} = \begin{pmatrix} \frac{1}{14} & \frac{3}{14} \\ \frac{2}{7} & -\frac{1}{7} \end{pmatrix} \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Perform the matrix multiplication:

$$\mathbf{x} = \begin{pmatrix} \frac{1}{14} \times 5 + \frac{3}{14} \times 1 \\ \frac{2}{7} \times 5 + -\frac{1}{7} \times 1 \end{pmatrix}$$

Simplifying the terms:

$$\mathbf{x} = \begin{pmatrix} \frac{5}{14} + \frac{3}{14} \\ \frac{10}{7} - \frac{1}{7} \end{pmatrix} = \begin{pmatrix} \frac{8}{14} \\ \frac{9}{7} \end{pmatrix} = \begin{pmatrix} \frac{4}{7} \\ \frac{9}{7} \end{pmatrix}$$

Thus, the solution to the system of equations is:

$$x = \frac{4}{7}, \quad y = \frac{9}{7}$$

7. Rank of a Matrix

Importance of the Rank of a Matrix in Robotics

The rank of matrix provides key insights into the properties and capabilities of robotic systems, particularly with regard to their kinematics, control, and motion planning. Here are the main reasons why:

Controllability and Observability

The rank of matrices, such as the *controllability* or *observability matrices*, determines whether a robotic system can be fully controlled or observed. If the rank is full (i.e., equal to the number of states), the system is fully controllable or observable, meaning that we can drive the system to any state or accurately observe its state. This is essential for designing effective controllers and estimators.

Kinematics

In both forward and inverse kinematics, the rank of the Jacobian matrix is critical. The Jacobian matrix relates joint velocities to end-effector velocities. If the Jacobian has full rank, the robot has enough degrees of freedom to move freely in the workspace. A loss of rank in the Jacobian can indicate singularities, where the robot's movement becomes constrained or certain motions are not achievable.

Redundancy and Degrees of Freedom

In redundant robotic systems, where the number of actuators (degrees of freedom) exceeds the number of tasks, the rank of the task-related matrix helps determine how the system can distribute the tasks across different joints or actuators. This is important for optimizing performance, energy consumption, or avoiding obstacles.

For example, a robotic arm with 7 degrees of freedom may be tasked with positioning its end-effector (the hand) at a specific location in 3D space. Since positioning in 3D space only requires 3 degrees of freedom, the robot has 4 additional degrees of freedom, which makes it redundant.

In this context, the rank of the task-related matrix (such as the Jacobian matrix, which relates joint velocities to end-effector velocities) tells us whether the robot can fully utilize its redundancy. If the matrix has full rank, it means the robot can distribute the task across all available joints or actuators effectively. This allows the robot to optimize its movements for various objectives, such as:

- Minimizing energy consumption: By choosing joint configurations that use less energy.
- Avoiding obstacles: By adjusting its joints in a way that avoids collisions while still completing the task.
- Increasing dexterity: By exploiting additional degrees of freedom to maneuver in tight or constrained environments.

If the rank of the matrix drops (e.g., due to a singularity), some degrees of freedom may become constrained, reducing the robot's ability to take advantage of its redundancy. This could limit the robot's flexibility and make it harder to achieve optimal performance.

Solvability of Linear Systems

Many robotic algorithms involve solving systems of linear equations, such as for kinematics, dynamics, or control. The rank of the coefficient matrix indicates whether the system has a unique solution, infinitely many solutions, or no solution. Full rank implies a unique solution, which is often desired in robotics for precise control and motion.

Singularity Detection

Singularities in robotic systems correspond to points where the rank of the Jacobian matrix drops. At these points, the robot loses one or more degrees of freedom, which can result in control difficulties or the inability to move in certain directions. Detecting these singularities using rank is crucial for avoiding such configurations.

Conclusion

Thus, the rank of a matrix provides vital information about the feasibility of control, the existence of singularities, and the overall capabilities of a robotic system.

Exercise:

Given the matrix:

$$E = \begin{pmatrix} 2 & 4 & 1 \\ 0 & 0 & 0 \\ 3 & 6 & 2 \end{pmatrix}$$

a) Find the rank of matrix E.

b) Based on the rank, determine whether the matrix is full rank or not.

Solution:

Steps to Calculate the Rank:

Form the Matrix: The rank of a matrix is defined as the maximum number of linearly independent rows (or columns) in the matrix. We determine this by row-reducing the matrix to its row echelon form (REF).

Row Reduction (Gaussian Elimination): We apply elementary row operations to reduce the matrix E to row echelon form. The operations include:

- Swapping two rows.
- Multiplying a row by a non-zero scalar.
- Adding or subtracting a multiple of one row from another row.

Step-by-Step Row Reduction:

We start with the matrix E:

$$E = \begin{pmatrix} 2 & 4 & 1 \\ 0 & 0 & 0 \\ 3 & 6 & 2 \end{pmatrix}$$

Step 1: Make the pivot of the first column a 1 by dividing the first row by 2:

$$\text{Row } 1 \to \frac{1}{2} \times \text{Row } 1$$

This gives:

$$\begin{pmatrix} 1 & 2 & \frac{1}{2} \\ 0 & 0 & 0 \\ 3 & 6 & 2 \end{pmatrix}$$

Step 2: Eliminate the entries below the pivot (1 in the first column) by subtracting appropriate multiples of the first row from the third row:

$$\text{Row } 3 \to \text{Row } 3 - 3 \times \text{Row } 1$$

This gives:

$$\begin{pmatrix} 1 & 2 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Conclusion:

The matrix is now in row echelon form, and we can count the number of non-zero rows. In this case, the non-zero rows are:

$$\begin{pmatrix} 1 & 2 & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Thus, there are 2 non-zero rows, so the **rank** of the matrix E is:

$$\text{Rank}(E) = 2$$

Final Result:

The rank of the matrix E is 2.

a) The rank of E is 2 (since two rows are linearly independent).

b) Since the rank is less than the number of columns, the matrix is not full rank.

8. Cross Product (3D Vectors)

The **cross product** of 3D vectors is a fundamental operation in robotics. The cross product is used extensively in various robotic applications such as calculating torques, rotational dynamics, force interactions, and orientation control. Here are some reasons why this is important:

Torque and Force Calculations

In robotics, the cross product is often used to compute the **torque** produced by a force applied at a distance (lever arm) from the axis of rotation. Given a force vector $\mathbf{F}$ applied at a position $\mathbf{r}$ relative to the origin, the torque $\boldsymbol{\tau}$ is calculated as:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

For example, in a 7-DOF robotic arm, when the arm exerts force on an object (e.g., lifting or rotating), the cross product allows us to calculate the resulting torque that the joints must provide to maintain or change the orientation and position of the end-effector.

Jacobian and Velocity Calculations

The **Jacobian matrix**, which relates joint velocities to end-effector velocities, uses the cross product to compute the linear velocity contributions of rotational joints. For a robotic arm, if a joint rotates around an axis, the cross product of the joint's rotation axis and the vector from the joint to the end-effector provides the linear velocity of the end-effector caused by that joint's motion.

In a 7-DOF arm, which may have complex joint interactions, the cross product helps in calculating the individual contributions of each joint to the end-effector's velocity, making the arm's motion smooth and precise.

Orientation and Rotational Dynamics

The cross product is crucial for **rotation operations**. For instance, to compute the angular velocity of the robot's links, the cross product is used between the angular velocity vectors and the position vectors of the robot's links. This is especially important in controlling a robotic arm's orientation in space (e.g., when manipulating objects or during complex tasks such as screwing or welding).

In a 6-DOF or 7-DOF arm, where the end-effector must maintain a specific orientation (e.g., in tasks like assembling), the cross product ensures that orientation control and dynamic adjustments are handled correctly.

Collision Avoidance and Interaction with the Environment

The cross product is also used to compute vectors orthogonal to the surfaces of objects for tasks like **collision avoidance** or **force feedback**. For example, if the robotic arm is interacting with an object, the cross product of surface normals and the applied forces helps calculate how the arm should adjust its trajectory to avoid collisions or apply proper forces without damaging the object.

Example: 7-DOF Robotic Arm

Consider a 7-DOF robotic arm performing a task like picking up an object and moving it to another location. The joints of the arm must rotate and translate to achieve both the correct position and orientation of the end-effector.

- **Torque Calculation**: When the arm grips the object and begins to lift, the cross product is used to calculate the torque required by each joint to counteract the gravitational force and move the object without losing control of its orientation.
- **Jacobian Velocity Calculation**: As the arm moves, the Jacobian matrix uses the cross product to compute the contribution of each rotational joint to the overall motion of the end-effector. This ensures that the arm moves smoothly and accurately to the target position.
- **Orientation Control**: The cross product is used to calculate angular velocities that control the end-effector's orientation, ensuring the arm holds the object at the correct angle throughout the movement.

The cross product is crucial in robotics for calculating torques, handling rotational dynamics, controlling end-effector orientation, and managing complex interactions between robotic joints and external forces. For multi-DOF arms, like a 7-DOF robotic arm, the cross product ensures precise, smooth, and stable movement and manipulation, making it an indispensable tool in robotic control and operations.

Exercise:

Given vectors $\mathbf{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ and $\mathbf{v} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$, compute the cross product $\mathbf{u} \times \mathbf{v}$.

Solution:

Cross Product (Vector Product)

The cross product of two 3D vectors $\mathbf{u}$ and $\mathbf{v}$ is calculated using the formula:

$$\mathbf{u} \times \mathbf{v} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$$

Substituting the components of $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u} \times \mathbf{v} = \begin{pmatrix} (2)(6) - (3)(5) \\ (3)(4) - (1)(6) \\ (1)(5) - (2)(4) \end{pmatrix} = \begin{pmatrix} 12 - 15 \\ 12 - 6 \\ 5 - 8 \end{pmatrix} = \begin{pmatrix} -3 \\ 6 \\ -3 \end{pmatrix}$$

Thus, the cross product $\mathbf{u} \times \mathbf{v} = \begin{pmatrix} -3 \\ 6 \\ -3 \end{pmatrix}$.

9. Projection of a Vector

In robotics, the **projection of a vector** is used to break down complex movements or forces into simpler components along different directions. This concept is essential for analyzing and controlling robotic systems, such as when decomposing forces, velocities, or motions into useful components for manipulation, navigation, or control.

Definition of Projection

The **projection** of vector $\mathbf{a}$ onto vector $\mathbf{b}$ is the component of $\mathbf{a}$ that lies in the direction of $\mathbf{b}$. This is a way to measure how much of vector $\mathbf{a}$ is aligned with vector $\mathbf{b}$.

Mathematically, the projection of $\mathbf{a}$ onto $\mathbf{b}$ is given by:

$$\text{proj}_{\mathbf{b}}\mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}}\mathbf{b}$$

where:

- $\mathbf{a} \cdot \mathbf{b}$ is the dot product of vectors $\mathbf{a}$ and $\mathbf{b}$,
- $\mathbf{b} \cdot \mathbf{b}$ is the dot product of vector $\mathbf{b}$ with itself, representing the square of the magnitude of $\mathbf{b}$.

Vector projection can be used for:

Decomposition of Forces and Velocities

In robotic systems, external forces or velocities acting on a robot may be applied in arbitrary directions. To control the robot effectively, these forces or velocities are often decomposed into components along the robots specific axes, such as the robots arm or base frame.

For example, the force exerted by a tool at the end-effector of a robotic arm might need to be decomposed into components along the robots joints or along a specific axis to understand how it influences the motion or to control the force applied in that direction.

Motion Planning

When planning the motion of a robot, especially in environments with obstacles, it is often necessary to project the robots velocity or position vectors onto specific directions (e.g., along a path or towards a goal) to determine how much progress is being made in that direction.

The projection helps in computing the alignment of the robots movement with the intended trajectory, which is crucial for tasks like navigation or path following.

Control in Different Reference Frames

Robots operate in multiple reference frames, such as the world frame or base frame. To apply control strategies effectively, vectors like velocities or forces may need to be projected from one reference frame to another. This ensures that control inputs are applied in the correct direction relative to the robots configuration and environment.

Gravitational Compensation

In robotic manipulation tasks, particularly with manipulators or humanoid robots, gravitational compensation involves projecting the gravitational force vector onto the robot's joint axes. This allows the robot to calculate the necessary torques or forces at each joint to counteract the effects of gravity and maintain a desired posture or motion.

Consider a 6-DOF robotic arm applying a force $\mathbf{F}$ to move an object. The projection of the force vector $\mathbf{F}$ onto each joint's axis allows the robot to determine how much of the force contributes to motion along each joint's direction. This projection is critical for:

- Distributing the load across different joints,
- Controlling joint movements,
- Ensuring that the robot applies the correct forces to perform the task efficiently.

By using vector projections, the robot ensures that its control system accurately reflects the physical effects of forces and movements within the multi-joint robotic system.

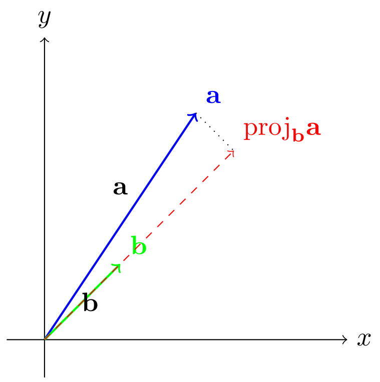
Exercise:

Given vectors $\mathbf{a} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, find the projection of vector $\mathbf{a}$ onto vector $\mathbf{b}$.

Solution:

The projection of $\mathbf{a}$ onto $\mathbf{b}$ is:

$$\text{proj}_{\mathbf{b}}\mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}}\mathbf{b} = \frac{2 \cdot 1 + 3 \cdot 1}{1^2 + 1^2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{5}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix}.$$



- The **blue vector** represents $\mathbf{a}$.
- The **green vector** represents $\mathbf{b}$.
- The **red dashed vector** represents the projection of $\mathbf{a}$ onto $\mathbf{b}$, which is $\text{proj}_{\mathbf{b}}\mathbf{a}$.
- The **dotted line** from the tip of $\mathbf{a}$ to the tip of the projection shows the relationship between $\mathbf{a}$ and its projection.

This projection is useful in robotics for decomposing movements, forces, or velocities along specific directions or axes, allowing for precise control and understanding of the robots behavior in different reference frames.

10. Change of Basis

The **change of basis** is crucial for robotics because it allows the transformation of vectors, such as positions, velocities, and forces, between different coordinate frames. This capability is essential for tasks like kinematics, control, and sensor data processing. Robotics systems often operate in multiple reference frames (such as the world frame, base frame, and end-effector frame), and understanding the relationship between these frames is vital for accurate robot operation.

Key Reasons Why Change of Basis is Important in Robotics:

Coordinate Transformations

Robots often need to operate in different coordinate systems. For example, a robot may compute its movements in its local joint frame but execute them in the global world frame. A change of basis allows the transformation of vectors between these different coordinate systems.

Kinematics

- **Forward Kinematics**: In forward kinematics, the position and orientation of the end-effector are computed in the world frame from joint angles in the robots local frame. This involves changing the basis from the joint frame to the global frame.
- **Inverse Kinematics**: In inverse kinematics, the robot must compute the required joint angles to achieve a desired position in the world frame. Here, changing the basis helps translate the world frame goal into joint-space commands.

Control

- **Controller Design**: Many control algorithms (e.g., for trajectory tracking or force control) require transforming control inputs and state variables between different frames. For instance, a control system may need to compute the required torques in the joint frame but apply them in the world frame.
- **Gravitational Compensation**: To compensate for gravity, forces need to be computed in the global frame, but these forces must be applied to individual joints. Changing the basis allows for the correct distribution of forces to the relevant joints.

Sensor Fusion

Sensors such as cameras, LiDAR, robot cameras or IMUs[1] typically measure data in their own local frame. To integrate this data with the robot's operation, the data needs to be transformed into a common reference frame, such as the world or base frame. This change of basis allows different sensor data to be combined effectively, enhancing the robot's understanding of its environment.

[1]Inertial Measurement Unit

Path Planning and Motion Control

In path planning, a robot may need to change its planned trajectory from one coordinate frame to another. For instance, a trajectory may be planned relative to a robot's base, but the execution must occur in a global frame as the robot interacts with its environment. The change of basis enables the translation of this trajectory to the appropriate frame for execution.

Simultaneous Localization and Mapping (SLAM)

In SLAM, robots must keep track of their position relative to a map while continuously updating the map with new observations. SLAM algorithms use a change of basis to update the robot's pose and the map coordinates as new sensor data is received, ensuring that everything is consistent in a common reference frame.

Example: Change of Basis Using `base_link` and `camera_link`

Consider a mobile robot with a camera mounted on it. The robot has two reference frames:

- `base_link`: This is the reference frame attached to the robot's body, representing the robot's origin.
- `camera_link`: This is the reference frame attached to the camera, defining its orientation and position relative to the `base_link`.

Suppose the camera is mounted on top of the robot, facing forward but offset from the robot's center. When the camera detects a visual target, the target's position is initially given in the `camera_link` frame. However, for the robot to act on this information, it needs to know the target's position in the `base_link` frame.

Transformation from `camera_link` to `base_link`

To transform the target's position from the `camera_link` frame to the `base_link` frame, a change of basis is required.

1. Camera Frame Data

Assume the camera detects a target and provides its position as:

$$\mathbf{p}_{\text{camera}} = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$$

This position is in the `camera_link` frame and represents a target 2 meters in front of the camera, 0 meters to the side, and 3 meters above the camera.

2. Transformation Matrix

The robot knows the position and orientation of the camera relative to the `base_link` frame. This relationship is described by a **transformation matrix** $T_{\text{base}}^{\text{camera}}$, which includes both rotation and translation:

$$T_{\text{base}}^{\text{camera}} = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

where R is the rotation matrix representing the camera's orientation relative to the robot's base, and $\mathbf{t}$ is the translation vector representing the camera's position relative to the robot's base.

3. Transformation Application

To transform the target's position from the `camera_link` frame to the `base_link` frame, the robot multiplies the position vector $\mathbf{p}_{\text{camera}}$ by the transformation matrix:

$$\mathbf{p}_{\text{base}} = T_{\text{base}}^{\text{camera}} \cdot \mathbf{p}_{\text{camera}}$$

4. Resulting Position in `base_link`

After the transformation, the robot now has the position of the target in its own `base_link` frame:

$$\mathbf{p}_{\text{base}} = \begin{pmatrix} x_{\text{base}} \\ y_{\text{base}} \\ z_{\text{base}} \end{pmatrix}$$

This information can be used by the robot for navigation or manipulation tasks, such as moving towards the target.

Practical Application

- **Object Grasping**: A robot arm mounted on the mobile robot could use the target's position in the `base_link` frame to calculate the joint angles needed to reach and grasp the object.
- **Navigation**: The robot can plan a path to approach the target based on its known position relative to the robot's base.

The change of basis allows the robot to translate sensor data from one frame (e.g., `camera_link`) into another frame (e.g., `base_link`), enabling the robot to make meaningful decisions and perform actions based on that data in a consistent global frame of reference.

Exercise:

A mobile robot has a camera mounted on top, and there are two reference frames defined:

- `base_link`: The robot's body frame, with its origin at the center of the robot's base.
- `camera_link`: The camera's frame, with its origin at the camera's optical center.

The transformation between the `camera_link` and `base_link` frames is given by the following transformation matrix:

$$T_{\text{base}}^{\text{camera}} = \begin{pmatrix} 0 & -1 & 0 & 0.5 \\ 1 & 0 & 0 & 1.0 \\ 0 & 0 & 1 & 1.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This matrix describes the rotation and translation from the `camera_link` frame to the `base_link` frame. The position of an object detected by the camera is given by the vector:

$$\mathbf{p}_{\text{camera}} = \begin{pmatrix} 2 \\ 1 \\ 3 \\ 1 \end{pmatrix}$$

1. Transform the position of the object from the `camera_link` frame to the `base_link` frame using the transformation matrix $T_{\text{base}}^{\text{camera}}$.
2. Explain the significance of this transformation in the context of robot control or navigation.

Solution

To transform the position vector $\mathbf{p}_{\text{camera}}$ from the `camera_link` frame to the `base_link` frame, we apply the transformation matrix $T_{\text{base}}^{\text{camera}}$ as follows:

$$\mathbf{p}_{\text{base}} = T_{\text{base}}^{\text{camera}} \cdot \mathbf{p}_{\text{camera}}$$

Substitute the values:

$$\mathbf{p}_{\text{base}} = \begin{pmatrix} 0 & -1 & 0 & 0.5 \\ 1 & 0 & 0 & 1.0 \\ 0 & 0 & 1 & 1.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 3 \\ 1 \end{pmatrix}$$

Perform the matrix multiplication:

$$\mathbf{p}_{\text{base}} = \begin{pmatrix} (0 \times 2) + (-1 \times 1) + (0 \times 3) + (0.5 \times 1) \\ (1 \times 2) + (0 \times 1) + (0 \times 3) + (1.0 \times 1) \\ (0 \times 2) + (0 \times 1) + (1 \times 3) + (1.5 \times 1) \\ (0 \times 2) + (0 \times 1) + (0 \times 3) + (1 \times 1) \end{pmatrix} = \begin{pmatrix} -1 + 0.5 \\ 2 + 1.0 \\ 3 + 1.5 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.5 \\ 3.0 \\ 4.5 \\ 1 \end{pmatrix}$$

Thus, the position of the object in the `base_link` frame is:

$$\mathbf{p}_{\text{base}} = \begin{pmatrix} -0.5 \\ 3.0 \\ 4.5 \\ 1 \end{pmatrix}$$