In [1]:
```
import numpy as np
from scipy.linalg import expm, logm
```

Exam 1 Review

Double check row/column vector and 2D matrix formatting

In [2]:
```
# This is a row vector
row_vec = np.array([1, 2, 3])
print("Row vector:\n", row_vec)
print("Shape of row vector:", row_vec.shape)

# This is a column vector
col_vec = np.array([[1], [2], [3]])
print("Column vector:\n", col_vec)
print("Shape of column vector:", col_vec.shape)
```

```
Row vector:
 [1 2 3]
Shape of row vector: (3,)
Column vector:
 [[1]
 [2]
 [3]]
Shape of column vector: (3, 1)
```

In [3]:
```
# This is a 2D matrix
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Matrix:\n", matrix)
print("Shape of matrix:", matrix.shape)

# This is not a 2D matrix
not_matrix = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]]])
print("Not a 2D matrix:\n", not_matrix)
print("Shape of not a 2D matrix:", not_matrix.shape)
```

```
Matrix:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape of matrix: (3, 3)
Not a 2D matrix:
 [[[1 2 3]
  [4 5 6]
  [7 8 9]]]
Shape of not a 2D matrix: (1, 3, 3)
```

3D rotation matrix format

In [4]:
```
# This is an example of a 3D rotation matrix
theta = np.radians(45)  # Rotate 45 degrees around the Z-axis
R_z = np.array([[np.cos(theta), -np.sin(theta), 0],
                [np.sin(theta), np.cos(theta), 0],
                [0, 0, 1]])
print("Rotation matrix R_z:\n", R_z)
print("Shape of R_z:", R_z.shape)
print('Determinant of the rotation matrix R_z:', np.linalg.det(R_z))
```

```
Rotation matrix R_z:
 [[ 0.70710678 -0.70710678  0.        ]
 [ 0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]
Shape of R_z: (3, 3)
Determinant of the rotation matrix R_z: 1.0
```

3D homogeneous transformation matrix

In [5]:
```
# A homogeneous transformation matrix combines a 3x3 rotation matrix and a 3x1 translation vector
R = R_z
t = np.array([[1], [2], [3]])
T = np.block([[R, t], [0, 0, 0, 1]])
print("Homogeneous transformation matrix T:\n", T)
print("Shape of T:", T.shape)
```

```
Homogeneous transformation matrix T:
 [[ 0.70710678 -0.70710678  0.          1.        ]
 [ 0.70710678  0.70710678  0.          2.        ]
 [ 0.          0.          1.          3.        ]
 [ 0.          0.          0.          1.        ]]
Shape of T: (4, 4)
```

Screw axis

In [6]:
```
# A screw axis is represented as a 6D column vector (angular velocity and linear velocity)
screw_axis = np.array([[0], [0], [1], [1], [0], [0]])  # Rotate around Z-axis and translate along X-axis
print("Screw axis:\n", screw_axis)
print("Shape of screw axis:", screw_axis.shape)
```

```
Screw axis:
 [[0]
 [0]
 [1]
 [1]
 [0]
 [0]]
Shape of screw axis: (6, 1)
```

In [7]:
```
# The matrix form of a screw axis can be represented as a 4x4 matrix

# First we need to extract the angular velocity (omega) and linear velocity (v) from the screw axis
omega1, omega2, omega3 = screw_axis[0:3, 0]
v1, v2, v3 = screw_axis[3:6, 0]
v = np.array([[v1], [v2], [v3]])

print("Linear velocity (v):\n", v)

# The skew-symmetric matrix of the angular velocity (omega) is given by:
omega_hat = np.array([[0, -omega3, omega2],
                      [omega3, 0, -omega1],
                      [-omega2, omega1, 0]])

print("Skew-symmetric matrix of angular velocity (omega_hat):\n", omega_hat)

# The matrix form of the screw axis (S) can be represented as:
S = np.block([[omega_hat, v], [0, 0, 0, 0]])
print("Matrix form of the screw axis S:\n", S)
print("Shape of S:", S.shape)
```

```
Linear velocity (v):
 [[1]
 [0]
 [0]]
Skew-symmetric matrix of angular velocity (omega_hat):
 [[ 0 -1  0]
 [ 1  0  0]
 [ 0  0  0]]
Matrix form of the screw axis S:
 [[ 0 -1  0  1]
 [ 1  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of S: (4, 4)
```

Adjoint of a transformation matrix

In [8]:
```
# The adjoint of a homogeneous transformation matrix T is given by:
R = T[0:3, 0:3]
t = T[0:3, 3]
print("Rotation part of T (R):\n", R)
print("Translation part of T (t):\n", t)
block_t = np.array([[0, -t[2], t[1]],
                    [t[2], 0, -t[0]],
                    [-t[1], t[0], 0]])
print("Block matrix of translation (block_t):\n", block_t)
Ad_T = np.block([[R, np.zeros((3, 3))], [block_t @ R, R]])
print("Adjoint of the homogeneous transformation matrix T (Ad_T):\n", Ad_T)
print("Shape of Ad_T:", Ad_T.shape)
```

```
Rotation part of T (R):
 [[ 0.70710678 -0.70710678  0.        ]
 [ 0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]
Translation part of T (t):
 [1. 2. 3.]
Block matrix of translation (block_t):
 [[ 0. -3.  2.]
 [ 3.  0. -1.]
 [-2.  1.  0.]]
Adjoint of the homogeneous transformation matrix T (Ad_T):
 [[ 0.70710678 -0.70710678  0.          0.          0.          0.        ]
 [ 0.70710678  0.70710678  0.          0.          0.          0.        ]
 [ 0.          0.          1.          0.          0.          0.        ]
 [-2.12132034 -2.12132034  2.          0.70710678 -0.70710678  0.        ]
 [ 2.12132034 -2.12132034 -1.          0.70710678  0.70710678  0.        ]
 [-0.70710678  2.12132034  0.          0.          0.          1.        ]]
Shape of Ad_T: (6, 6)
```

Forward Kinematics Practice

1R Robot

In [9]:
```python
# Let's build up the M matrix
R = np.array([[-1, 0, 0],
              [0 , 0, 1],
              [0 , 1, 0]])

print("Rotation matrix R:\n", R)
print("Shape of R:", R.shape)
print("Determinant of R:", np.linalg.det(R))

t = np.array([[-4],
              [0 ],
              [0 ]])

print("Translation vector t:\n", t)
print("Shape of t:", t.shape)

M = np.block([[R, t],
              [0, 0, 0, 1]])

print("Homogeneous transformation matrix M:\n", M)
print("Shape of M:", M.shape)
```

```
Rotation matrix R:
 [[-1  0  0]
 [ 0  0  1]
 [ 0  1  0]]
Shape of R: (3, 3)
Determinant of R: 1.0
Translation vector t:
 [[-4]
 [ 0]
 [ 0]]
Shape of t: (3, 1)
Homogeneous transformation matrix M:
 [[-1  0  0 -4]
 [ 0  0  1  0]
 [ 0  1  0  0]
 [ 0  0  0  1]]
Shape of M: (4, 4)
```

In [10]:
```python
# Joint 0 Screw axis (fixed frame)
w_s_0 = np.array([[-1],
                  [0 ],
                  [0 ]])
print("Angular velocity (w_s_0):\n", w_s_0)
q_s_0 = np.array([[-2],
                  [0 ],
                  [0 ]])
print("Point on the screw axis (q_s_0):\n", q_s_0)
v_s_0 = np.cross(-w_s_0.flatten(), q_s_0.flatten()).reshape(3, 1)
print("Linear velocity (v_s_0):\n", v_s_0)
s_0 = np.block([[w_s_0],
                [v_s_0]])
print("Screw axis for joint 0 (s_0):\n", s_0)
print("Shape of s_0:", s_0.shape)
```

```
# Joint 0 Screw axis (body frame)
w_b_0 = np.array([[1],
                  [0],
                  [0]])
print("Angular velocity (w_b_0):\n", w_b_0)
q_b_0 = np.array([[-2],
                  [0 ],
                  [0 ]])
print("Point on the screw axis (q_b_0):\n", q_b_0)
v_b_0 = np.cross(-w_b_0.flatten(), q_b_0.flatten()).reshape(3, 1)
print("Linear velocity (v_b_0):\n", v_b_0)
b_0 = np.block([[w_b_0],
                [v_b_0]])
print("Screw axis for joint 0 (b_0):\n", b_0)
print("Shape of b_0:", b_0.shape)
```

```
Angular velocity (w_s_0):
 [[-1]
 [ 0]
 [ 0]]
Point on the screw axis (q_s_0):
 [[-2]
 [ 0]
 [ 0]]
Linear velocity (v_s_0):
 [[0]
 [0]
 [0]]
Screw axis for joint 0 (s_0):
 [[-1]
 [ 0]
 [ 0]
 [ 0]
 [ 0]
 [ 0]]
Shape of s_0: (6, 1)
Angular velocity (w_b_0):
 [[1]
 [0]
 [0]]
Point on the screw axis (q_b_0):
 [[-2]
 [ 0]
 [ 0]]
Linear velocity (v_b_0):
 [[0]
 [0]
 [0]]
Screw axis for joint 0 (b_0):
 [[1]
 [0]
 [0]
 [0]
 [0]
 [0]]
Shape of b_0: (6, 1)
```

In [11]:
```
# Function to form the skew-symmetric matrix of a vector
def skew_symmetric(vec):
    return np.array([[0, -vec[2, 0], vec[1, 0]],
                     [vec[2, 0], 0, -vec[0, 0]],
                     [-vec[1, 0], vec[0, 0], 0]])
```

In [12]:
```
# Joint 0 Screw matrix (fixed frame)
bracket_w_s_0 = skew_symmetric(w_s_0)
print("Skew-symmetric matrix of w_s_0 (bracket_w_s_0):\n", bracket_w_s_0)

mat_s_0 = np.block([[bracket_w_s_0, v_s_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_0 (mat_s_0):\n", mat_s_0)
print("Shape of mat_s_0:", mat_s_0.shape)

# Joint 0 Screw matrix (body frame)
bracket_w_b_0 = skew_symmetric(w_b_0)
print("Skew-symmetric matrix of w_b_0 (bracket_w_b_0):\n", bracket_w_b_0)

mat_b_0 = np.block([[bracket_w_b_0, v_b_0],
                    [0, 0, 0, 0]])
```

```
print("Matrix form of screw axis b_0 (mat_b_0):\n", mat_b_0)
print("Shape of mat_b_0:", mat_b_0.shape)
```

```
Skew-symmetric matrix of w_s_0 (bracket_w_s_0):
 [[ 0  0  0]
 [ 0  0  1]
 [ 0 -1  0]]
Matrix form of screw axis s_0 (mat_s_0):
 [[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  0]
 [ 0  0  0  0]]
Shape of mat_s_0: (4, 4)
Skew-symmetric matrix of w_b_0 (bracket_w_b_0):
 [[ 0  0  0]
 [ 0  0 -1]
 [ 0  1  0]]
Matrix form of screw axis b_0 (mat_b_0):
 [[ 0  0  0  0]
 [ 0  0 -1  0]
 [ 0  1  0  0]
 [ 0  0  0  0]]
Shape of mat_b_0: (4, 4)
```

In [13]:
```
# Example theta for joint 0
theta_0 = 0.45

# Product of exponentials formula (fixed frame)
T_1_in_0_theta_s = expm(mat_s_0 * theta_0) @ M

print("Transformation matrix T_1_in_0_theta_s (fixed frame):\n", T_1_in_0_theta_s)
print("Shape of T_1_in_0_theta_s:", T_1_in_0_theta_s.shape)

# Product of exponentials formula (body frame)
T_1_in_0_theta_b = M @ expm(mat_b_0 * theta_0)

print("Transformation matrix T_1_in_0_theta_b (body frame):\n", T_1_in_0_theta_b)
print("Shape of T_1_in_0_theta_b:", T_1_in_0_theta_b.shape)
```

```
Transformation matrix T_1_in_0_theta_s (fixed frame):
 [[-1.          0.          0.         -4.        ]
 [ 0.          0.43496553  0.9004471   0.        ]
 [ 0.          0.9004471  -0.43496553  0.        ]
 [ 0.          0.          0.          1.        ]]
Shape of T_1_in_0_theta_s: (4, 4)
Transformation matrix T_1_in_0_theta_b (body frame):
 [[-1.          0.          0.         -4.        ]
 [ 0.          0.43496553  0.9004471   0.        ]
 [ 0.          0.9004471  -0.43496553  0.        ]
 [ 0.          0.          0.          1.        ]]
Shape of T_1_in_0_theta_b: (4, 4)
```

1P Robot

In [14]:
```
# Let's build up the M matrix
R = np.array([[-1, 0 , 0 ],
              [0 , 0 , -1],
              [0 , -1, 0 ]])

print("Rotation matrix R:\n", R)
print("Shape of R:", R.shape)
print("Determinant of R:", np.linalg.det(R))

t = np.array([[-4],
              [0 ],
              [2 ]])

print("Translation vector t:\n", t)
print("Shape of t:", t.shape)

M = np.block([[R, t],
              [0, 0, 0, 1]])

print("Homogeneous transformation matrix M:\n", M)
print("Shape of M:", M.shape)
```

```
Rotation matrix R:
 [[-1  0  0]
 [ 0  0 -1]
 [ 0 -1  0]]
Shape of R: (3, 3)
Determinant of R: 1.0
Translation vector t:
 [[-4]
 [ 0]
 [ 2]]
Shape of t: (3, 1)
Homogeneous transformation matrix M:
 [[-1  0  0 -4]
 [ 0  0 -1  0]
 [ 0 -1  0  2]
 [ 0  0  0  1]]
Shape of M: (4, 4)
```

In [15]:
```python
# Joint 0 Screw axis (fixed frame)
w_s_0 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_s_0):\n", w_s_0)
v_s_0 = np.array([[-1],
                  [0 ],
                  [0 ]])
print("Linear velocity (v_s_0):\n", v_s_0)
s_0 = np.block([[w_s_0],
                [v_s_0]])
print("Screw axis for joint 0 (s_0):\n", s_0)
print("Shape of s_0:", s_0.shape)

# Joint 0 Screw axis (body frame)
w_b_0 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_b_0):\n", w_b_0)
v_b_0 = np.array([[1],
                  [0],
                  [0]])
print("Linear velocity (v_b_0):\n", v_b_0)
b_0 = np.block([[w_b_0],
                [v_b_0]])
print("Screw axis for joint 0 (b_0):\n", b_0)
print("Shape of b_0:", b_0.shape)
```

```
Angular velocity (w_s_0):
 [[0]
 [0]
 [0]]
Linear velocity (v_s_0):
 [[-1]
 [ 0]
 [ 0]]
Screw axis for joint 0 (s_0):
 [[ 0]
 [ 0]
 [ 0]
 [-1]
 [ 0]
 [ 0]]
Shape of s_0: (6, 1)
Angular velocity (w_b_0):
 [[0]
 [0]
 [0]]
Linear velocity (v_b_0):
 [[1]
 [0]
 [0]]
Screw axis for joint 0 (b_0):
 [[0]
 [0]
 [0]
 [1]
 [0]
 [0]]
Shape of b_0: (6, 1)
```

In [16]:
```python
# Joint 0 Screw matrix (fixed frame)
bracket_w_s_0 = skew_symmetric(w_s_0)
```

```
print("Skew-symmetric matrix of w_s_0 (bracket_w_s_0):\n", bracket_w_s_0)

mat_s_0 = np.block([[bracket_w_s_0, v_s_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_0 (mat_s_0):\n", mat_s_0)
print("Shape of mat_s_0:", mat_s_0.shape)

# Joint 0 Screw matrix (body frame)
bracket_w_b_0 = skew_symmetric(w_b_0)
print("Skew-symmetric matrix of w_b_0 (bracket_w_b_0):\n", bracket_w_b_0)

mat_b_0 = np.block([[bracket_w_b_0, v_b_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis b_0 (mat_b_0):\n", mat_b_0)
print("Shape of mat_b_0:", mat_b_0.shape)
```

```
Skew-symmetric matrix of w_s_0 (bracket_w_s_0):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis s_0 (mat_s_0):
 [[ 0  0  0 -1]
 [ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_s_0: (4, 4)
Skew-symmetric matrix of w_b_0 (bracket_w_b_0):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis b_0 (mat_b_0):
 [[0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
Shape of mat_b_0: (4, 4)
```

In [17]:

```
# Example theta for joint 0
theta_0 = 0.45

# Product of exponentials formula (fixed frame)
T_1_in_0_theta_s = expm(mat_s_0 * theta_0) @ M

print("Transformation matrix T_1_in_0_theta_s (fixed frame):\n", T_1_in_0_theta_s)
print("Shape of T_1_in_0_theta_s:", T_1_in_0_theta_s.shape)

# Product of exponentials formula (body frame)
T_1_in_0_theta_b = M @ expm(mat_b_0 * theta_0)

print("Transformation matrix T_1_in_0_theta_b (body frame):\n", T_1_in_0_theta_b)
print("Shape of T_1_in_0_theta_b:", T_1_in_0_theta_b.shape)
```

```
Transformation matrix T_1_in_0_theta_s (fixed frame):
 [[-1.    0.    0.   -4.45]
 [ 0.    0.   -1.    0.  ]
 [ 0.   -1.    0.    2.  ]
 [ 0.    0.    0.    1.  ]]
Shape of T_1_in_0_theta_s: (4, 4)
Transformation matrix T_1_in_0_theta_b (body frame):
 [[-1.    0.    0.   -4.45]
 [ 0.    0.   -1.    0.  ]
 [ 0.   -1.    0.    2.  ]
 [ 0.    0.    0.    1.  ]]
Shape of T_1_in_0_theta_b: (4, 4)
```

RRP Robot

In [18]:

```
# Let's build up the M matrix
R = np.array([[0, 1, 0 ],
              [1, 0, 0 ],
              [0, 0, -1]])

print("Rotation matrix R:\n", R)
print("Shape of R:", R.shape)
print("Determinant of R:", np.linalg.det(R))

t = np.array([[0],
              [0 ],
              [-6]])

print("Translation vector t:\n", t)
```

```
print("Shape of t:", t.shape)

M = np.block([[R, t],
              [0, 0, 0, 1]])

print("Homogeneous transformation matrix M:\n", M)
print("Shape of M:", M.shape)
```

```
Rotation matrix R:
 [[ 0  1  0]
 [ 1  0  0]
 [ 0  0 -1]]
Shape of R: (3, 3)
Determinant of R: 1.0
Translation vector t:
 [[ 0]
 [ 0]
 [-6]]
Shape of t: (3, 1)
Homogeneous transformation matrix M:
 [[ 0  1  0  0]
 [ 1  0  0  0]
 [ 0  0 -1 -6]
 [ 0  0  0  1]]
Shape of M: (4, 4)
```

In [19]:

```
# Joint 0 Screw axis (fixed frame)
w_s_0 = np.array([[0 ],
                  [0 ],
                  [-1]])
print("Angular velocity (w_s_0):\n", w_s_0)
q_s_0 = np.array([[-2],
                  [0 ],
                  [-2]])
print("Point on the screw axis (q_s_0):\n", q_s_0)
v_s_0 = np.cross(-w_s_0.flatten(), q_s_0.flatten()).reshape(3, 1)
print("Linear velocity (v_s_0):\n", v_s_0)
s_0 = np.block([[w_s_0],
                [v_s_0]])
print("Screw axis for joint 0 (s_0):\n", s_0)
print("Shape of s_0:", s_0.shape)

# Joint 0 Screw axis (body frame)
w_b_0 = np.array([[0],
                  [0],
                  [1]])
print("Angular velocity (w_b_0):\n", w_b_0)
q_b_0 = np.array([[0 ],
                  [-2],
                  [-4]])
print("Point on the screw axis (q_b_0):\n", q_b_0)
v_b_0 = np.cross(-w_b_0.flatten(), q_b_0.flatten()).reshape(3, 1)
print("Linear velocity (v_b_0):\n", v_b_0)
b_0 = np.block([[w_b_0],
                [v_b_0]])
print("Screw axis for joint 0 (b_0):\n", b_0)
print("Shape of b_0:", b_0.shape)

# Joint 1 Screw axis (fixed frame)
w_s_1 = np.array([[0 ],
                  [0 ],
                  [-1]])
print("Angular velocity (w_s_1):\n", w_s_1)
q_s_1 = np.array([[-2],
                  [0 ],
                  [-4]])
print("Point on the screw axis (q_s_1):\n", q_s_1)
v_s_1 = np.cross(-w_s_1.flatten(), q_s_1.flatten()).reshape(3, 1)
print("Linear velocity (v_s_1):\n", v_s_1)
s_1 = np.block([[w_s_1],
                [v_s_1]])
print("Screw axis for joint 1 (s_1):\n", s_1)
print("Shape of s_1:", s_1.shape)

# Joint 1 Screw axis (body frame)
w_b_1 = np.array([[0],
                  [0],
                  [1]])
print("Angular velocity (w_b_1):\n", w_b_1)
q_b_1 = np.array([[0 ],
```

```
                  [-2],
                  [-2]])
print("Point on the screw axis (q_b_1):\n", q_b_1)
v_b_1 = np.cross(-w_b_1.flatten(), q_b_1.flatten()).reshape(3, 1)
print("Linear velocity (v_b_1):\n", v_b_1)
b_1 = np.block([[w_b_1],
                [v_b_1]])
print("Screw axis for joint 1 (b_1):\n", b_1)
print("Shape of b_1:", b_1.shape)

# Joint 2 Screw axis (fixed frame)
w_s_2 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_s_2):\n", w_s_2)
v_s_2 = np.array([[0 ],
                  [0 ],
                  [-1]])
print("Linear velocity (v_s_2):\n", v_s_2)
s_2 = np.block([[w_s_2],
                [v_s_2]])
print("Screw axis for joint 2 (s_2):\n", s_2)
print("Shape of s_2:", s_2.shape)

# Joint 2 Screw axis (body frame)
w_b_2 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_b_2):\n", w_b_2)
v_b_2 = np.array([[0],
                  [0],
                  [1]])
print("Linear velocity (v_b_2):\n", v_b_2)
b_2 = np.block([[w_b_2],
                [v_b_2]])
print("Screw axis for joint 2 (b_2):\n", b_2)
print("Shape of b_2:", b_2.shape)
```

```
Angular velocity (w_s_0):
 [[ 0]
 [ 0]
 [-1]]
Point on the screw axis (q_s_0):
 [[-2]
 [ 0]
 [-2]]
Linear velocity (v_s_0):
 [[ 0]
 [-2]
 [ 0]]
Screw axis for joint 0 (s_0):
 [[ 0]
 [ 0]
 [-1]
 [ 0]
 [-2]
 [ 0]]
Shape of s_0: (6, 1)
Angular velocity (w_b_0):
 [[0]
 [0]
 [1]]
Point on the screw axis (q_b_0):
 [[ 0]
 [-2]
 [-4]]
Linear velocity (v_b_0):
 [[-2]
 [ 0]
 [ 0]]
Screw axis for joint 0 (b_0):
 [[ 0]
 [ 0]
 [ 1]
 [-2]
 [ 0]
 [ 0]]
Shape of b_0: (6, 1)
Angular velocity (w_s_1):
 [[ 0]
 [ 0]
 [-1]]
Point on the screw axis (q_s_1):
 [[-2]
 [ 0]
 [-4]]
Linear velocity (v_s_1):
 [[ 0]
 [-2]
 [ 0]]
Screw axis for joint 1 (s_1):
 [[ 0]
 [ 0]
 [-1]
 [ 0]
 [-2]
 [ 0]]
Shape of s_1: (6, 1)
Angular velocity (w_b_1):
 [[0]
 [0]
 [1]]
Point on the screw axis (q_b_1):
 [[ 0]
 [-2]
 [-2]]
Linear velocity (v_b_1):
 [[-2]
 [ 0]
 [ 0]]
Screw axis for joint 1 (b_1):
 [[ 0]
 [ 0]
 [ 1]
 [-2]
 [ 0]
 [ 0]]
Shape of b_1: (6, 1)
```

```
Angular velocity (w_s_2):
 [[0]
 [0]
 [0]]
Linear velocity (v_s_2):
 [[ 0]
 [ 0]
 [-1]]
Screw axis for joint 2 (s_2):
 [[ 0]
 [ 0]
 [ 0]
 [ 0]
 [ 0]
 [-1]]
Shape of s_2: (6, 1)
Angular velocity (w_b_2):
 [[0]
 [0]
 [0]]
Linear velocity (v_b_2):
 [[0]
 [0]
 [1]]
Screw axis for joint 2 (b_2):
 [[0]
 [0]
 [0]
 [0]
 [0]
 [1]]
Shape of b_2: (6, 1)
```

In [20]:

```python
# Joint 0 Screw matrix (fixed frame)
bracket_w_s_0 = skew_symmetric(w_s_0)
print("Skew-symmetric matrix of w_s_0 (bracket_w_s_0):\n", bracket_w_s_0)

mat_s_0 = np.block([[bracket_w_s_0, v_s_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_0 (mat_s_0):\n", mat_s_0)
print("Shape of mat_s_0:", mat_s_0.shape)

# Joint 0 Screw matrix (body frame)
bracket_w_b_0 = skew_symmetric(w_b_0)
print("Skew-symmetric matrix of w_b_0 (bracket_w_b_0):\n", bracket_w_b_0)

mat_b_0 = np.block([[bracket_w_b_0, v_b_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis b_0 (mat_b_0):\n", mat_b_0)
print("Shape of mat_b_0:", mat_b_0.shape)

# Joint 1 Screw matrix (fixed frame)
bracket_w_s_1 = skew_symmetric(w_s_1)
print("Skew-symmetric matrix of w_s_1 (bracket_w_s_1):\n", bracket_w_s_1)

mat_s_1 = np.block([[bracket_w_s_1, v_s_1],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_1 (mat_s_1):\n", mat_s_1)
print("Shape of mat_s_1:", mat_s_1.shape)

# Joint 1 Screw matrix (body frame)
bracket_w_b_1 = skew_symmetric(w_b_1)
print("Skew-symmetric matrix of w_b_1 (bracket_w_b_1):\n", bracket_w_b_1)

mat_b_1 = np.block([[bracket_w_b_1, v_b_1],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis b_1 (mat_b_1):\n", mat_b_1)
print("Shape of mat_b_1:", mat_b_1.shape)

# Joint 2 Screw matrix (fixed frame)
bracket_w_s_2 = skew_symmetric(w_s_2)
print("Skew-symmetric matrix of w_s_2 (bracket_w_s_2):\n", bracket_w_s_2)

mat_s_2 = np.block([[bracket_w_s_2, v_s_2],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_2 (mat_s_2):\n", mat_s_2)
print("Shape of mat_s_2:", mat_s_2.shape)

# Joint 2 Screw matrix (body frame)
bracket_w_b_2 = skew_symmetric(w_b_2)
```

```
print("Skew-symmetric matrix of w_b_2 (bracket_w_b_2):\n", bracket_w_b_2)

mat_b_2 = np.block([[bracket_w_b_2, v_b_2],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis b_2 (mat_b_2):\n", mat_b_2)
print("Shape of mat_b_2:", mat_b_2.shape)
```

```
Skew-symmetric matrix of w_s_0 (bracket_w_s_0):
 [[ 0  1  0]
 [-1  0  0]
 [ 0  0  0]]
Matrix form of screw axis s_0 (mat_s_0):
 [[ 0  1  0  0]
 [-1  0  0 -2]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_s_0: (4, 4)
Skew-symmetric matrix of w_b_0 (bracket_w_b_0):
 [[ 0 -1  0]
 [ 1  0  0]
 [ 0  0  0]]
Matrix form of screw axis b_0 (mat_b_0):
 [[ 0 -1  0 -2]
 [ 1  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_b_0: (4, 4)
Skew-symmetric matrix of w_s_1 (bracket_w_s_1):
 [[ 0  1  0]
 [-1  0  0]
 [ 0  0  0]]
Matrix form of screw axis s_1 (mat_s_1):
 [[ 0  1  0  0]
 [-1  0  0 -2]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_s_1: (4, 4)
Skew-symmetric matrix of w_b_1 (bracket_w_b_1):
 [[ 0 -1  0]
 [ 1  0  0]
 [ 0  0  0]]
Matrix form of screw axis b_1 (mat_b_1):
 [[ 0 -1  0 -2]
 [ 1  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_b_1: (4, 4)
Skew-symmetric matrix of w_s_2 (bracket_w_s_2):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis s_2 (mat_s_2):
 [[ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0 -1]
 [ 0  0  0  0]]
Shape of mat_s_2: (4, 4)
Skew-symmetric matrix of w_b_2 (bracket_w_b_2):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis b_2 (mat_b_2):
 [[0 0 0 0]
 [0 0 0 0]
 [0 0 0 1]
 [0 0 0 0]]
Shape of mat_b_2: (4, 4)
```

In [21]:
```
# Example theta for joints 0, 1, and 2
theta_0 = 0.25
theta_1 = 0.50
theta_2 = 0.75

# Product of exponentials formula (fixed frame)
T_1_in_0_theta_s = expm(mat_s_0 * theta_0) @ expm(mat_s_1 * theta_1) @ expm(mat_s_2 * theta_2) @ M

print("Transformation matrix T_1_in_0_theta_s (fixed frame):\n", T_1_in_0_theta_s)
print("Shape of T_1_in_0_theta_s:", T_1_in_0_theta_s.shape)

# Product of exponentials formula (body frame)
```

```
T_1_in_0_theta_b = M @ expm(mat_b_0 * theta_0) @ expm(mat_b_1 * theta_1) @ expm(mat_b_2 * theta_2)

print("Transformation matrix T_1_in_0_theta_b (body frame):\n", T_1_in_0_theta_b)
print("Shape of T_1_in_0_theta_b:", T_1_in_0_theta_b.shape)
```

```
Transformation matrix T_1_in_0_theta_s (fixed frame):
 [[ 0.68163876  0.73168887  0.         -0.53662226]
 [ 0.73168887 -0.68163876  0.         -1.36327752]
 [ 0.          0.         -1.         -6.75      ]
 [ 0.          0.          0.          1.        ]]
Shape of T_1_in_0_theta_s: (4, 4)
Transformation matrix T_1_in_0_theta_b (body frame):
 [[ 0.68163876  0.73168887  0.         -0.53662226]
 [ 0.73168887 -0.68163876  0.         -1.36327752]
 [ 0.          0.         -1.         -6.75      ]
 [ 0.          0.          0.          1.        ]]
Shape of T_1_in_0_theta_b: (4, 4)
```

Velocity Kinematics Practice

PRPR Robot

In [22]:

```
# Joint 0 Screw axis (fixed frame)
w_s_0 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_s_0):\n", w_s_0)
v_s_0 = np.array([[-1],
                  [0 ],
                  [0 ]])
print("Linear velocity (v_s_0):\n", v_s_0)
s_0 = np.block([[w_s_0],
                [v_s_0]])
print("Screw axis for joint 0 (s_0):\n", s_0)
print("Shape of s_0:", s_0.shape)

# Joint 1 Screw axis (fixed frame)
w_s_1 = np.array([[-1],
                  [0 ],
                  [0 ]])
print("Angular velocity (w_s_1):\n", w_s_1)
q_s_1 = np.array([[-2],
                  [2 ],
                  [0 ]])
print("Point on the screw axis (q_s_1):\n", q_s_1)
v_s_1 = np.cross(-w_s_1.flatten(), q_s_1.flatten()).reshape(3, 1)
print("Linear velocity (v_s_1):\n", v_s_1)
s_1 = np.block([[w_s_1],
                [v_s_1]])
print("Screw axis for joint 1 (s_1):\n", s_1)
print("Shape of s_1:", s_1.shape)

# Joint 2 Screw axis (fixed frame)
w_s_2 = np.array([[0],
                  [0],
                  [0]])
print("Angular velocity (w_s_2):\n", w_s_2)
v_s_2 = np.array([[0],
                  [1],
                  [0]])
print("Linear velocity (v_s_2):\n", v_s_2)
s_2 = np.block([[w_s_2],
                [v_s_2]])
print("Screw axis for joint 2 (s_2):\n", s_2)
print("Shape of s_2:", s_2.shape)

# Joint 3 Screw axis (fixed frame)
w_s_3 = np.array([[-1],
                  [0 ],
                  [0 ]])
print("Angular velocity (w_s_3):\n", w_s_3)
q_s_3 = np.array([[-2],
                  [6 ],
                  [0 ]])
print("Point on the screw axis (q_s_3):\n", q_s_3)
v_s_3 = np.cross(-w_s_3.flatten(), q_s_3.flatten()).reshape(3, 1)
print("Linear velocity (v_s_3):\n", v_s_3)
s_3 = np.block([[w_s_3],
```

```
                [v_s_3]])
print("Screw axis for joint 3 (s_3):\n", s_3)
print("Shape of s_3:", s_3.shape)
```

```
Angular velocity (w_s_0):
 [[0]
 [0]
 [0]]
Linear velocity (v_s_0):
 [[-1]
 [ 0]
 [ 0]]
Screw axis for joint 0 (s_0):
 [[ 0]
 [ 0]
 [ 0]
 [-1]
 [ 0]
 [ 0]]
Shape of s_0: (6, 1)
Angular velocity (w_s_1):
 [[-1]
 [ 0]
 [ 0]]
Point on the screw axis (q_s_1):
 [[-2]
 [ 2]
 [ 0]]
Linear velocity (v_s_1):
 [[0]
 [0]
 [2]]
Screw axis for joint 1 (s_1):
 [[-1]
 [ 0]
 [ 0]
 [ 0]
 [ 0]
 [ 2]]
Shape of s_1: (6, 1)
Angular velocity (w_s_2):
 [[0]
 [0]
 [0]]
Linear velocity (v_s_2):
 [[0]
 [1]
 [0]]
Screw axis for joint 2 (s_2):
 [[0]
 [0]
 [0]
 [0]
 [1]
 [0]]
Shape of s_2: (6, 1)
Angular velocity (w_s_3):
 [[-1]
 [ 0]
 [ 0]]
Point on the screw axis (q_s_3):
 [[-2]
 [ 6]
 [ 0]]
Linear velocity (v_s_3):
 [[0]
 [0]
 [6]]
Screw axis for joint 3 (s_3):
 [[-1]
 [ 0]
 [ 0]
 [ 0]
 [ 0]
 [ 6]]
Shape of s_3: (6, 1)
```

In [23]:
```
# Joint 0 Screw matrix (fixed frame)
bracket_w_s_0 = skew_symmetric(w_s_0)
print("Skew-symmetric matrix of w_s_0 (bracket_w_s_0):\n", bracket_w_s_0)
```

```
mat_s_0 = np.block([[bracket_w_s_0, v_s_0],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_0 (mat_s_0):\n", mat_s_0)
print("Shape of mat_s_0:", mat_s_0.shape)

# Joint 1 Screw matrix (fixed frame)
bracket_w_s_1 = skew_symmetric(w_s_1)
print("Skew-symmetric matrix of w_s_1 (bracket_w_s_1):\n", bracket_w_s_1)

mat_s_1 = np.block([[bracket_w_s_1, v_s_1],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_1 (mat_s_1):\n", mat_s_1)
print("Shape of mat_s_1:", mat_s_1.shape)

# Joint 2 Screw matrix (fixed frame)
bracket_w_s_2 = skew_symmetric(w_s_2)
print("Skew-symmetric matrix of w_s_2 (bracket_w_s_2):\n", bracket_w_s_2)

mat_s_2 = np.block([[bracket_w_s_2, v_s_2],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_2 (mat_s_2):\n", mat_s_2)
print("Shape of mat_s_2:", mat_s_2.shape)

# Joint 3 Screw matrix (fixed frame)
bracket_w_s_3 = skew_symmetric(w_s_3)
print("Skew-symmetric matrix of w_s_3 (bracket_w_s_3):\n", bracket_w_s_3)

mat_s_3 = np.block([[bracket_w_s_3, v_s_3],
                    [0, 0, 0, 0]])
print("Matrix form of screw axis s_3 (mat_s_3):\n", mat_s_3)
print("Shape of mat_s_3:", mat_s_3.shape)
```

```
Skew-symmetric matrix of w_s_0 (bracket_w_s_0):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis s_0 (mat_s_0):
 [[ 0  0  0 -1]
 [ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
Shape of mat_s_0: (4, 4)
Skew-symmetric matrix of w_s_1 (bracket_w_s_1):
 [[ 0  0  0]
 [ 0  0  1]
 [ 0 -1  0]]
Matrix form of screw axis s_1 (mat_s_1):
 [[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  2]
 [ 0  0  0  0]]
Shape of mat_s_1: (4, 4)
Skew-symmetric matrix of w_s_2 (bracket_w_s_2):
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Matrix form of screw axis s_2 (mat_s_2):
 [[0 0 0 0]
 [0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]]
Shape of mat_s_2: (4, 4)
Skew-symmetric matrix of w_s_3 (bracket_w_s_3):
 [[ 0  0  0]
 [ 0  0  1]
 [ 0 -1  0]]
Matrix form of screw axis s_3 (mat_s_3):
 [[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  6]
 [ 0  0  0  0]]
Shape of mat_s_3: (4, 4)
```

In [24]:
```
# Function to form the adjoint of a homogeneous transformation matrix
def adjoint(T):
    R = T[0:3, 0:3]
    t = T[0:3, 3].reshape(3, 1)
    block_t = skew_symmetric(t)
```

```
    Ad_T = np.block([[R, np.zeros((3, 3))], [block_t @ R, R]])
    return Ad_T
```

In [26]:
```
# Example theta for joints 0, 1, 2, and 3
theta_0 = 0.25
theta_1 = 0.50
theta_2 = 0.75
theta_3 = 1.00

# Jacobian column 0 (fixed frame)
J_s_0 = s_0
print("Jacobian column for joint 0 (fixed frame) J_s_0:\n", J_s_0)

# Jacobian column 1 (fixed frame)
Ad_exp_s_0_theta_0 = adjoint(expm(mat_s_0 * theta_0))
J_s_1 = Ad_exp_s_0_theta_0 @ s_1
print("Jacobian column for joint 1 (fixed frame) J_s_1:\n", J_s_1)

# Jacobian column 2 (fixed frame)
Ad_exp_s_1_theta_1 = adjoint(expm(mat_s_1 * theta_1))
J_s_2 = Ad_exp_s_0_theta_0 @ Ad_exp_s_1_theta_1 @ s_2
print("Jacobian column for joint 2 (fixed frame) J_s_2:\n", J_s_2)

# Jacobian column 3 (fixed frame)
Ad_exp_s_2_theta_2 = adjoint(expm(mat_s_2 * theta_2))
J_s_3 = Ad_exp_s_0_theta_0 @ Ad_exp_s_1_theta_1 @ Ad_exp_s_2_theta_2 @ s_3
print("Jacobian column for joint 3 (fixed frame) J_s_3:\n", J_s_3)

# Jacobian matrix (fixed frame)
J_s = np.block([[J_s_0, J_s_1, J_s_2, J_s_3]])
print("Jacobian matrix in the fixed frame J_s:\n", J_s)
```

```
Jacobian column for joint 0 (fixed frame) J_s_0:
 [[ 0]
 [ 0]
 [ 0]
 [-1]
 [ 0]
 [ 0]]
Jacobian column for joint 1 (fixed frame) J_s_1:
 [[-1.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 2.]]
Jacobian column for joint 2 (fixed frame) J_s_2:
 [[ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.87758256]
 [-0.47942554]]
Jacobian column for joint 3 (fixed frame) J_s_3:
 [[-1.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 2.27727131]
 [ 6.16851717]]
Jacobian matrix in the fixed frame J_s:
 [[ 0.         -1.          0.         -1.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [-1.          0.          0.          0.        ]
 [ 0.          0.          0.87758256  2.27727131]
 [ 0.          2.         -0.47942554  6.16851717]]
```