Intro to Robotics: Forward and Velocity Kinematics Review

Neeloy Chakraborty

neeloyc2@illinois.edu

Agenda

- Common Mistakes on Exam
- Recap on Screws
- Forward Kinematics
- Velocity Kinematics
- Manipulability

Common Mistakes on Exam

- Formatting of row vs. column vector
- Adding an extra bracket around a 2D matrix
- Missing a 1 in the [0 0 0 1] row of a homogeneous transformation matrix
- Adding a 1 in the [0 0 0 0] row of the matrix form of the screw; i.e. [V]
- Multiplying pR instead of [p]R in the adjoint matrix
- Rounding errors

Recap on Screws

While rotation matrices capture rotation, homogeneous transformations capture both rotation and translation

We want to find an operator that governs the ODE for the pose of the robot body

$$\dot{R} = [\omega_s]R = R[\omega_b]$$

$$\dot{T} = [V_s]T = T[V_b]$$

Recap on Screws

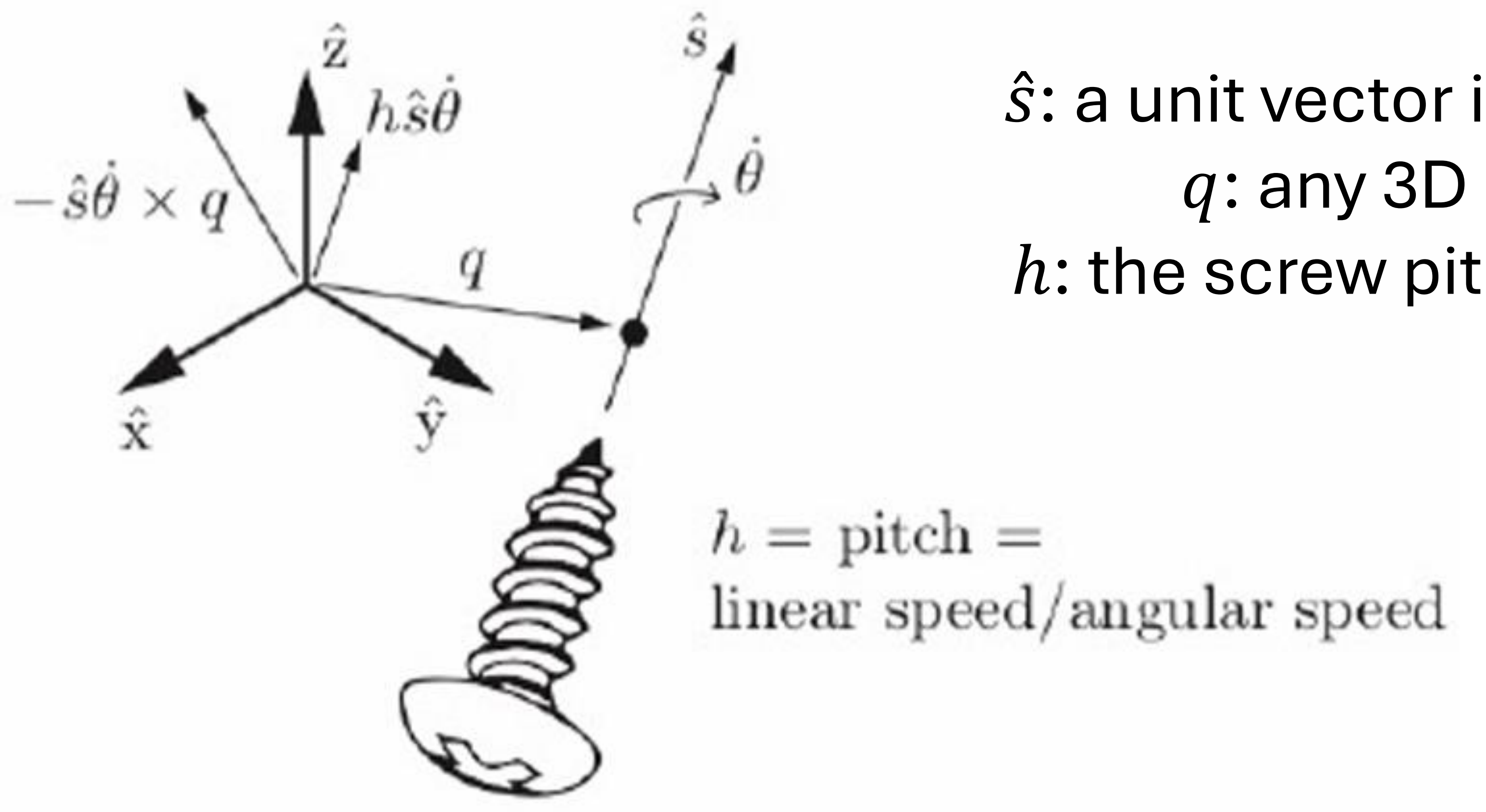
Twist V is a 6D real-valued vector containing the angular and linear velocity

$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} \quad [\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{V}_s = [Ad_T]\mathcal{V}_b = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \mathcal{V}_b$$

Recap on Screws

A twist V can be viewed in terms of a screw motion S



$\hat{s}$: a unit vector in the direction of the axis
q: any 3D point along the axis
h: the screw pitch (linear/angular speed)

Recap on Screws

Screw to Twist:

$\hat{s}$: a unit vector in the direction of the axis
q: any 3D point along the axis
h: the screw pitch (linear/angular speed)

$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \hat{s}\dot{\theta} \\ h\hat{s}\dot{\theta} - \hat{s}\dot{\theta} \times q \end{bmatrix}$$

Twist to Screw:

$$\hat{s} = \frac{\omega}{||\omega||} = \hat{\omega}, \quad \dot{\theta} = ||\omega||, \quad h = \frac{\omega^T v}{||\omega||^2}$$

$$\text{if } \omega \neq 0, \quad q = \frac{\hat{s} \times v}{\dot{\theta}}$$

Recap on Screws

A twist V for 1 second around the screw axis at angular velocity θ results in frame T

A twist V for θ seconds around the screw axis at angular velocity 1 results in frame T

Recap on Screws

$\hat{s}$: a unit vector in the direction of the axis
q: any 3D point along the axis
h: the screw pitch (linear/angular speed)

$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \hat{s}\dot{\theta} \\ h\hat{s}\dot{\theta} - \hat{s}\dot{\theta} \times q \end{bmatrix}$$

$$v = h\omega - \omega \times q$$

If $||\omega|| = 1$ and $h = 0$:
Pure rotation
Revolute joint

If $\omega = 0$ and $||v|| = 1$:
Pure translation
Prismatic joint

Solving the ODE $\dot{T} = [V_s]T$

scipy.linalg.expm()

Recall how we solved for the orientation R_{sb} given the angular velocity ω:

$$\dot{R} = [\omega_s]R = R[\omega_b]$$

$$R(\omega, \theta) = e^{[\omega_s]\theta} R(0) = R(0)e^{[\omega_b]\theta}$$

Similarly, solving for the pose T_{sb} after applying twist V:

$$\dot{T} = [V_s]T = T[V_b]$$

$$T(V, \theta) = e^{[V_s]\theta} T(0) = T(0)e^{[V_b]\theta}$$

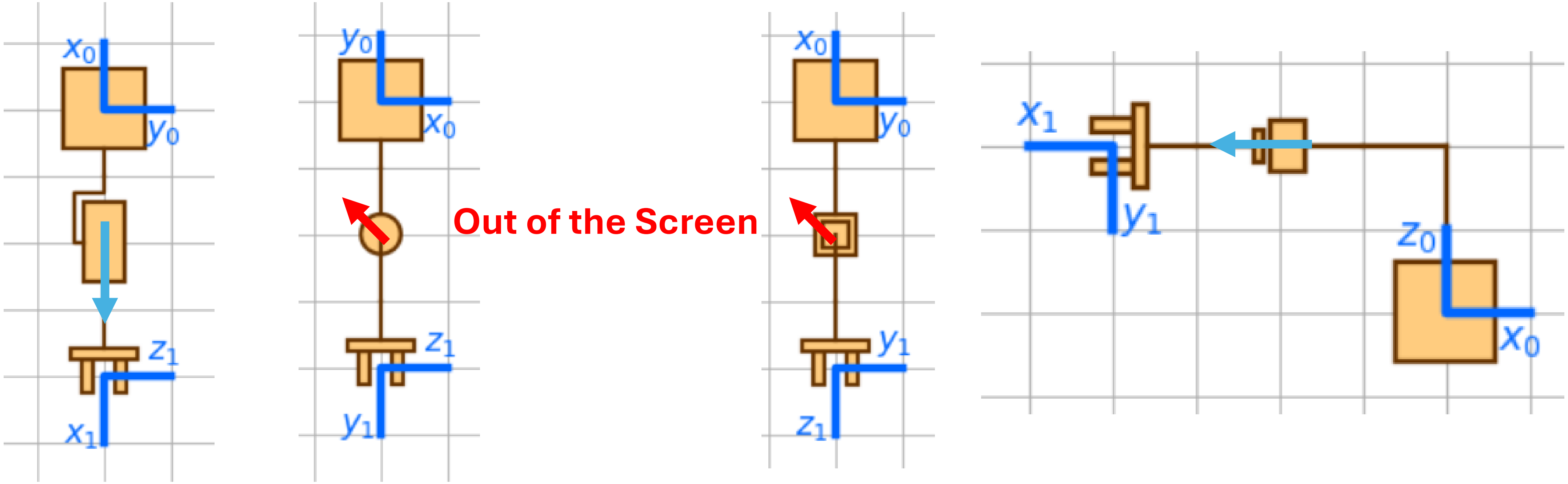
Why do we care about screws???

We can calculate the end effector pose given robot joint angles by treating each joint as a screw axis!

Forward Kinematics

Revolute Joint

Prismatic Joint



Product of Exponentials

$$T(\theta) = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} \dots e^{[\mathcal{S}_{n-1}]\theta_{n-1}} e^{[\mathcal{S}_n]\theta_n} M$$

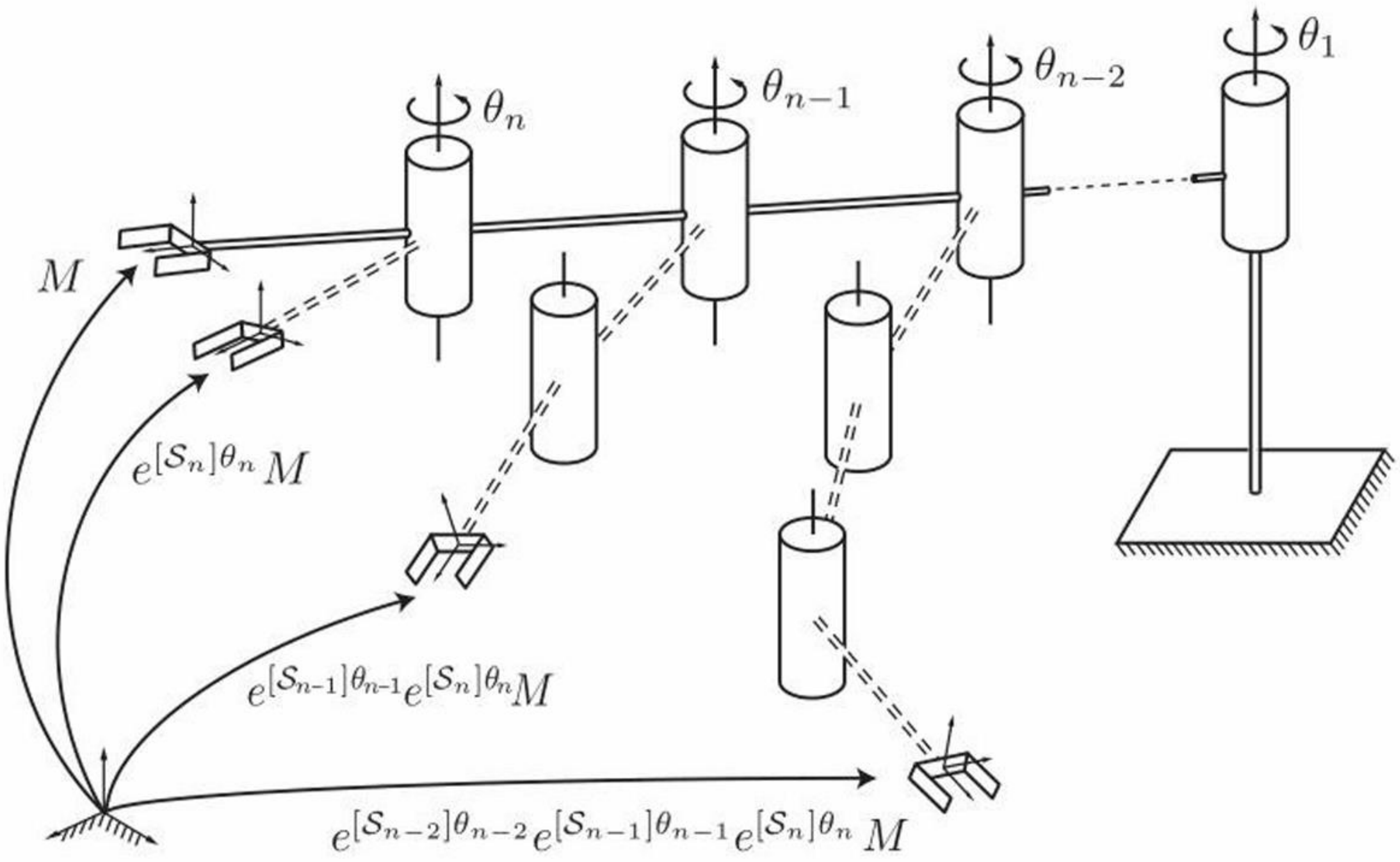


Figure 4.2: Illustration of the PoE formula for an n-link spatial open chain.

Product of Exponentials Algorithm

1. Identify the orientations of the fixed and tool frames with the right-hand rule

2. Create the homogeneous transformation M of the tool frame in the fixed frame

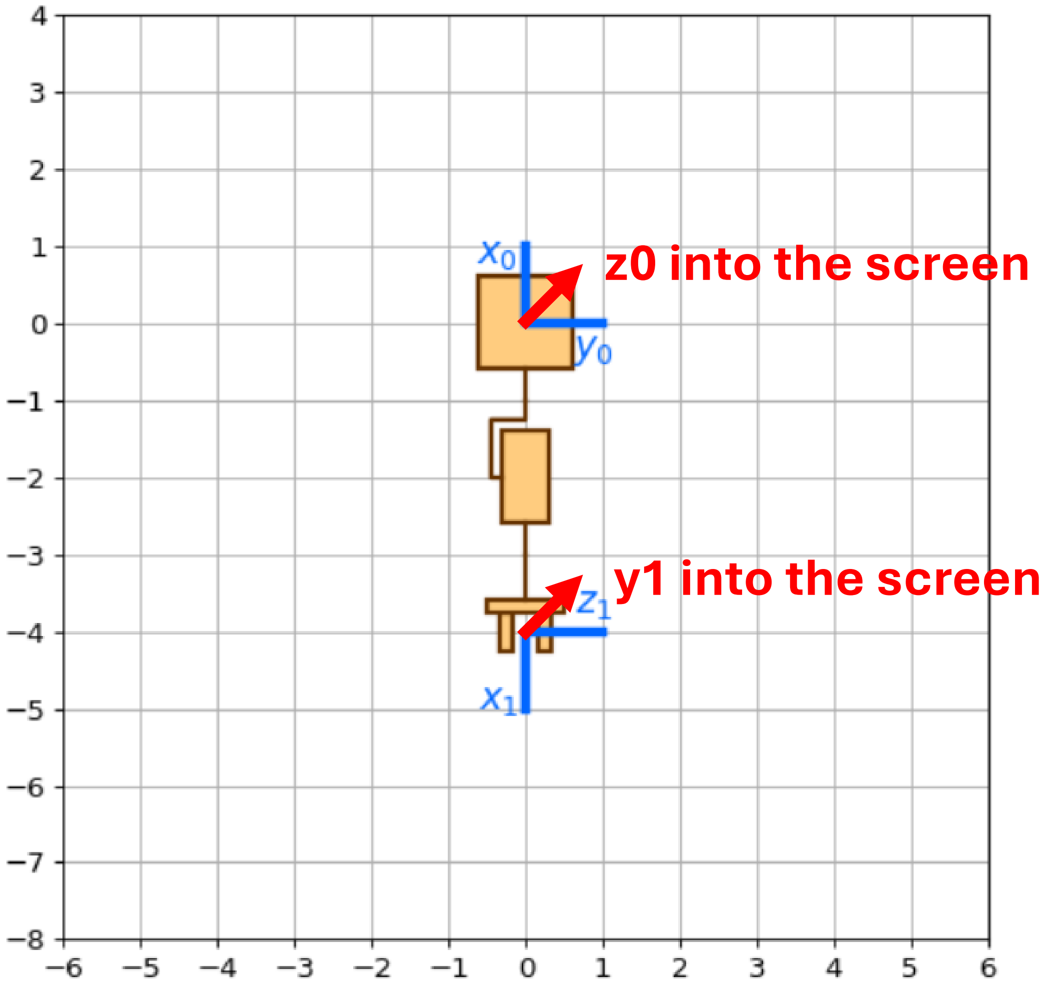
3. Build the screw axis for each joint
   - Ask yourself if we want the tool frame screw, the fixed frame screw, or either
   - If we specifically ask for S -> fixed frame screw
   - If we specifically ask for B -> body frame screw
   - Else -> your preference

4. Build the matrix form [V] of each screw axis

5. Use the PoE formula
   - If you used S in step 3 -> $T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \dots e^{[S_n]\theta_n} M$
   - If you used B in step 3 -> $T(\theta) = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} \dots e^{[B_n]\theta_n}$

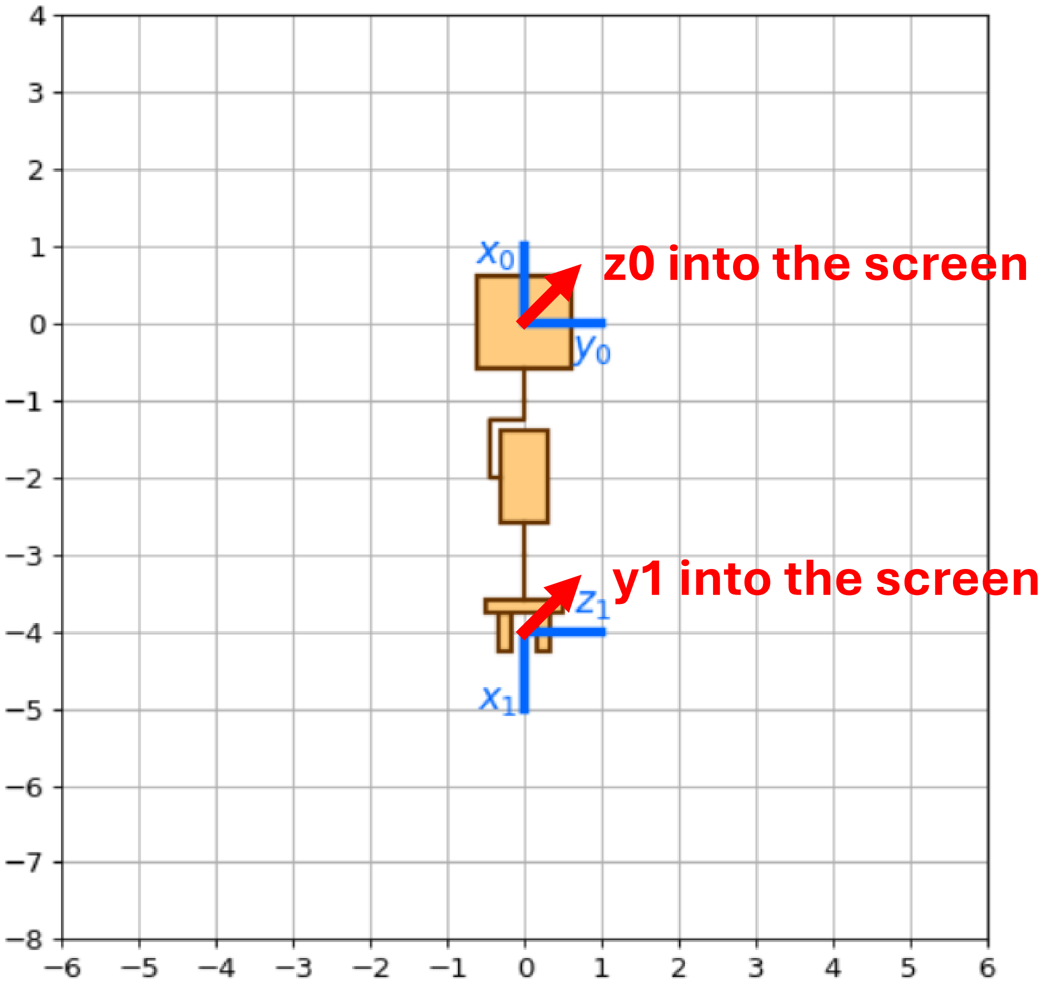
1 DoF Revolute Robot Example

1. Identify the orientations of the fixed and tool frames with the right-hand rule



1 DoF Revolute Robot Example

2. Create the homogeneous transformation M of the tool frame in the fixed frame



x1 is pointing in –x0 direction
y1 is pointing in z0 direction
z1 is pointing in y0 direction

Tool frame is translated by -4 in x0 direction

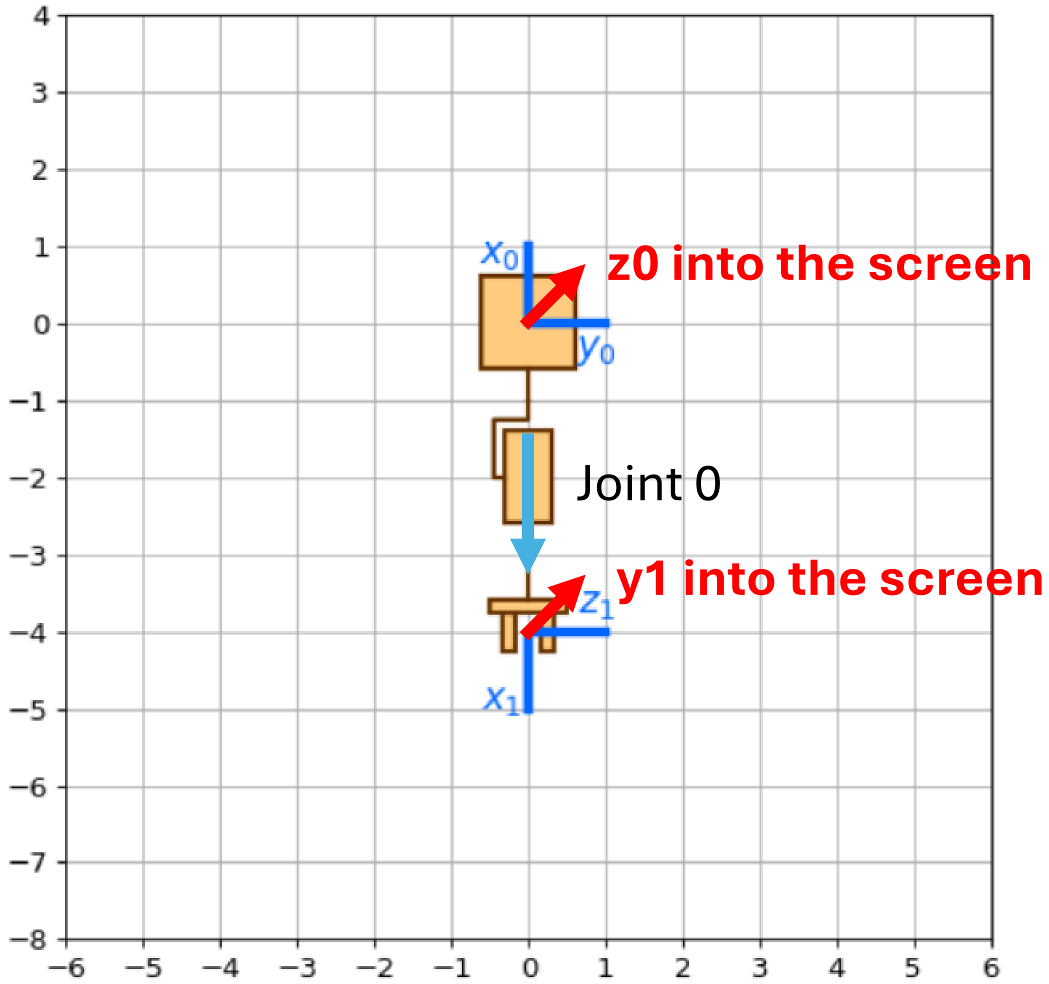
```
R = np.array([[-1, 0, 0],
              [0 , 0, 1],
              [0 , 1, 0]])
```

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

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

1 DoF Revolute Robot Example

3. Build the screw axis for each joint

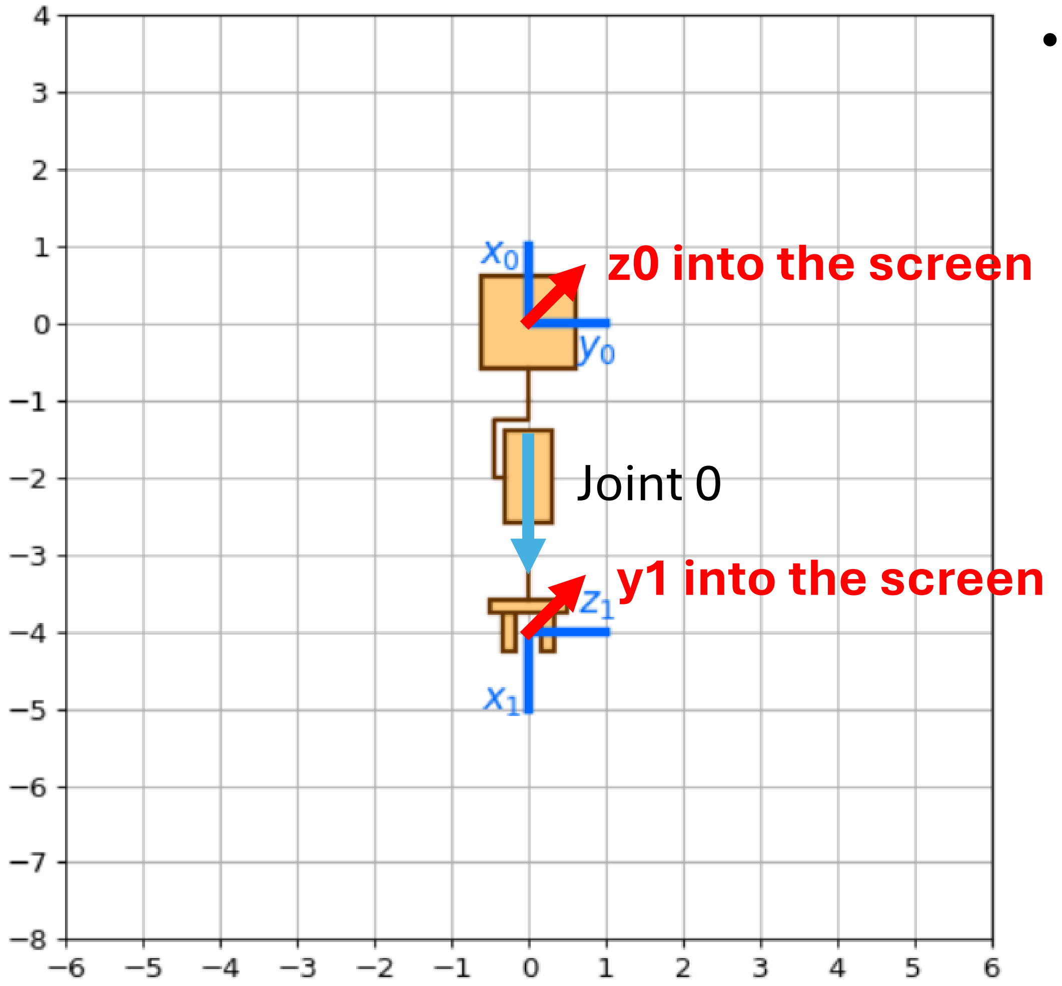


- Joint 0
  - revolute
  - S
    - w = [-1,0,0]
    - q = [-2,0,0]
    - v = -w x q = [0,0,0]
    - S = [w v] = [-1,0,0,0,0,0]
  - B
    - w = [1,0,0]
    - q = [-2,0,0]
    - v = -w x q = [0,0,0]
    - B = [w v] = [1,0,0,0,0,0]

1 DoF Revolute Robot Example

4. Build the matrix form of the screw axis per joint

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$



- Joint 0
  - S = [w v] = [-1,0,0,0,0,0]

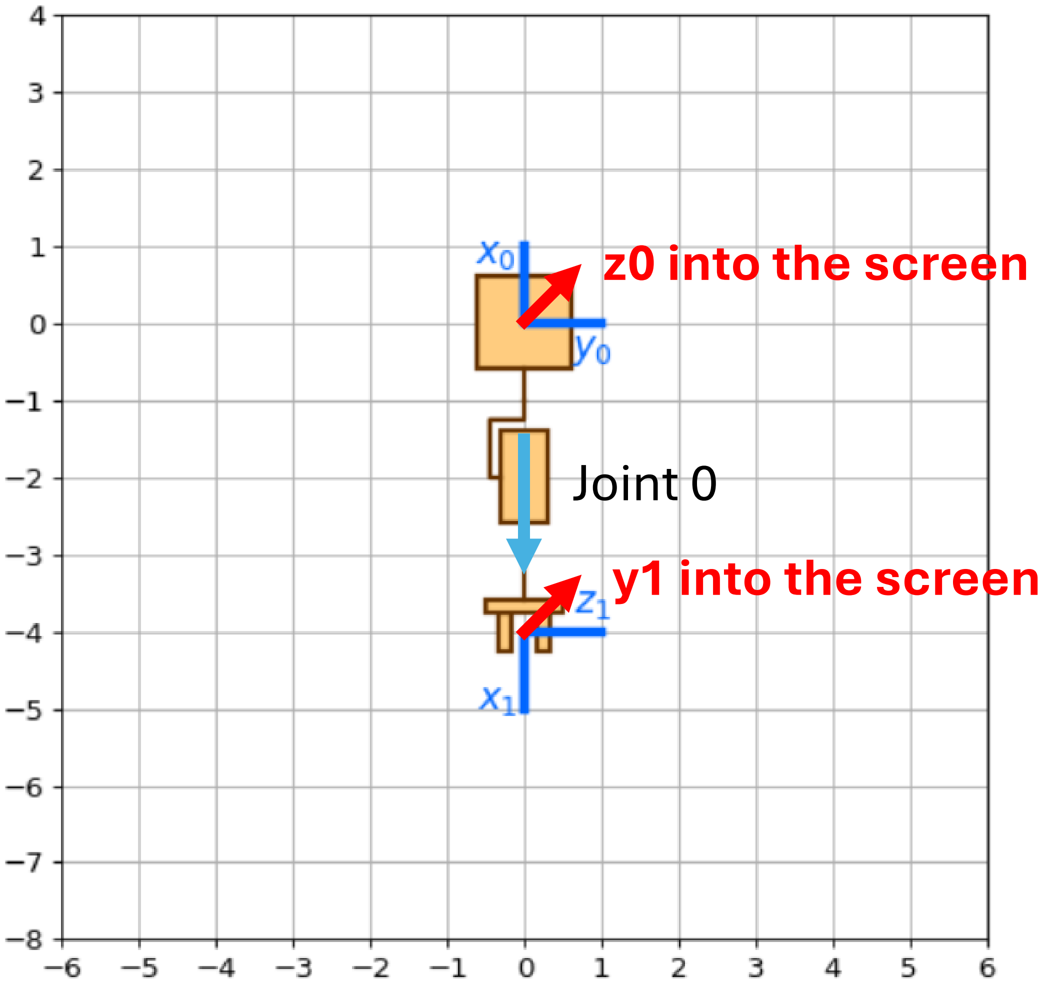
```
[[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  0]
 [ 0  0  0  0]]
```

  - B = [w v] = [1,0,0,0,0,0]

```
[[ 0  0  0  0]
 [ 0  0 -1  0]
 [ 0  1  0  0]
 [ 0  0  0  0]]
```

1 DoF Revolute Robot Example

5. Apply product of exponentials formula



- Fixed frame conventions

$$T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \dots e^{[S_n]\theta_n} M$$

```
T_1_in_0_theta_s = expm(mat_s_0 * theta_0) @ M
```

```
[[-1.          0.          0.         -4.        ]
 [ 0.          0.43496553  0.9004471   0.        ]
 [ 0.          0.9004471  -0.43496553  0.        ]
 [ 0.          0.          0.          1.        ]]
```

- Body frame conventions

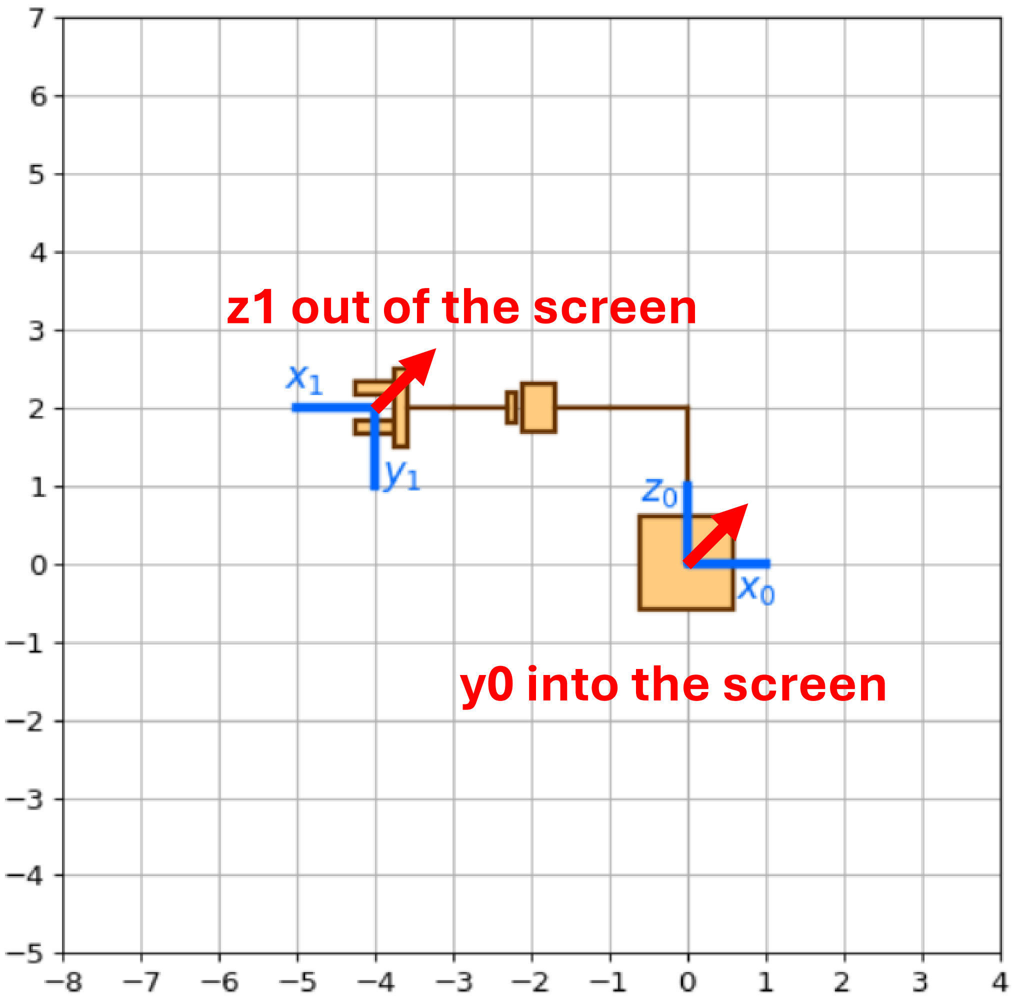
$$T(\theta) = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} \dots e^{[B_n]\theta_n}$$

```
T_1_in_0_theta_b = M @ expm(mat_b_0 * theta_0)
```

```
[[-1.          0.          0.         -4.        ]
 [ 0.          0.43496553  0.9004471   0.        ]
 [ 0.          0.9004471  -0.43496553  0.        ]
 [ 0.          0.          0.          1.        ]]
```

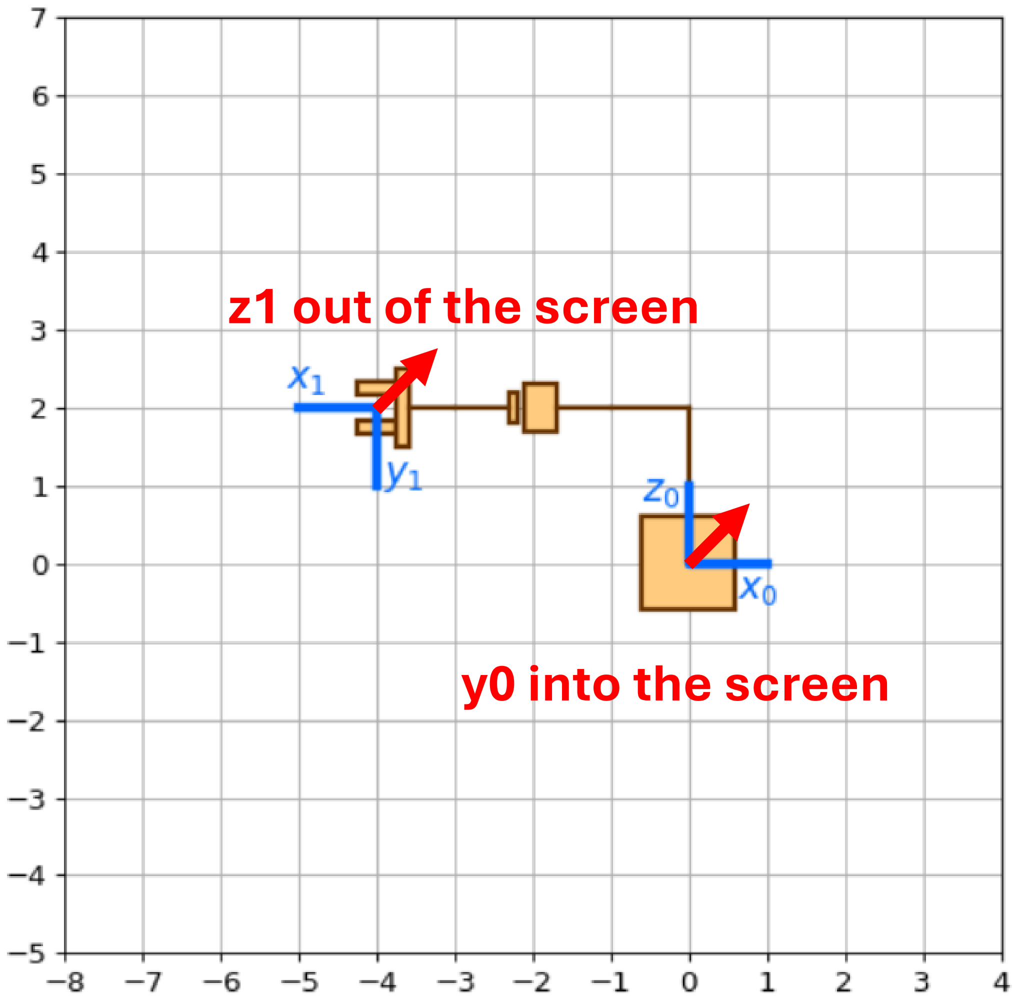
1 DoF Prismatic Robot Example

1. Identify the orientations of the fixed and tool frames with the right-hand rule



1 DoF Prismatic Robot Example

2. Create the homogeneous transformation M of the tool frame in the fixed frame



x1 is pointing in –x0 direction
y1 is pointing in –z0 direction
z1 is pointing in –y0 direction

Tool frame is translated by -4 in x0 direction and 2 in z0 direction

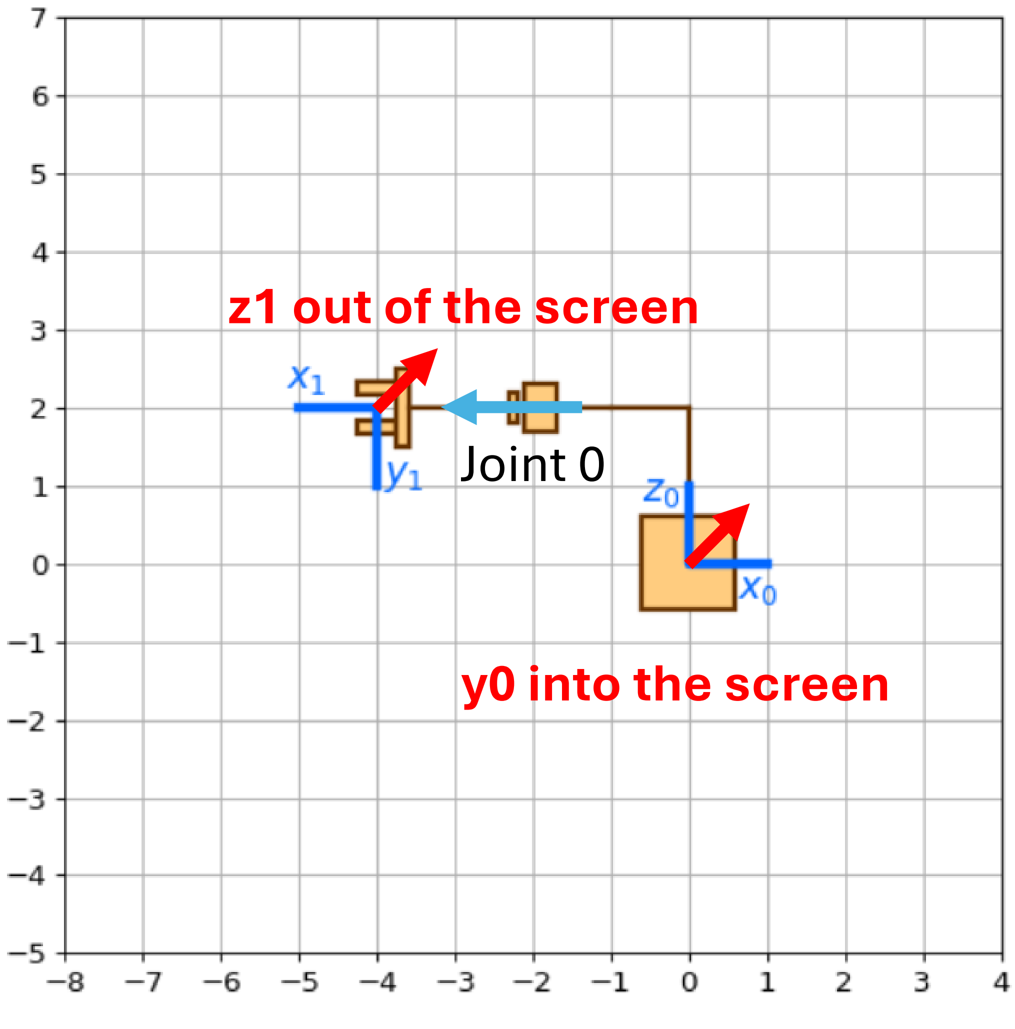
```
R = np.array([[-1, 0 , 0 ],
              [0 , 0 , -1],
              [0 , -1, 0 ]])
```

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

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

1 DoF Prismatic Robot Example

3. Build the screw axis for each joint

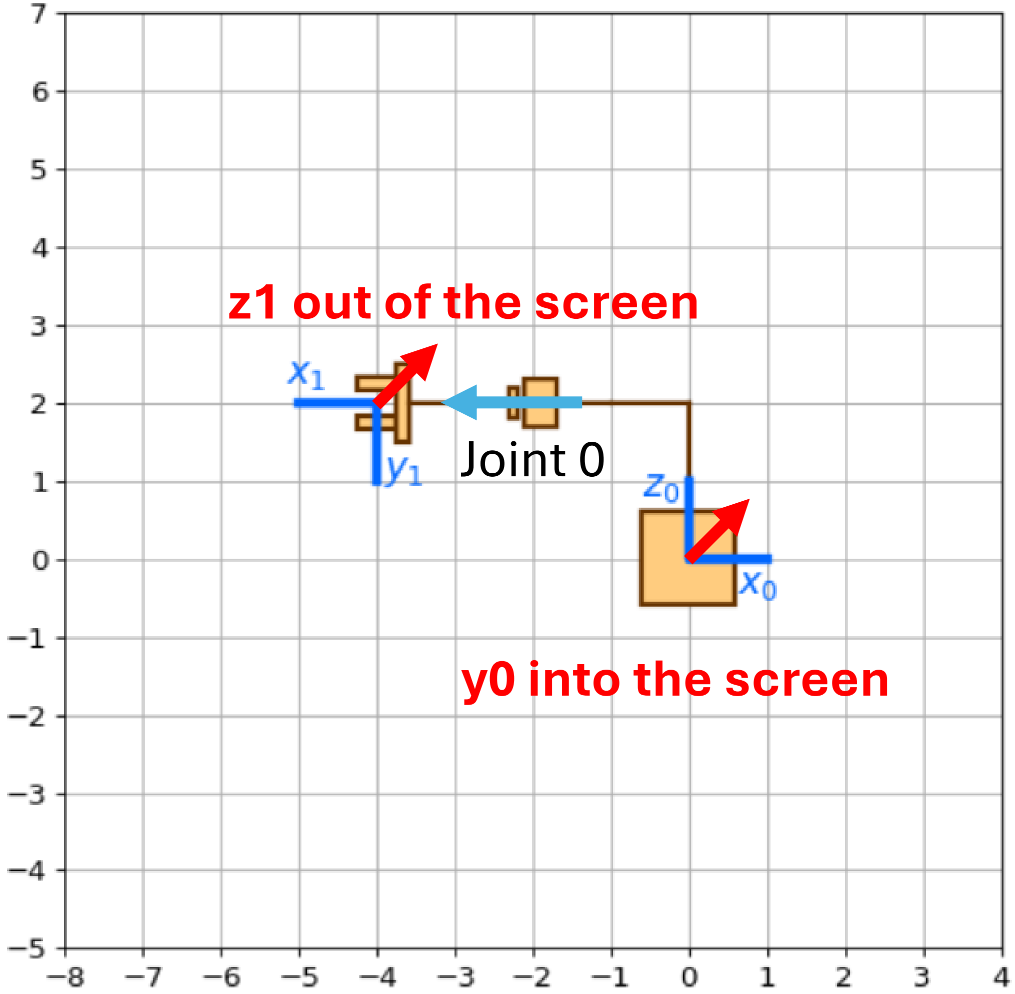


- Joint 0
  - prismatic
  - S
    - w = [0,0,0]
    - v = [-1,0,0]
    - S = [w v] = [0,0,0,-1,0,0]
  - B
    - w = [0,0,0]
    - v = [1,0,0]
    - B = [w v] = [0,0,0,1,0,0]

1 DoF Prismatic Robot Example

4. Build the matrix form of the screw axis per joint

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$



- Joint 0
  - S = [w v] = [0,0,0,-1,0,0]

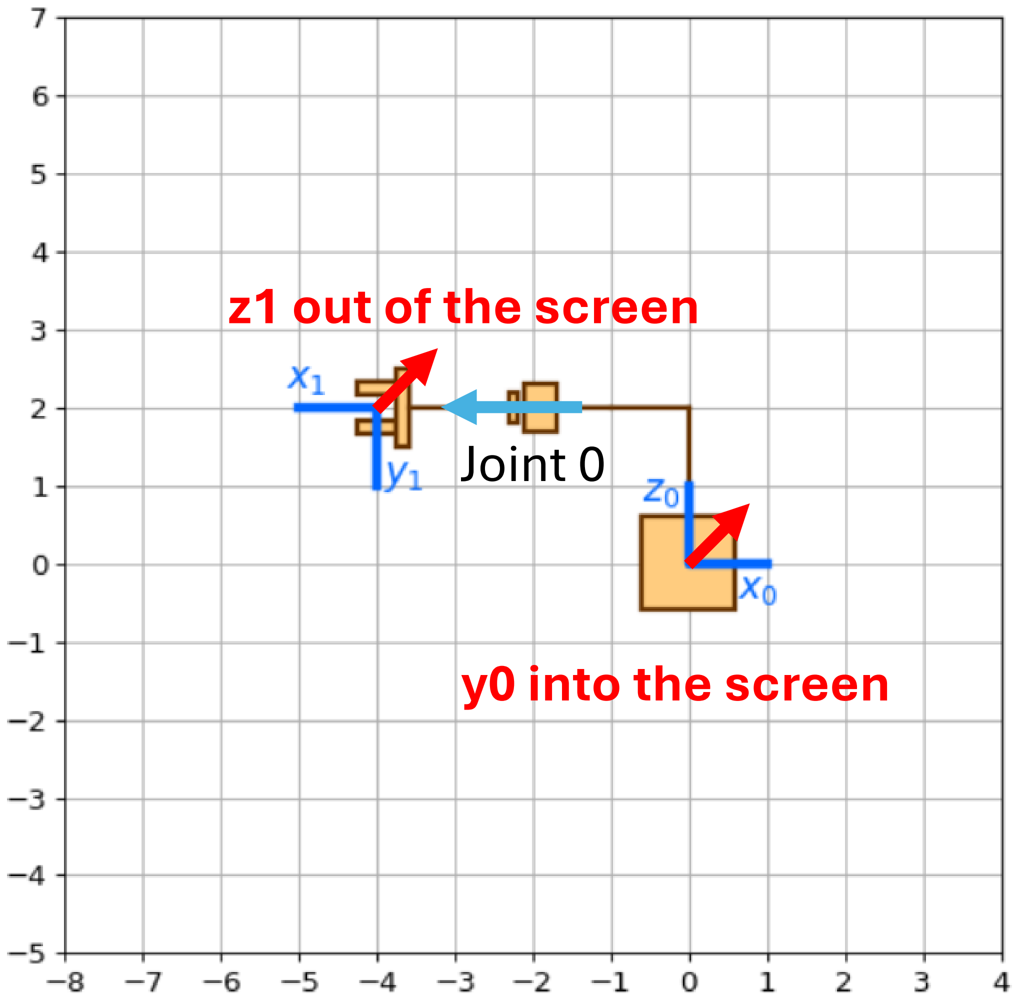
```
[[ 0  0  0 -1]
 [ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

  - B = [w v] = [0,0,0,1,0,0]

```
[[0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

1 DoF Prismatic Robot Example

5. Apply product of exponentials formula



- Fixed frame conventions

$$T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \dots e^{[S_n]\theta_n} M$$

```
T_1_in_0_theta_s = expm(mat_s_0 * theta_0) @ M
```

```
[[-1.    0.    0.   -4.45]
 [ 0.    0.   -1.    0.  ]
 [ 0.   -1.    0.    2.  ]
 [ 0.    0.    0.    1.  ]]
```

- Body frame conventions

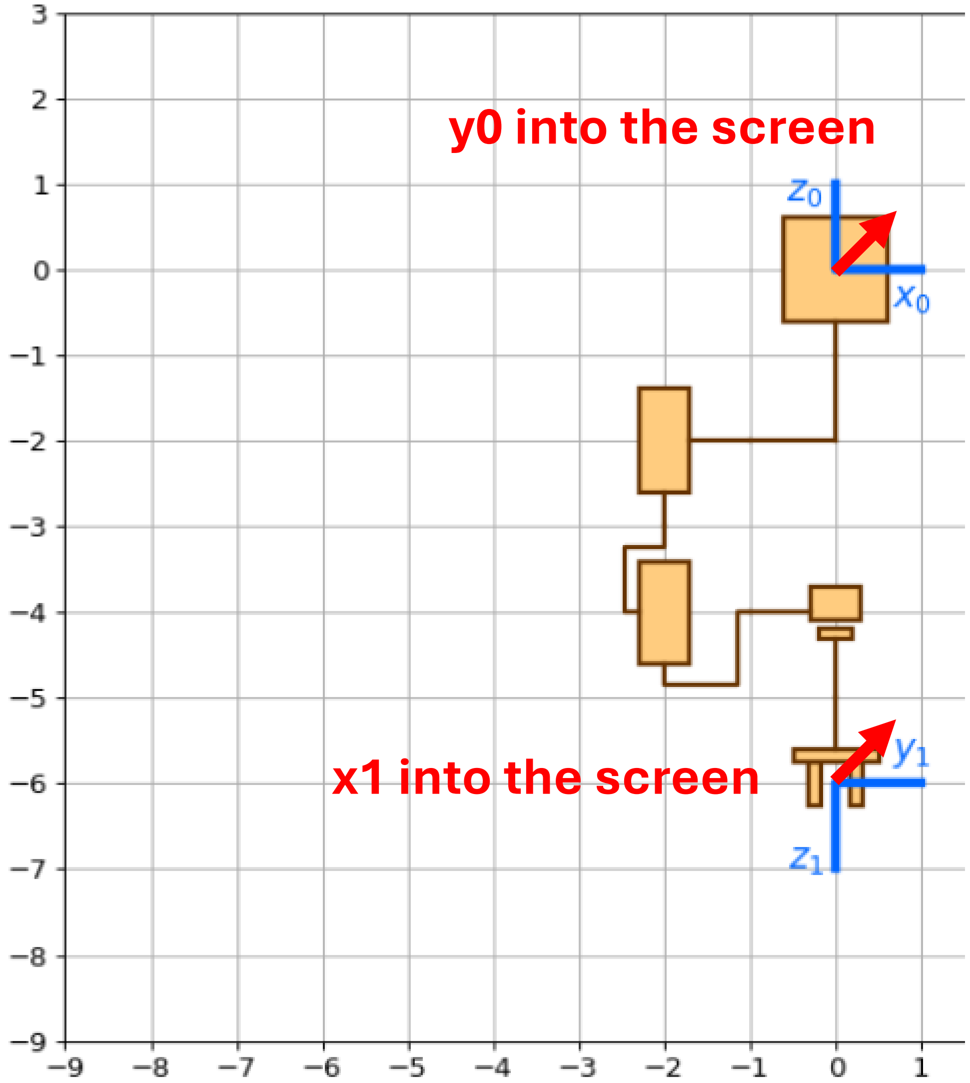
$$T(\theta) = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} \dots e^{[B_n]\theta_n}$$

```
T_1_in_0_theta_b = M @ expm(mat_b_0 * theta_0)
```

```
[[-1.    0.    0.   -4.45]
 [ 0.    0.   -1.    0.  ]
 [ 0.   -1.    0.    2.  ]
 [ 0.    0.    0.    1.  ]]
```

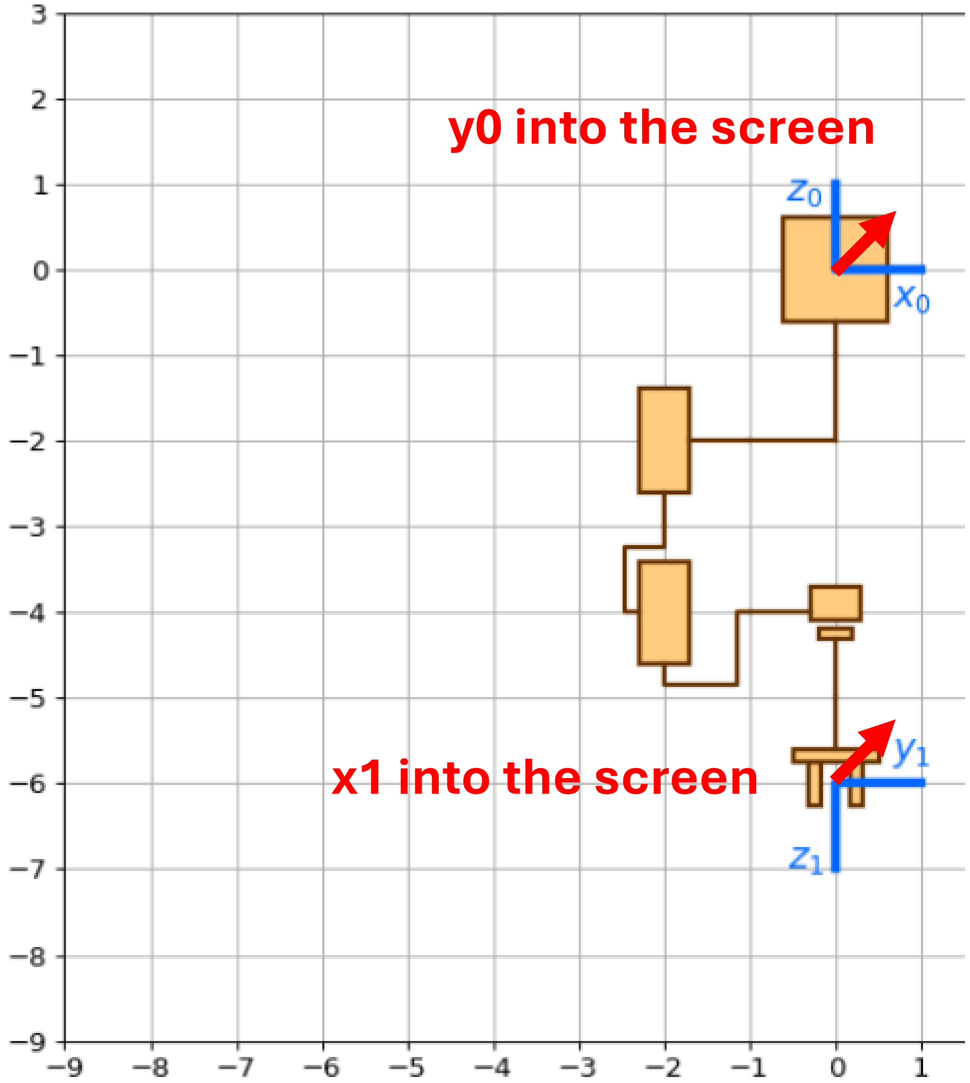
3 DoF RRP Robot Example

1. Identify the orientations of the fixed and tool frames with the right-hand rule



3 DoF RRP Robot Example

2. Create the homogeneous transformation M of the tool frame in the fixed frame



x1 is pointing in y0 direction
y1 is pointing in x0 direction
z1 is pointing in –z0 direction

Tool frame is translated by -6 in z0 direction

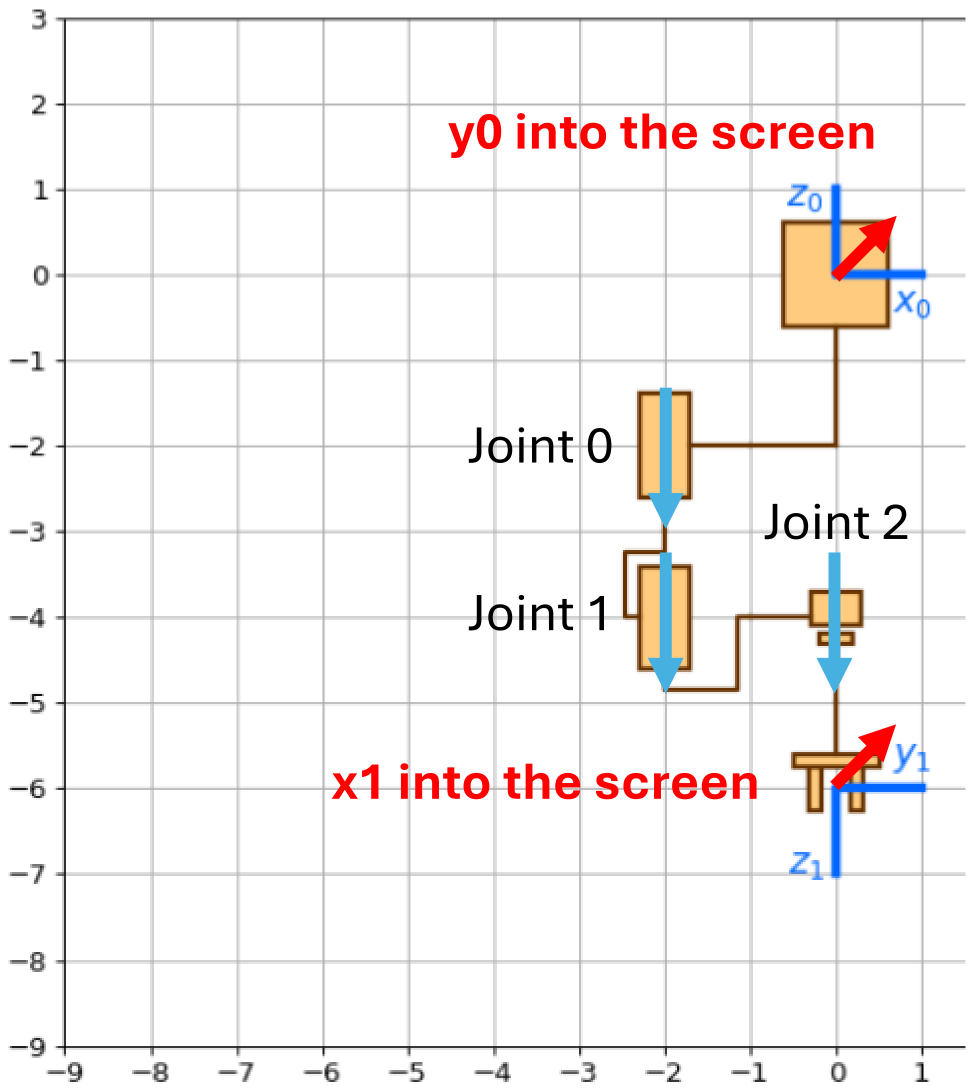
```
R = np.array([[0, 1, 0 ],
              [1, 0, 0 ],
              [0, 0, -1]])
```

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

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

3 DoF RRP Robot Example

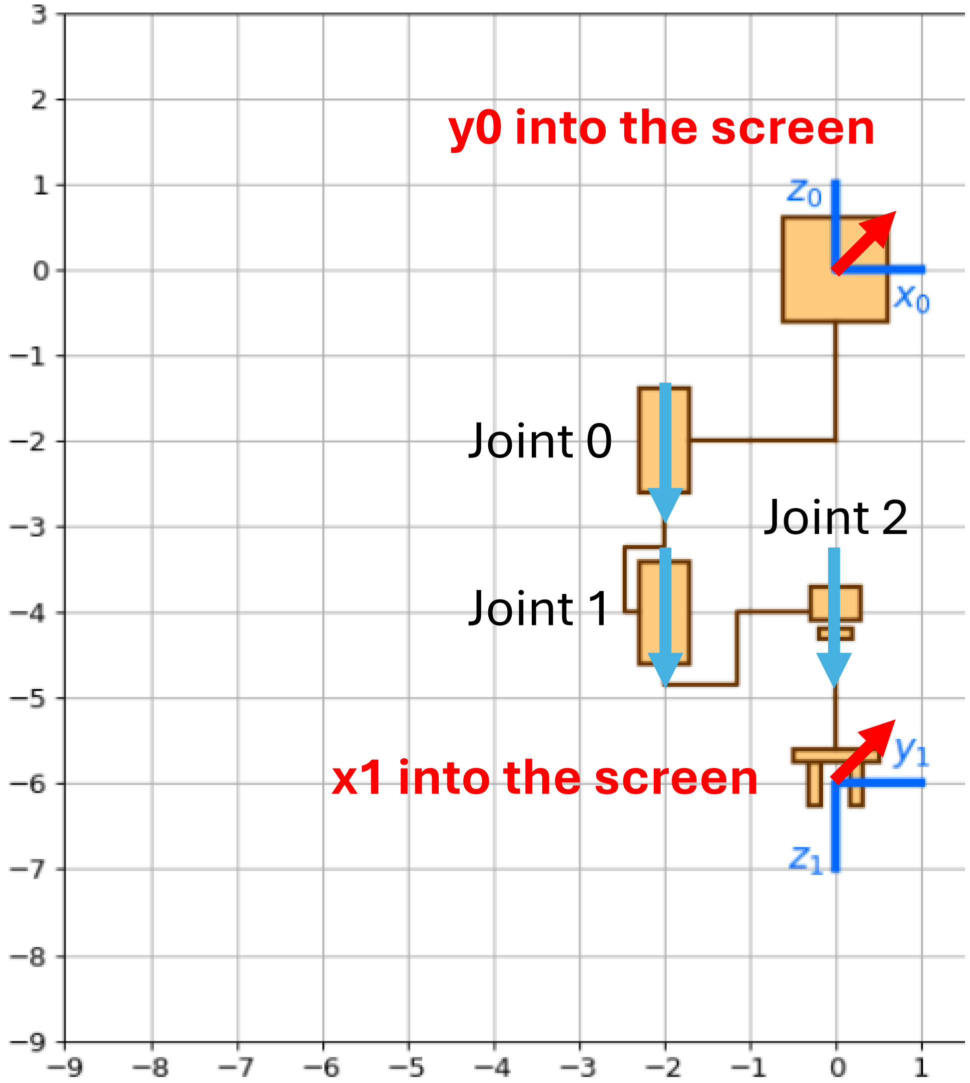
3. Build the screw axis for each joint



- Joint 0
  - revolute
  - S
    - w = [0,0,-1]
    - q = [-2,0,-2]
    - v = -w x q = [0,-2,0]
    - S = [w v] = [0,0,-1,0,-2,0]
  - B
    - w = [0,0,1]
    - q = [0,-2,-4]
    - v = -w x q = [-2,0,0]
    - B = [w v] = [0,0,1,-2,0,0]

3 DoF RRP Robot Example

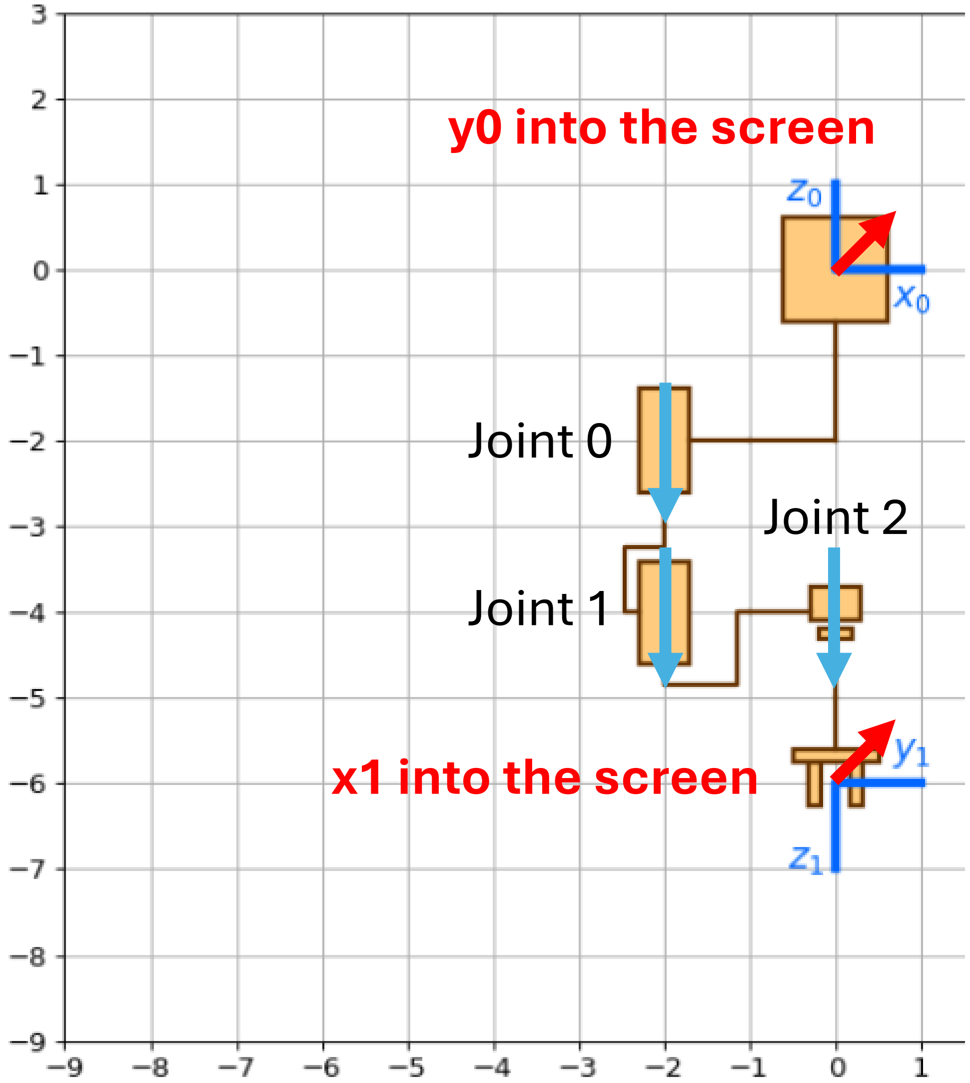
3. Build the screw axis for each joint



- Joint 1
  - revolute
  - S
    - w = [0,0,-1]
    - q = [-2,0,-4]
    - v = -w x q = [0,-2,0]
    - S = [w v] = [0,0,-1,0,-2,0]
  - B
    - w = [0,0,1]
    - q = [0,-2,-2]
    - v = -w x q = [-2,0,0]
    - B = [w v] = [0,0,1,-2,0,0]

3 DoF RRP Robot Example

3. Build the screw axis for each joint

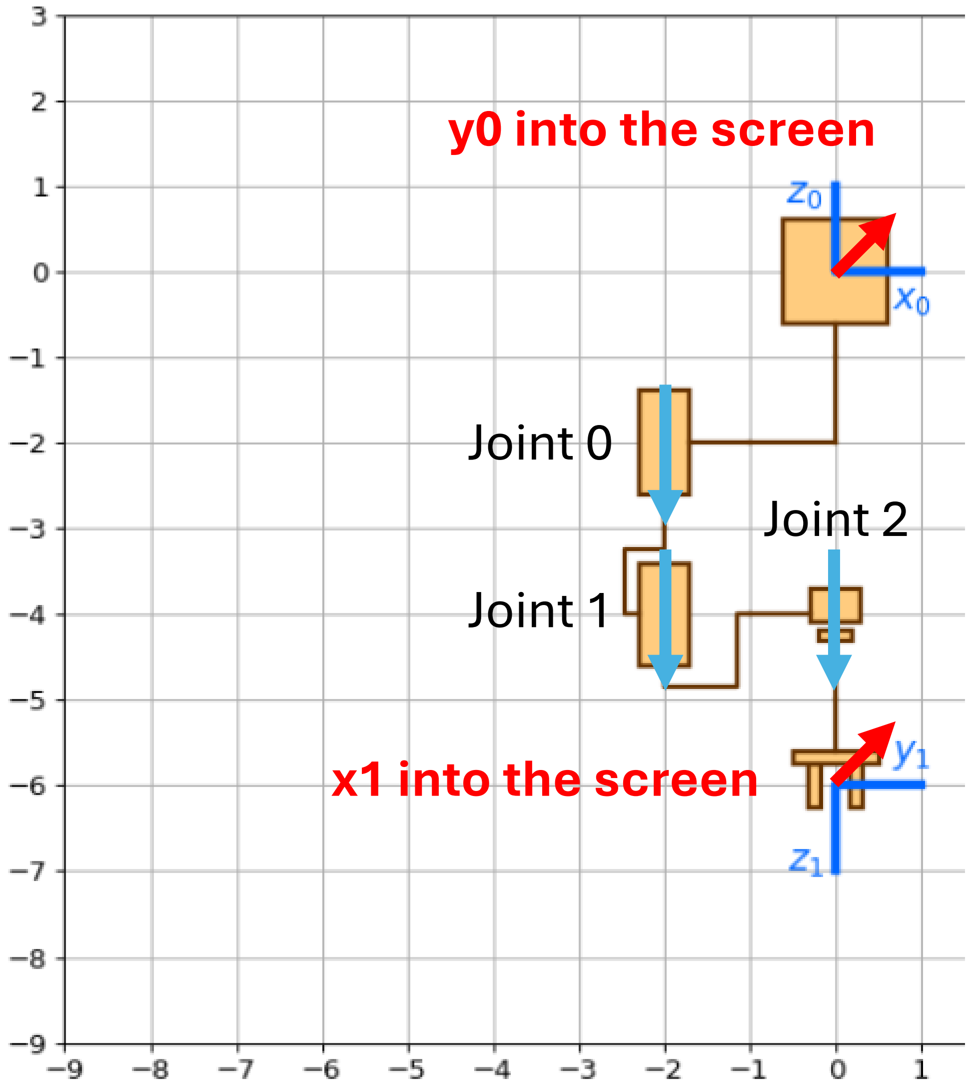


- Joint 2
  - prismatic
  - S
    - w = [0,0,0]
    - v = [0,0,-1]
    - S = [w v] = [0,0,0,0,0,-1]
  - B
    - w = [0,0,0]
    - v = [0,0,1]
    - B = [w v] = [0,0,0,0,0,1]

3 DoF RRP Robot Example

4. Build the matrix form of the screw axis per joint

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$



- Joint 0
  - S = [w v] = [0,0,-1,0,-2,0]

```
[[ 0  1  0  0]
 [-1  0  0 -2]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

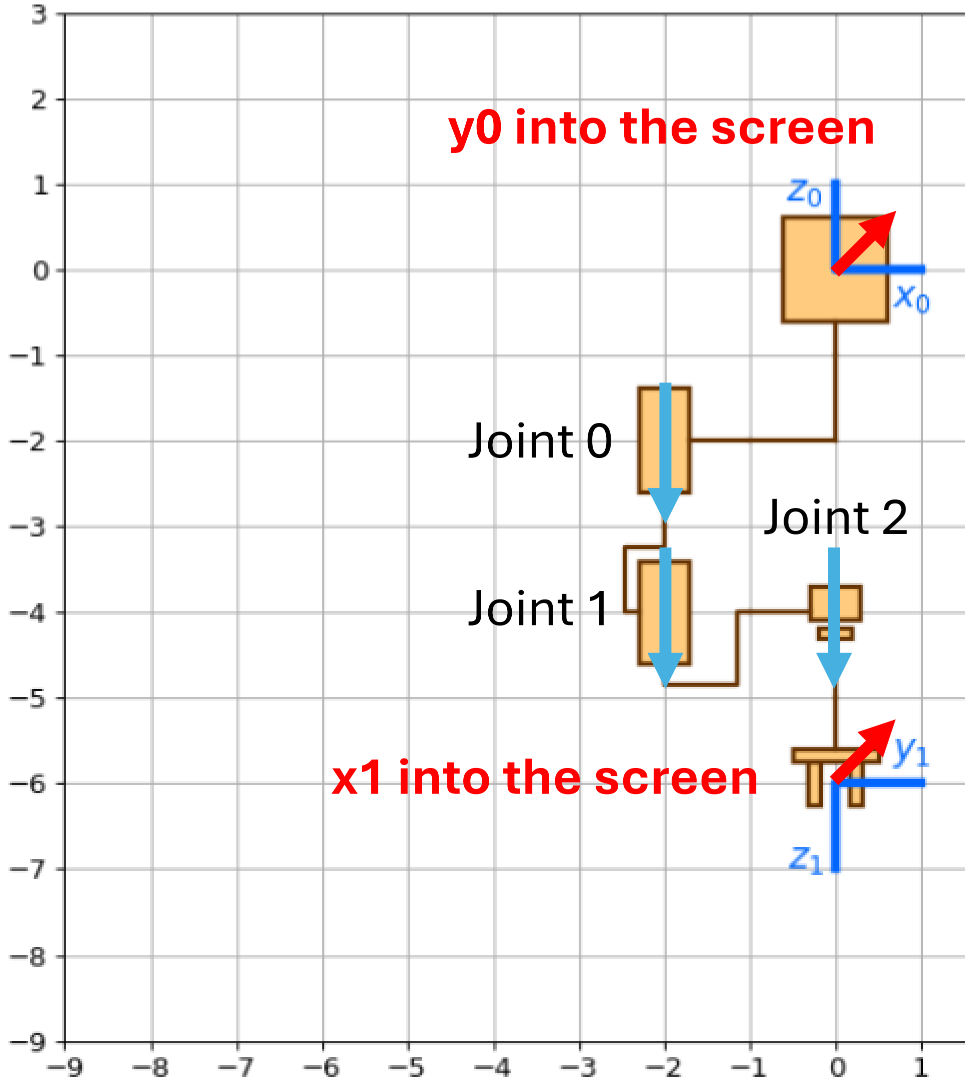
  - B = [w v] = [0,0,1,-2,0,0]

```
[[ 0 -1  0 -2]
 [ 1  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

3 DoF RRP Robot Example

4. Build the matrix form of the screw axis per joint

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$



- Joint 1
  - S = [w v] = [0,0,-1,0,-2,0]

```
[[ 0  1  0  0]
 [-1  0  0 -2]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

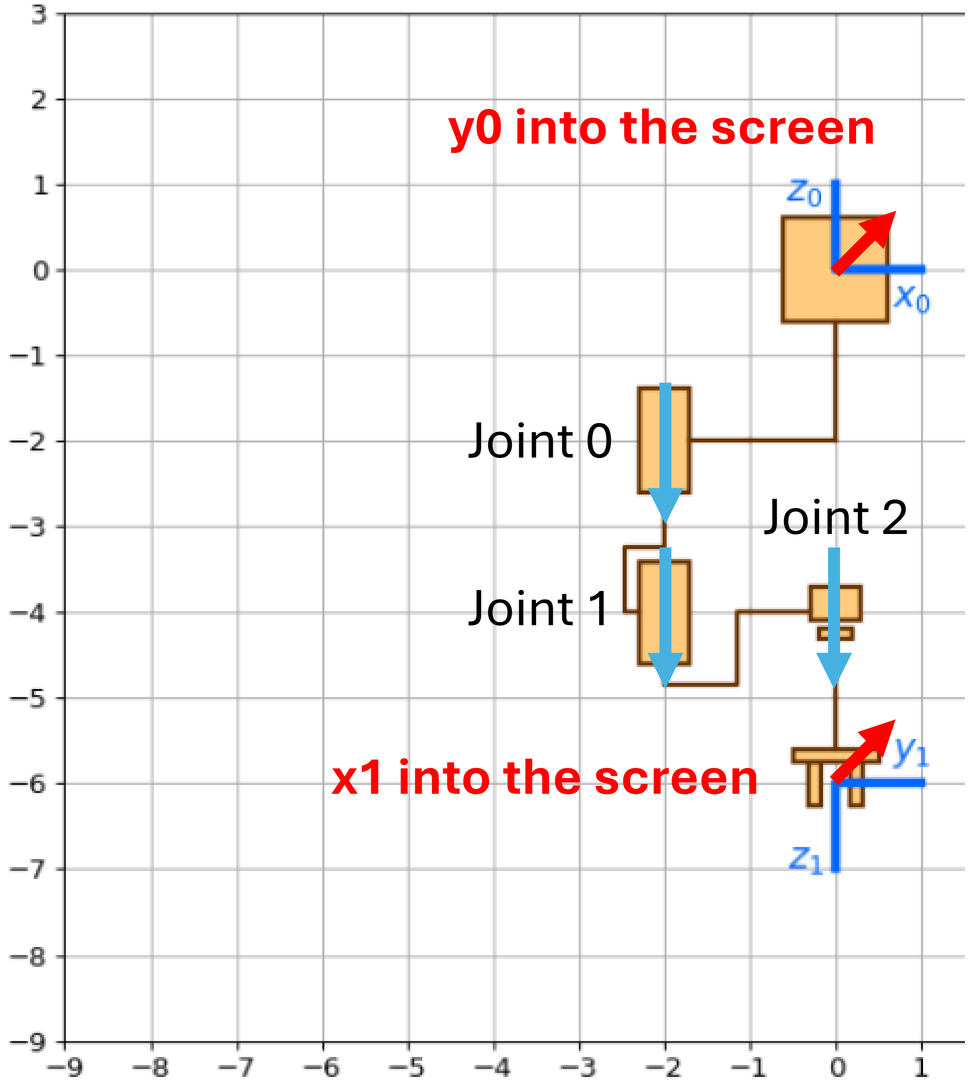
  - B = [w v] = [0,0,1,-2,0,0]

```
[[ 0 -1  0 -2]
 [ 1  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

3 DoF RRP Robot Example

4. Build the matrix form of the screw axis per joint

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$



- Joint 2
  - S = [w v] = [0,0,0,0,0,-1]

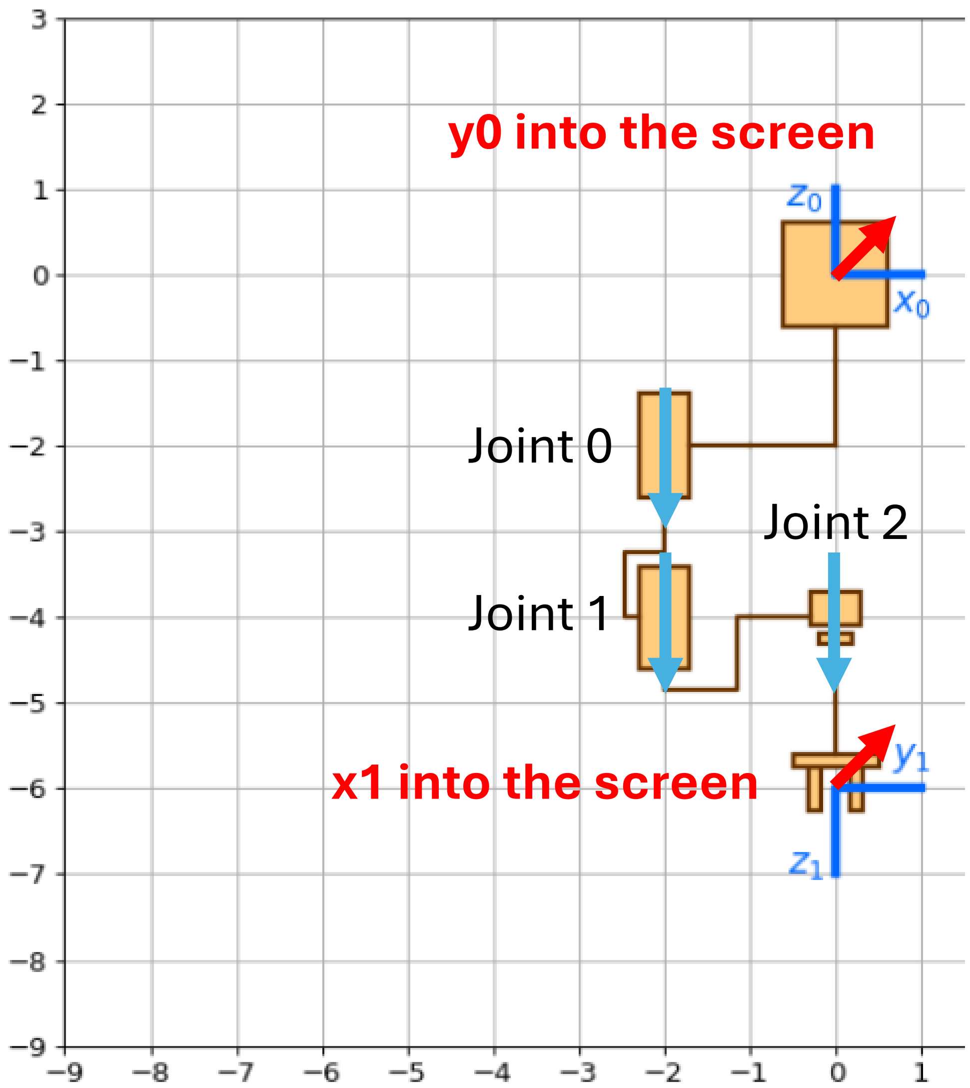
```
[[ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0 -1]
 [ 0  0  0  0]]
```

  - B = [w v] = [0,0,0,0,0,1]

```
[[ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0 -1]
 [ 0  0  0  0]]
```

3 DoF RRP Robot Example

5. Apply product of exponentials formula



- Fixed frame conventions

$$T(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \dots e^{[S_n]\theta_n} M$$

```
[[ 0.68163876  0.73168887  0.         -0.53662226]
 [ 0.73168887 -0.68163876  0.         -1.36327752]
 [ 0.          0.         -1.         -6.75      ]
 [ 0.          0.          0.          1.        ]]
```

- Body frame conventions

$$T(\theta) = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} \dots e^{[B_n]\theta_n}$$

```
[[ 0.68163876  0.73168887  0.         -0.53662226]
 [ 0.73168887 -0.68163876  0.         -1.36327752]
 [ 0.          0.         -1.         -6.75      ]
 [ 0.          0.          0.          1.        ]]
```

Velocity Kinematics

Forward Kinematics

Calculate the position of the end-effector of an open chain given joint angles

$$\underbrace{x(t)}_{\text{pose of tool}} = f(\underbrace{\theta(t)}_{\text{joint angles}})$$

f → FK map / $T(\theta)$ / PoE

Velocity Kinematics

Calculate the velocity (twist!) of the end-effector frame

$$\dot{x} = \frac{d}{dt} x(t) = \underbrace{\frac{\partial f}{\partial \theta}(\theta)}_{J(\theta)} \cdot \dot{\theta}$$

$J(\theta) \rightarrow$ Jacobian

Definition 5.1. Let the forward kinematics of an n-link open chain be expressed in the following product of exponentials form:

$$T = e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_n]\theta_n} M. \tag{5.9}$$

The **space Jacobian** $J_s(\theta) \in \mathbb{R}^{6\times n}$ relates the joint rate vector $\dot{\theta} \in \mathbb{R}^n$ to the spatial twist $\mathcal{V}_s$ via

$$\mathcal{V}_s = J_s(\theta)\dot{\theta}. \tag{5.10}$$

The ith column of $J_s(\theta)$ is

$$J_{si}(\theta) = \mathrm{Ad}_{e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_{i-1}]\theta_{i-1}}}(\mathcal{S}_i), \tag{5.11}$$

for $i = 2, \ldots, n$, with the first column $J_{s1} = \mathcal{S}_1$.

On the formula sheet …

Velocity Kinematics

Space Jacobian

$$\mathcal{V}_s = J_s(\theta)\dot{\theta} = \begin{bmatrix} J_{s,1} & J_{s,2} & \cdots & J_{s,n} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}$$

$$J_{s,1} = S_1 \quad J_{s,2} = Ad_{e^{[S_1]\theta_1}}(S_2) \quad \cdots \quad J_{s,n} = Ad_{e^{[S_1]\theta_1} \ldots e^{[S_{n-1}]\theta_{n-1}}}(S_n)$$

Algorithm for Building the Jacobian

1. Identify the orientations of the fixed frame with the right-hand rule

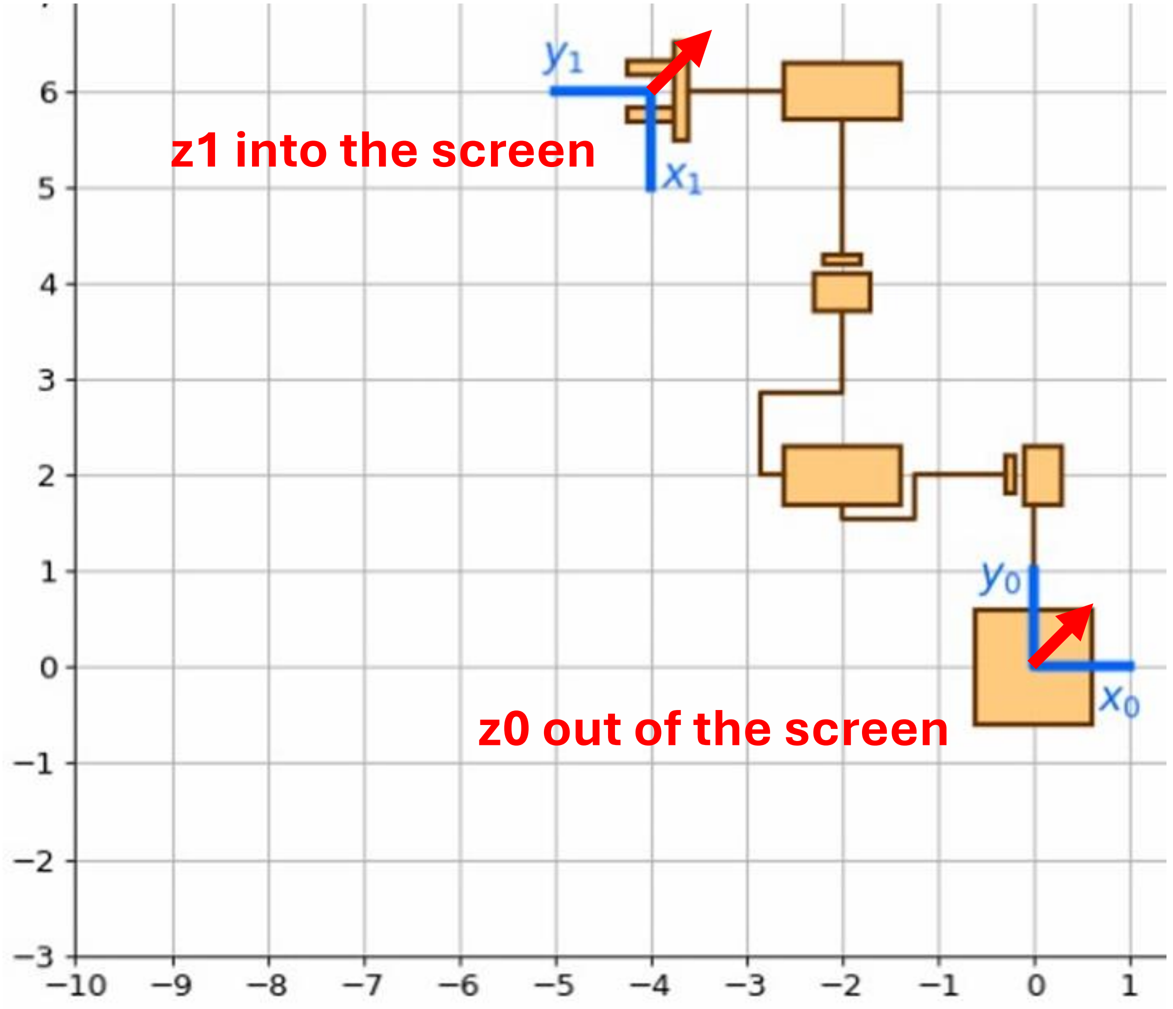
2. Build the screw axis for each joint
   - Ask yourself if we want space Jacobian or the body Jacobian
   - If we specifically ask for Js -> fixed frame screw
   - If we specifically ask for Jb -> body frame screw
     - I prefer to solve for Js and then use the Adj map to convert to Jb

3. For each column i of Js
   - Solve for the PoE up to joint i – 1
   - Compute its adjoint
   - Multiply the adjoint by the screw axis of joint i

4. If we want Jb, $J_b = \left[Ad_{T_{bs}}\right] J_s$ $\quad [Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}$

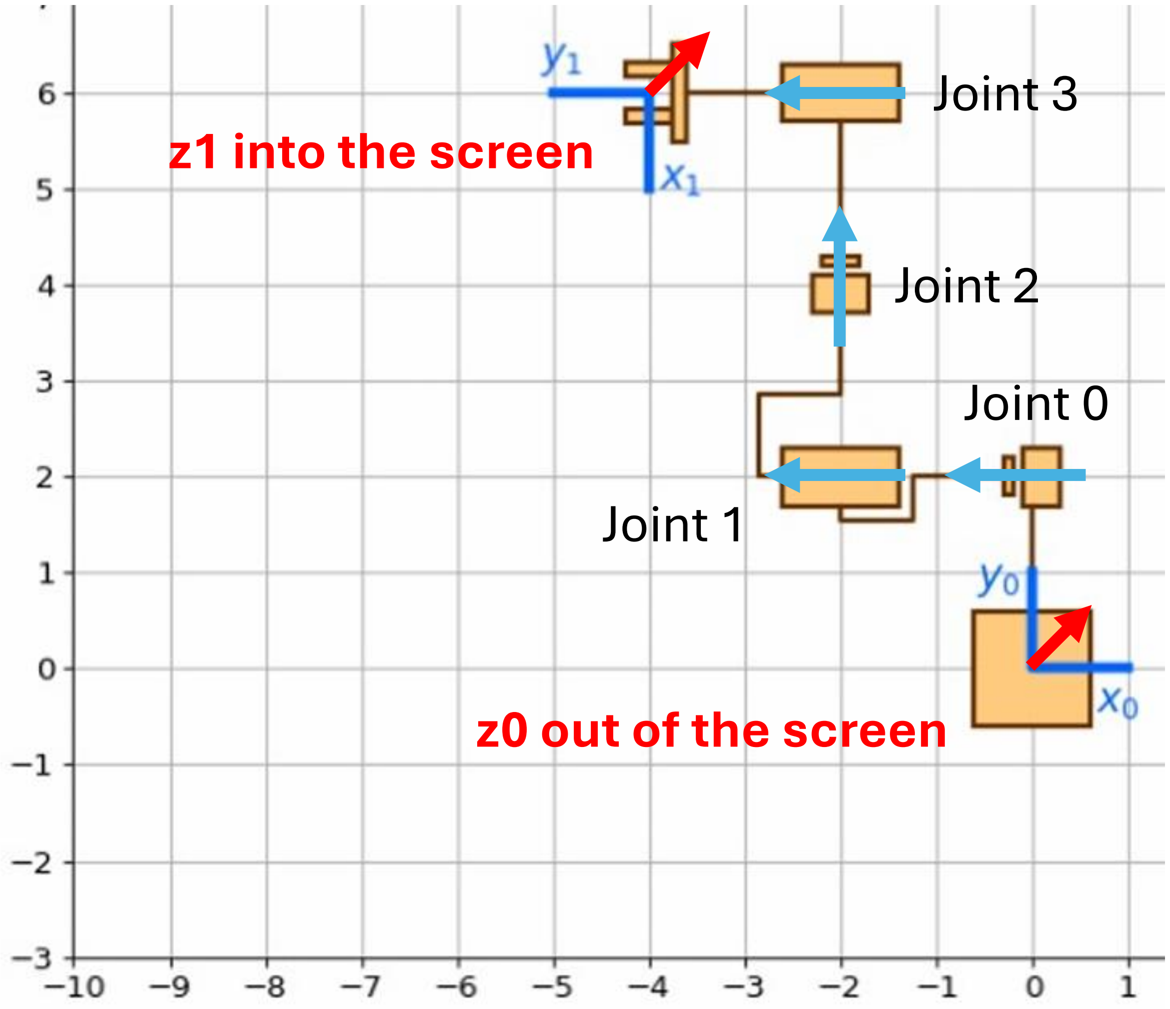
4 DoF PRPR Robot Example

1. Identify the orientations of the fixed and tool frames with the right-hand rule



4 DoF PRPR Robot Example

2. Build the screw axis (and matrix form) for each joint

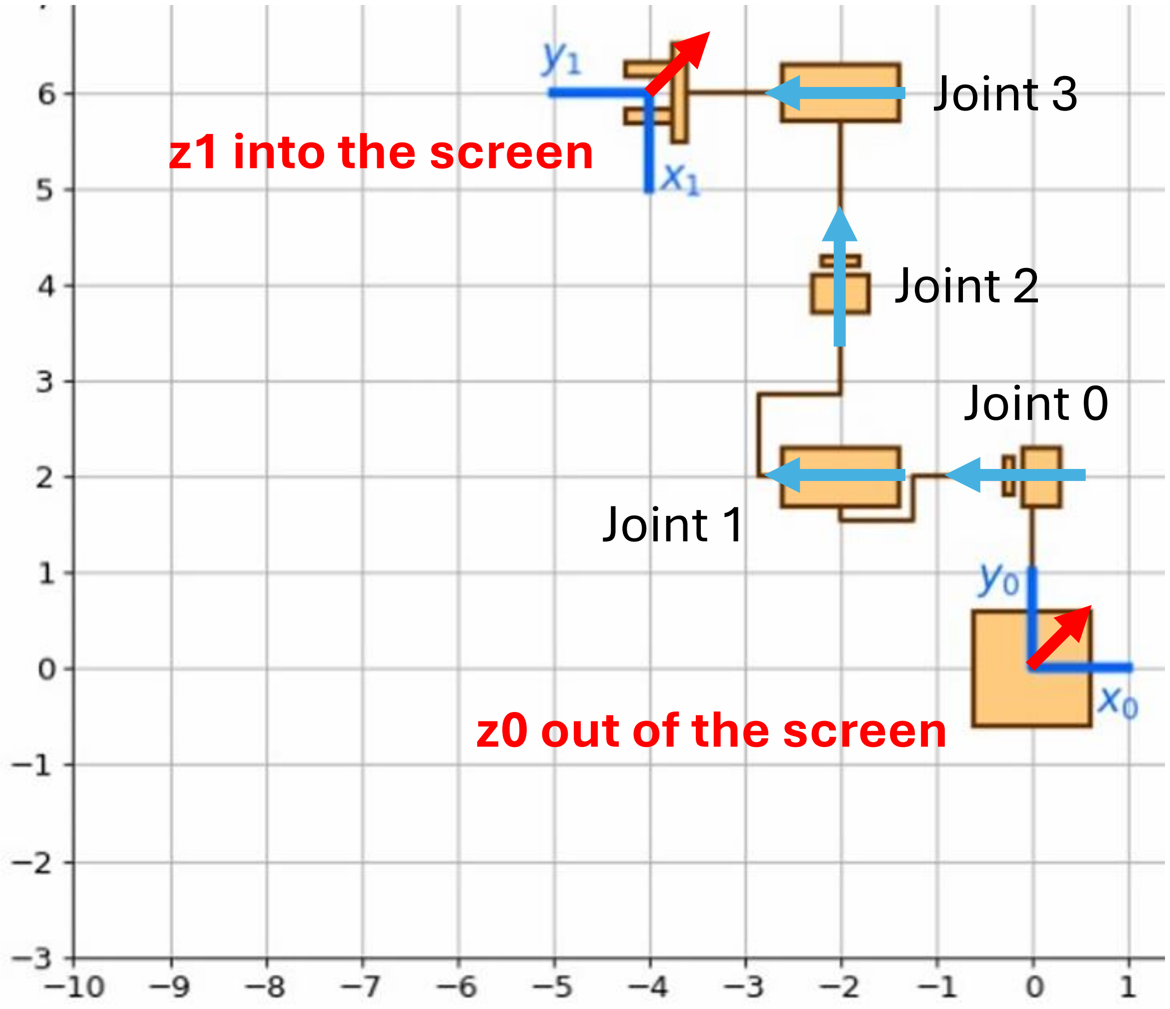


- Joint 0
  - prismatic
  - S0
    - w = [0,0,0]
    - v = [-1,0,0]
    - S = [w v] = [0,0,0,-1,0,0]
  - [S0]

```
[[ 0  0  0 -1]
 [ 0  0  0  0]
 [ 0  0  0  0]
 [ 0  0  0  0]]
```

4 DoF PRPR Robot Example

2. Build the screw axis (and matrix form) for each joint

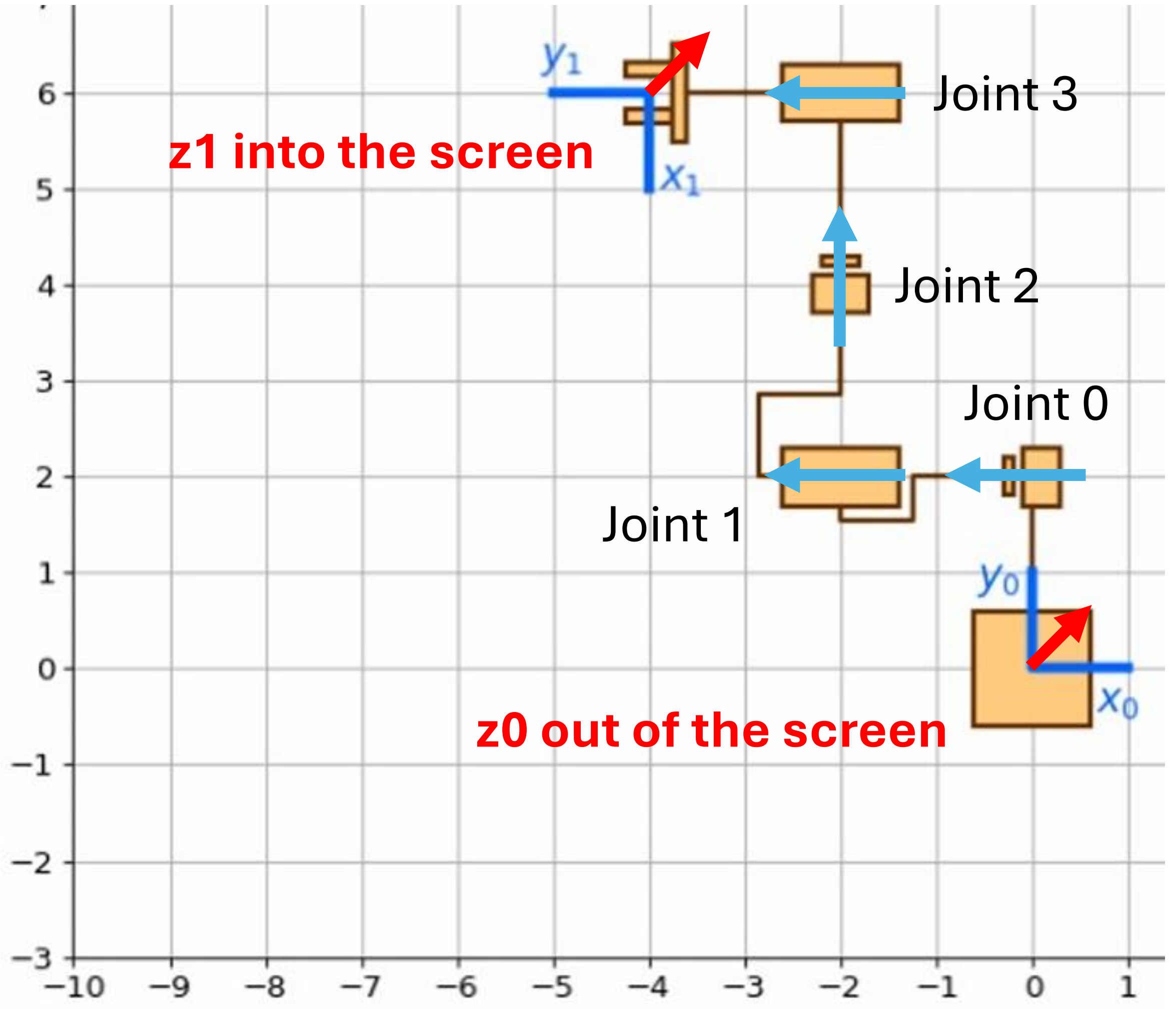


- Joint 1
  - revolute
  - S1
    - w = [-1,0,0]
    - q = [-2,2,0]
    - v = -w x q = [0,0,2]
    - S = [w v] = [-1,0,0,0,0,2]
  - [S1]

```
[[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  2]
 [ 0  0  0  0]]
```

4 DoF PRPR Robot Example

2. Build the screw axis (and matrix form) for each joint

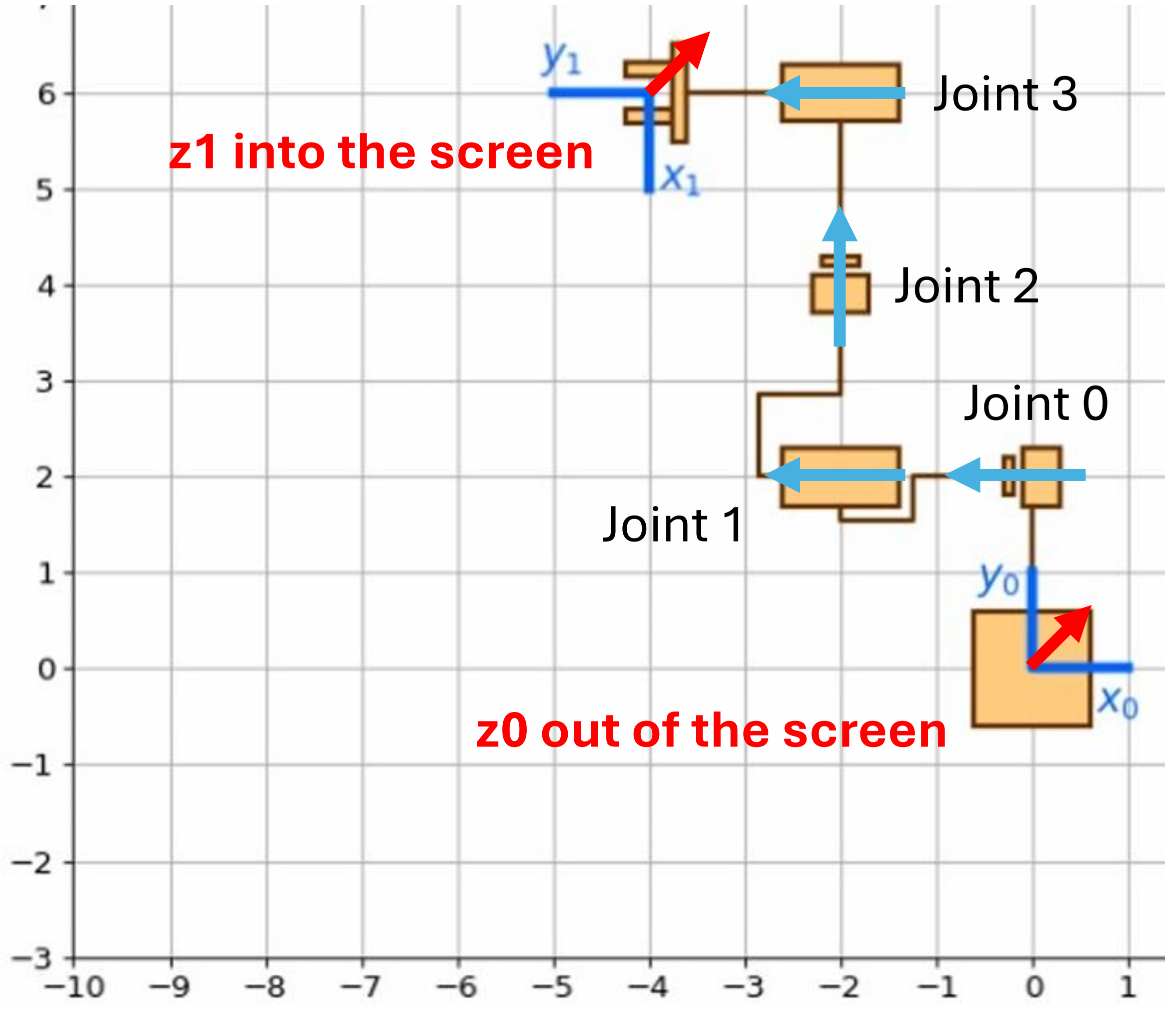


- Joint 2
  - prismatic
  - S2
    - w = [0,0,0]
    - v = [0,1,0]
    - S = [w v] = [0,0,0,0,1,0]
  - [S2]

```
[[0 0 0 0]
 [0 0 0 1]
 [0 0 0 0]
 [0 0 0 0]]
```

4 DoF PRPR Robot Example

2. Build the screw axis (and matrix form) for each joint

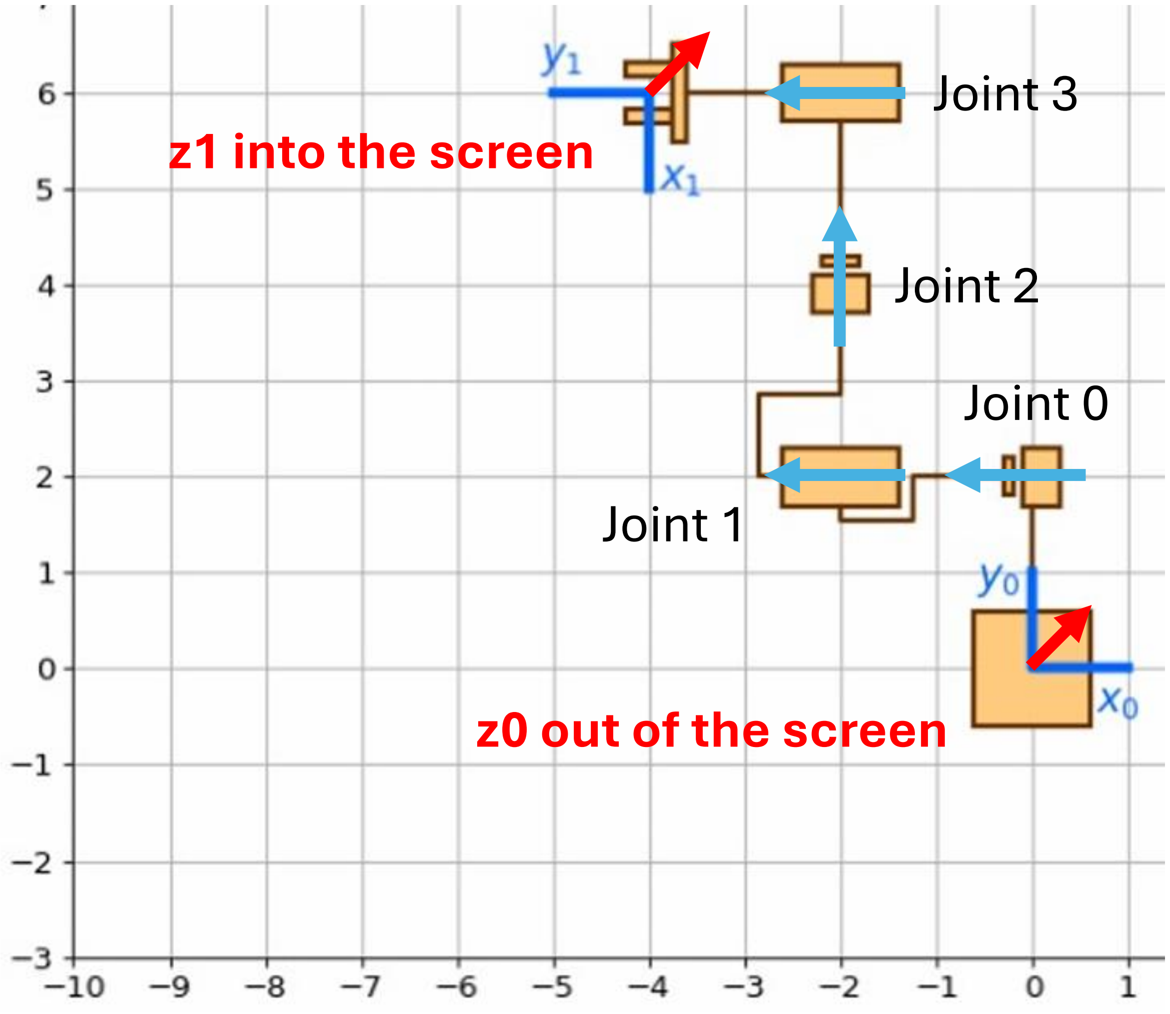


- Joint 3
  - revolute
  - S3
    - w = [-1,0,0]
    - q = [-2,6,0]
    - v = -w x q = [0,0,6]
    - S = [w v] = [-1,0,0,0,0,6]
  - [S3]

```
[[ 0  0  0  0]
 [ 0  0  1  0]
 [ 0 -1  0  6]
 [ 0  0  0  0]]
```

4 DoF PRPR Robot Example

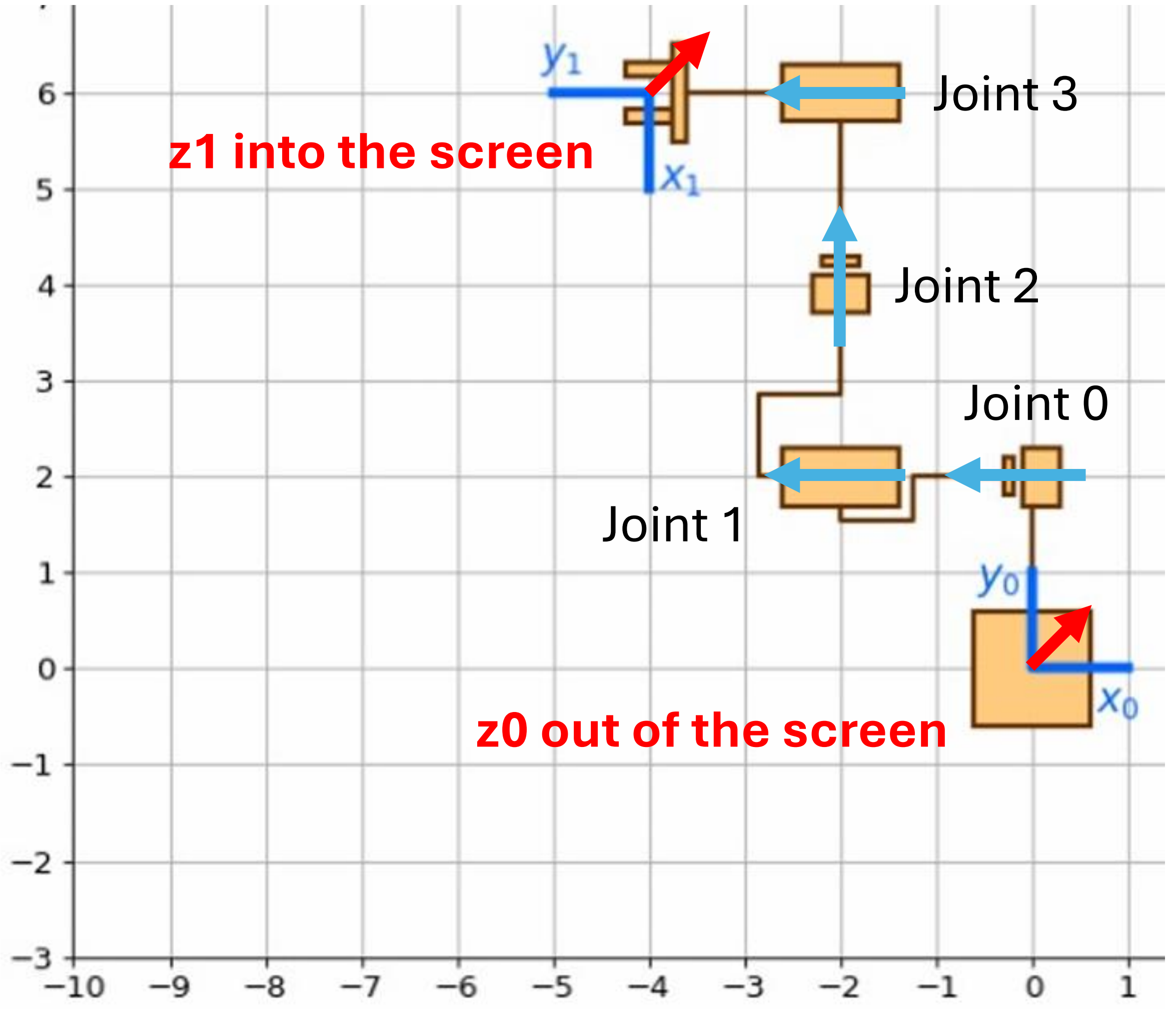
3. Fill in the columns of the Jacobian



- Column 0
  - S0 = [0,0,0,-1,0,0]
  - J0 = [0,0,0,-1,0,0]

4 DoF PRPR Robot Example

3. Fill in the columns of the Jacobian



- Column 1
  - S1 = [-1,0,0,0,0,2]
  - J1 = [Ad_T1] S1

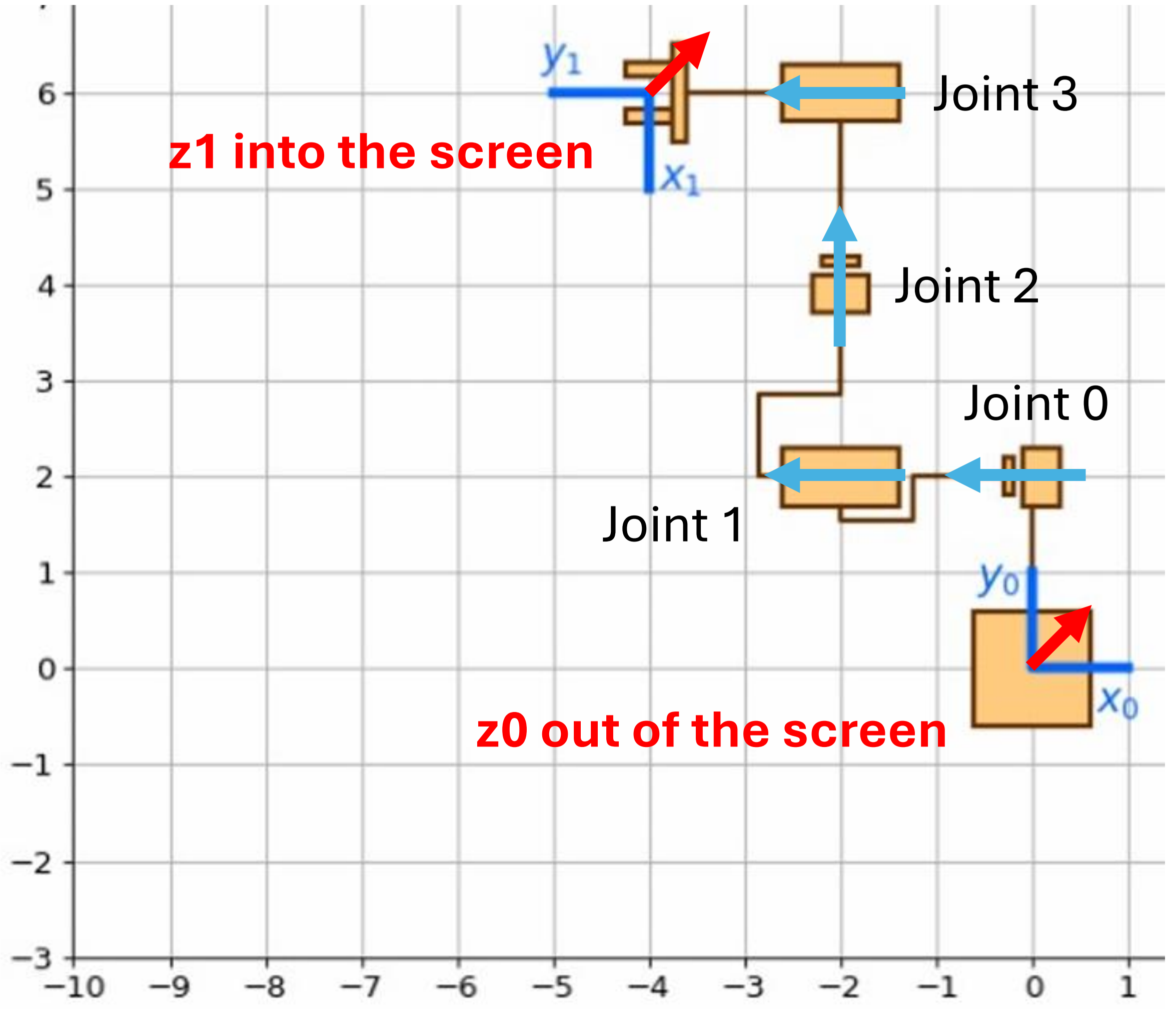
```
Ad_exp_s_0_theta_0 = adjoint(expm(mat_s_0 * theta_0))
J_s_1 = Ad_exp_s_0_theta_0 @ s_1
```

$$[Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}$$

```
[[-1.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 0.]
 [ 2.]]
```

4 DoF PRPR Robot Example

3. Fill in the columns of the Jacobian



- Column 2
  - S2 = [0,0,0,0,1,0]
  - J2 = [Ad_T2] S2

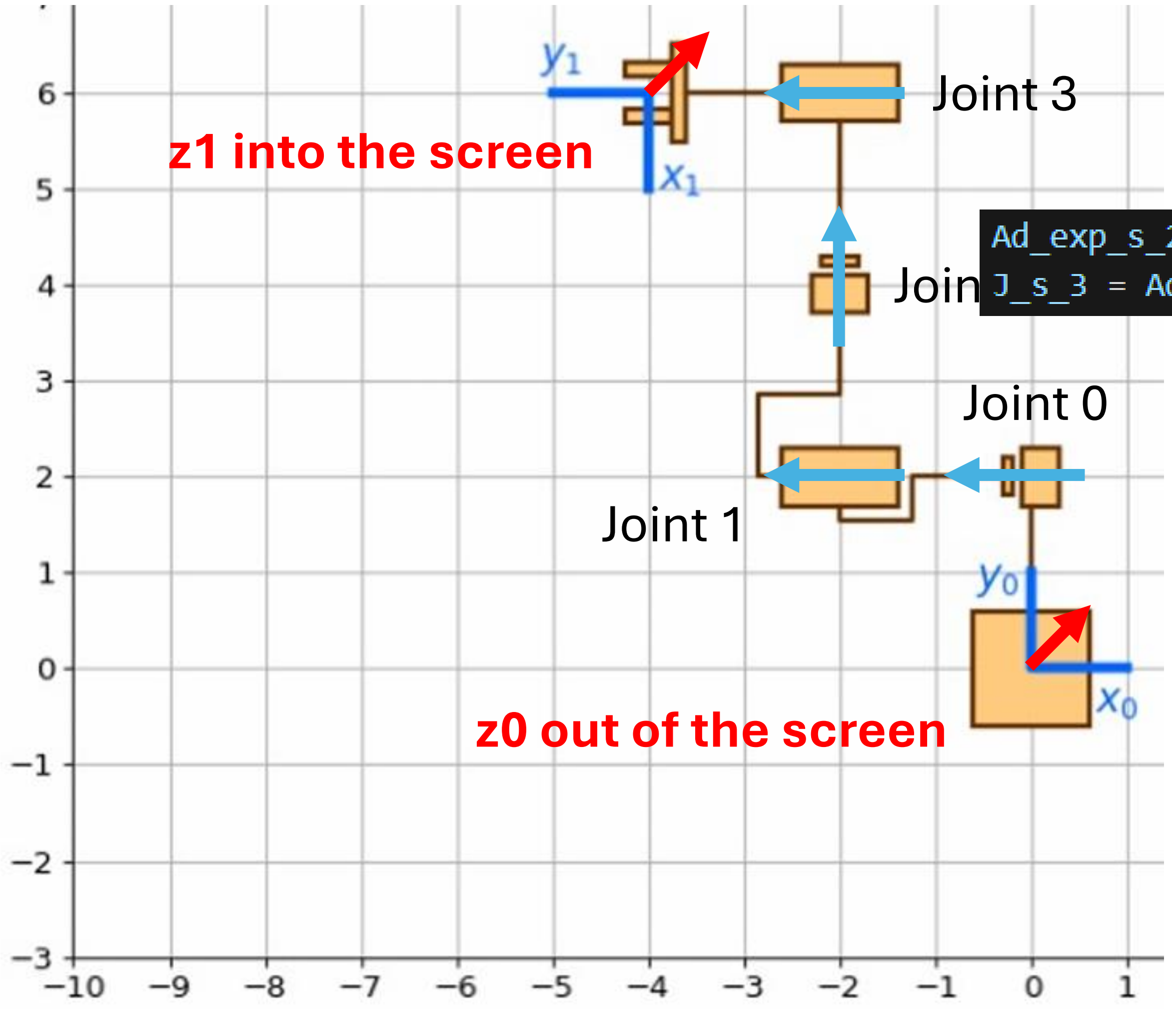
```
Ad_exp_s_1_theta_1 = adjoint(expm(mat_s_1 * theta_1))
J_s_2 = Ad_exp_s_0_theta_0 @ Ad_exp_s_1_theta_1 @ s_2
```

$$[Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}$$

```
[[ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.87758256]
 [-0.47942554]]
```

4 DoF PRPR Robot Example

3. Fill in the columns of the Jacobian



- Column 3
  - S3 = [-1,0,0,0,0,6]
  - J3 = [Ad_T3] S3

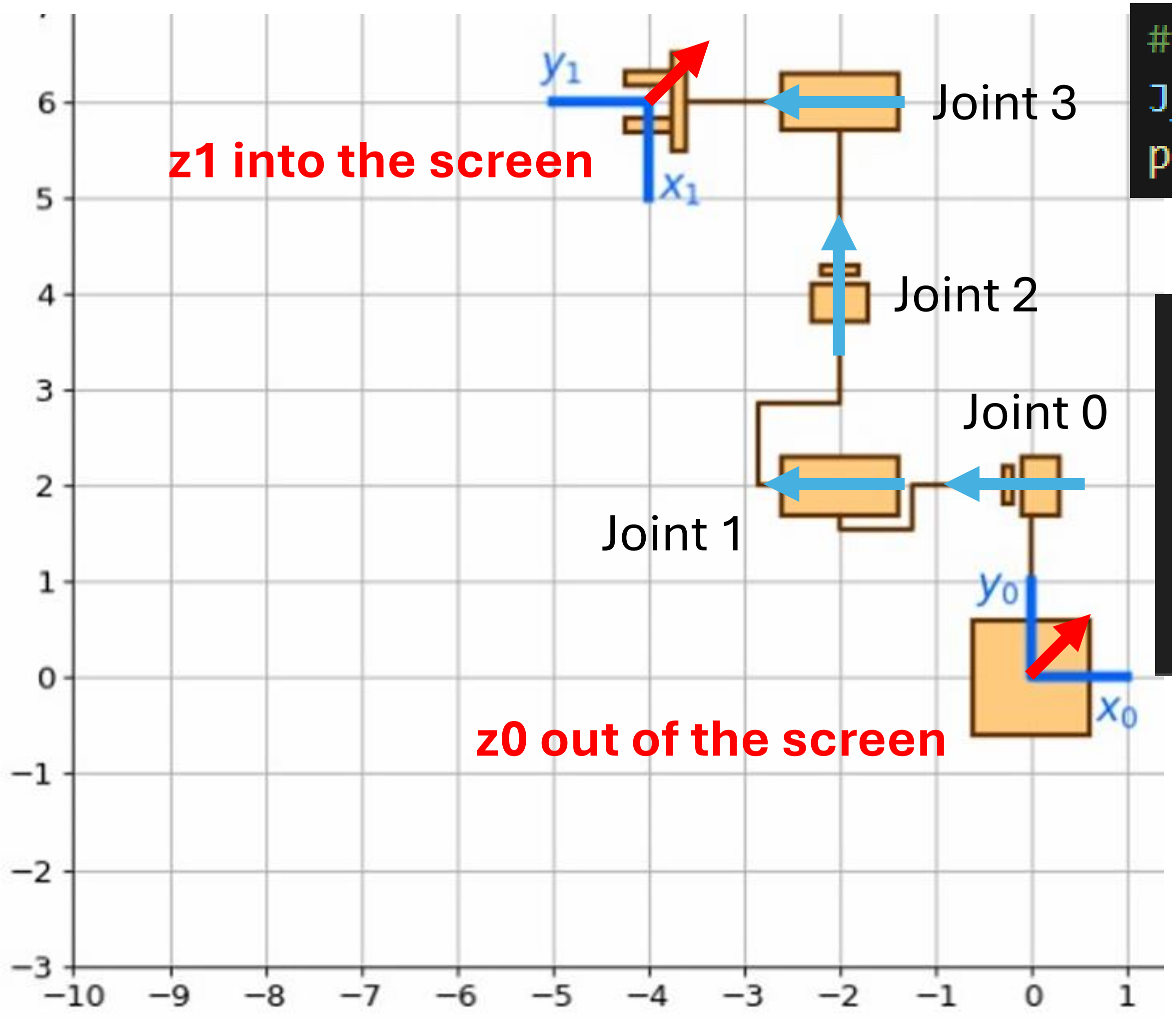
```
Ad_exp_s_2_theta_2 = adjoint(expm(mat_s_2 * theta_2))
J_s_3 = Ad_exp_s_0_theta_0 @ Ad_exp_s_1_theta_1 @ Ad_exp_s_2_theta_2 @ s_3
```

$$[Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}$$

```
[[-1.        ]
 [ 0.        ]
 [ 0.        ]
 [ 0.        ]
 [ 2.27727131]
 [ 6.16851717]]
```

4 DoF PRPR Robot Example

3. Fill in the columns of the Jacobian




```
# 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)
```

```
[[ 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]]
```

Why do we care about the Jacobian???

The Jacobian is a mapping from joint velocities to end-effector velocities

The resulting ellipsoid from the mapping tells us how easy it is for the end-effector to move in certain directions

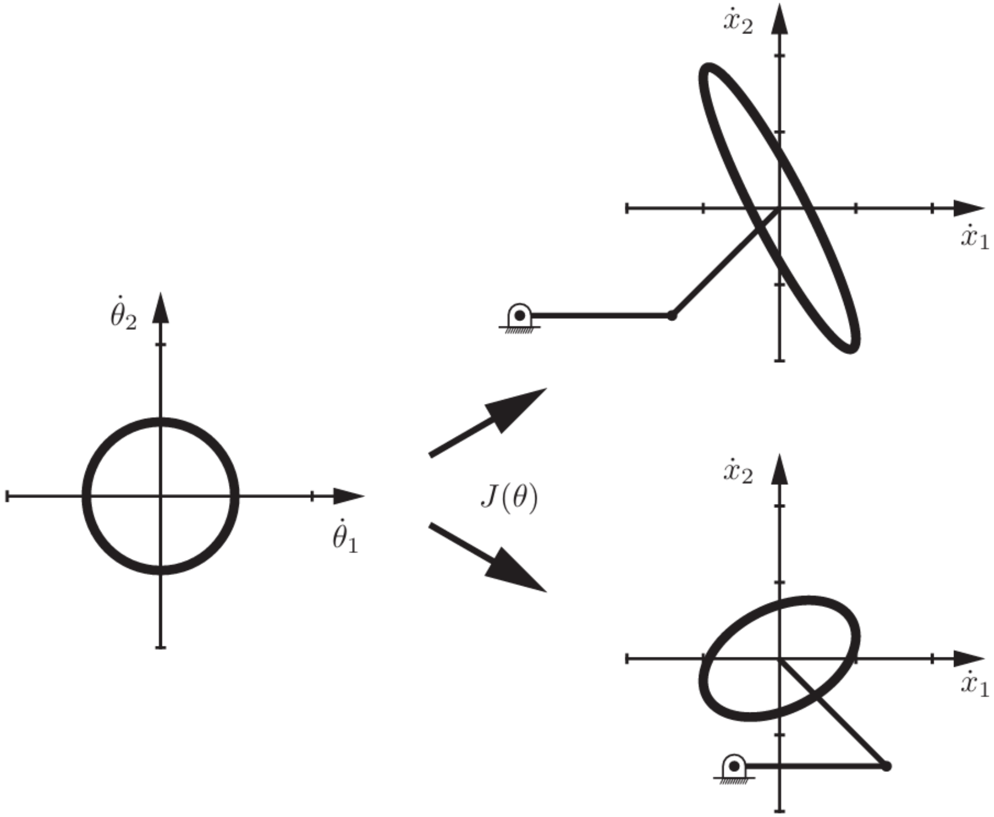
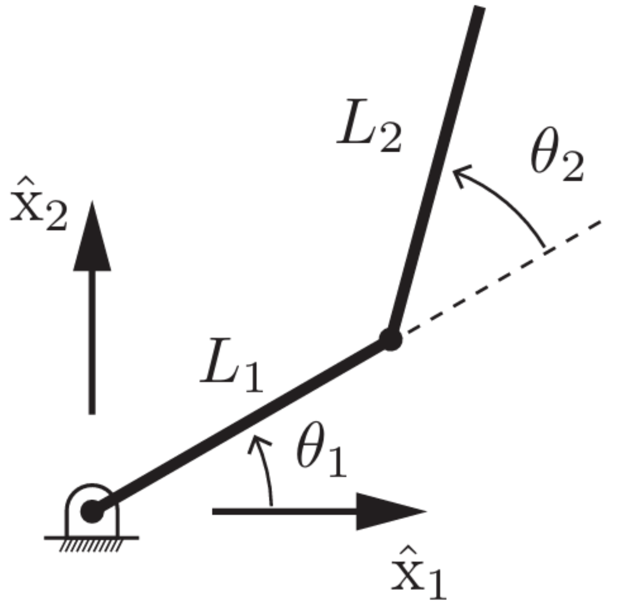


Figure 5.3: Manipulability ellipsoids for two different postures of the 2R planar open chain.

For a planar 2R robot:

$$
\begin{aligned}
x_1 &= L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\
x_2 &= L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2).
\end{aligned}
$$



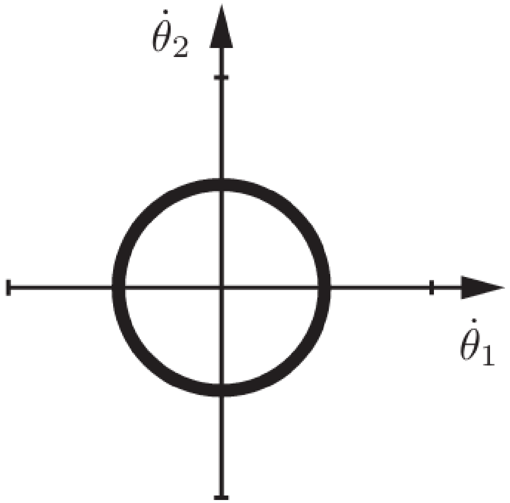
$$
\begin{aligned}
\dot{x}_1 &= -L_1\dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2) \\
\dot{x}_2 &= L_1\dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2),
\end{aligned}
$$

$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \boxed{\begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}
$$

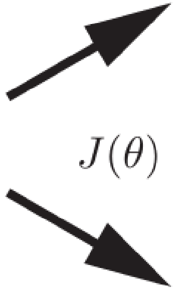
Jacobian J

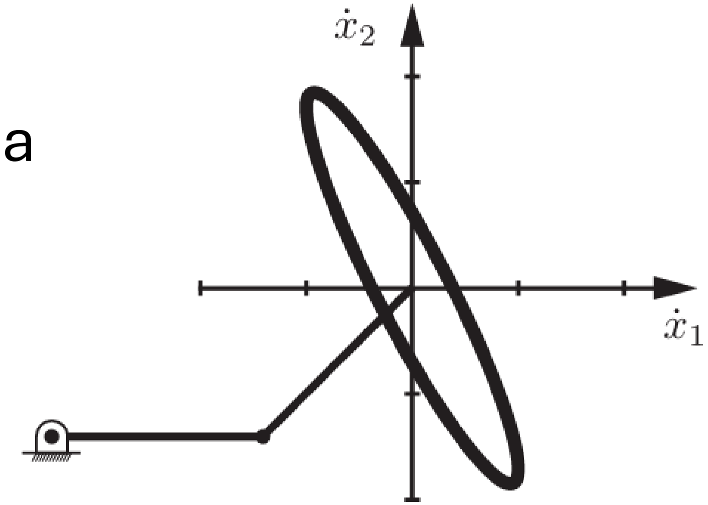
Visualizing the manipulability ellipse

For every robot configuration, we have a new Jacobian

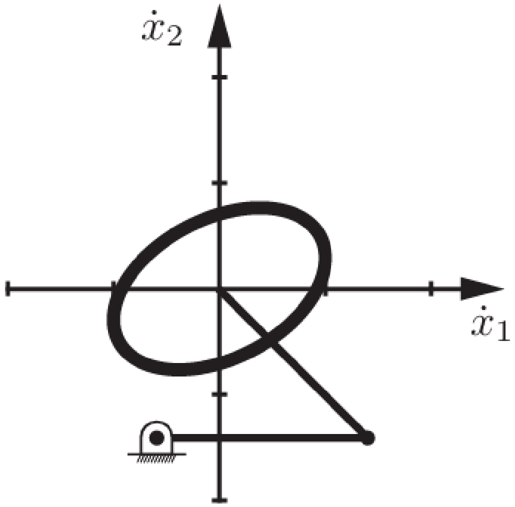


Letting the joint velocity limits be a unit circle





This configuration is closer to a singularity



This configuration is further from a singularity

Computing the manipulability ellipse for 2R

If J is invertible, let $A = JJ^T \in R^{2\text{x}2}$

Let λ_1 be the larger eigenvalue of A and λ_2 be the smaller one

Eigenvector v_1 is the direction of the major principal axis of the ellipse, and v_2 is the direction of the minor axis

The length of the major semi-axis is $\sqrt{\lambda_1}$ and the length of the minor semi-axis is $\sqrt{\lambda_2}$

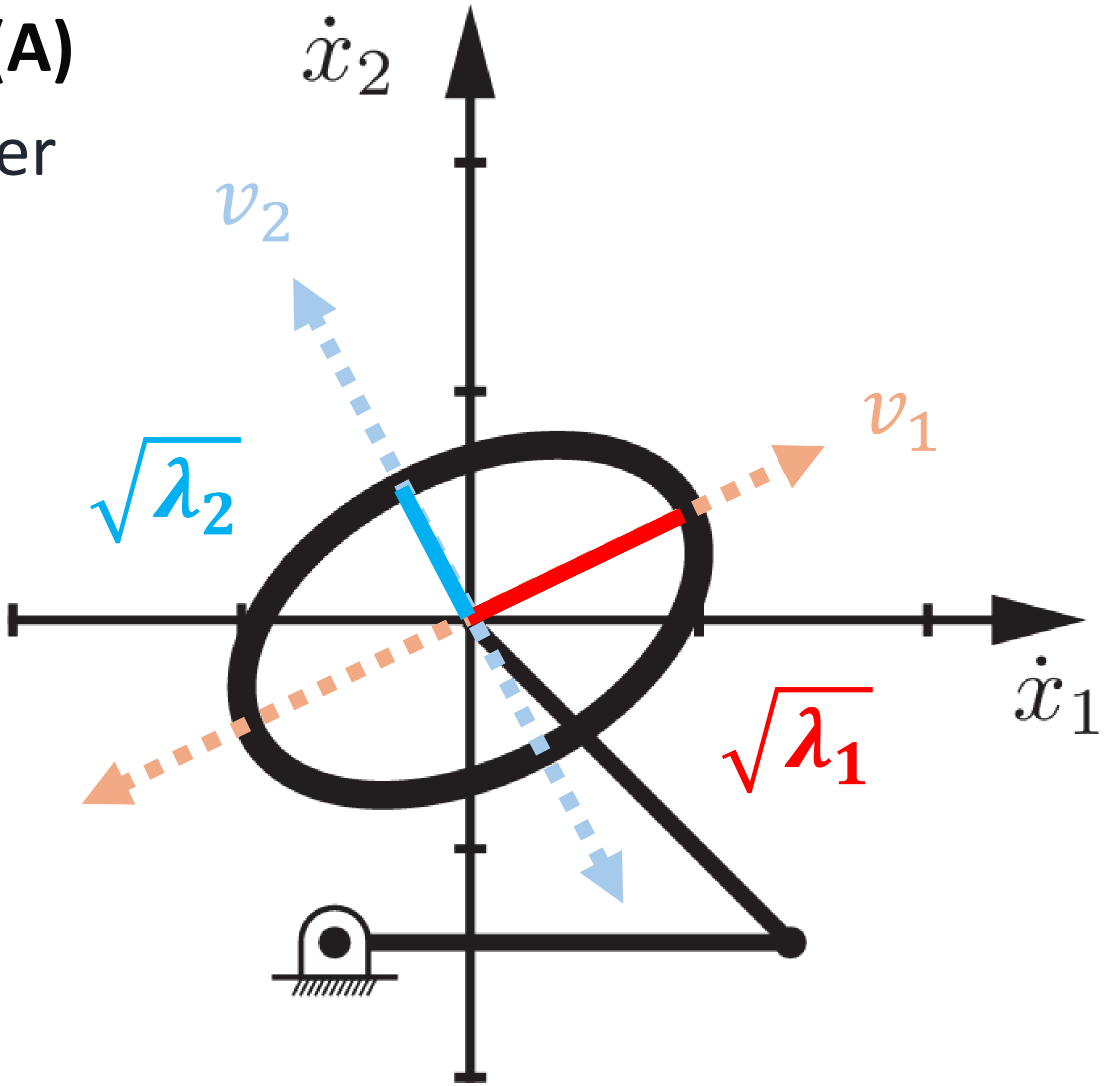
Computing the manipulability ellipse for 2R

eigenvalues, eigenvectors = np.linalg.eigh(A)

Note: eigenvalues will be in ascending order

If J is invertible, let

$$A = JJ^T \in R^{2x2}$$



https://numpy.org/doc/stable/reference/generated/numpy.linalg.eigh.html

The force ellipsoid

J^{-T} is a mapping from joint torques to end-effector forces

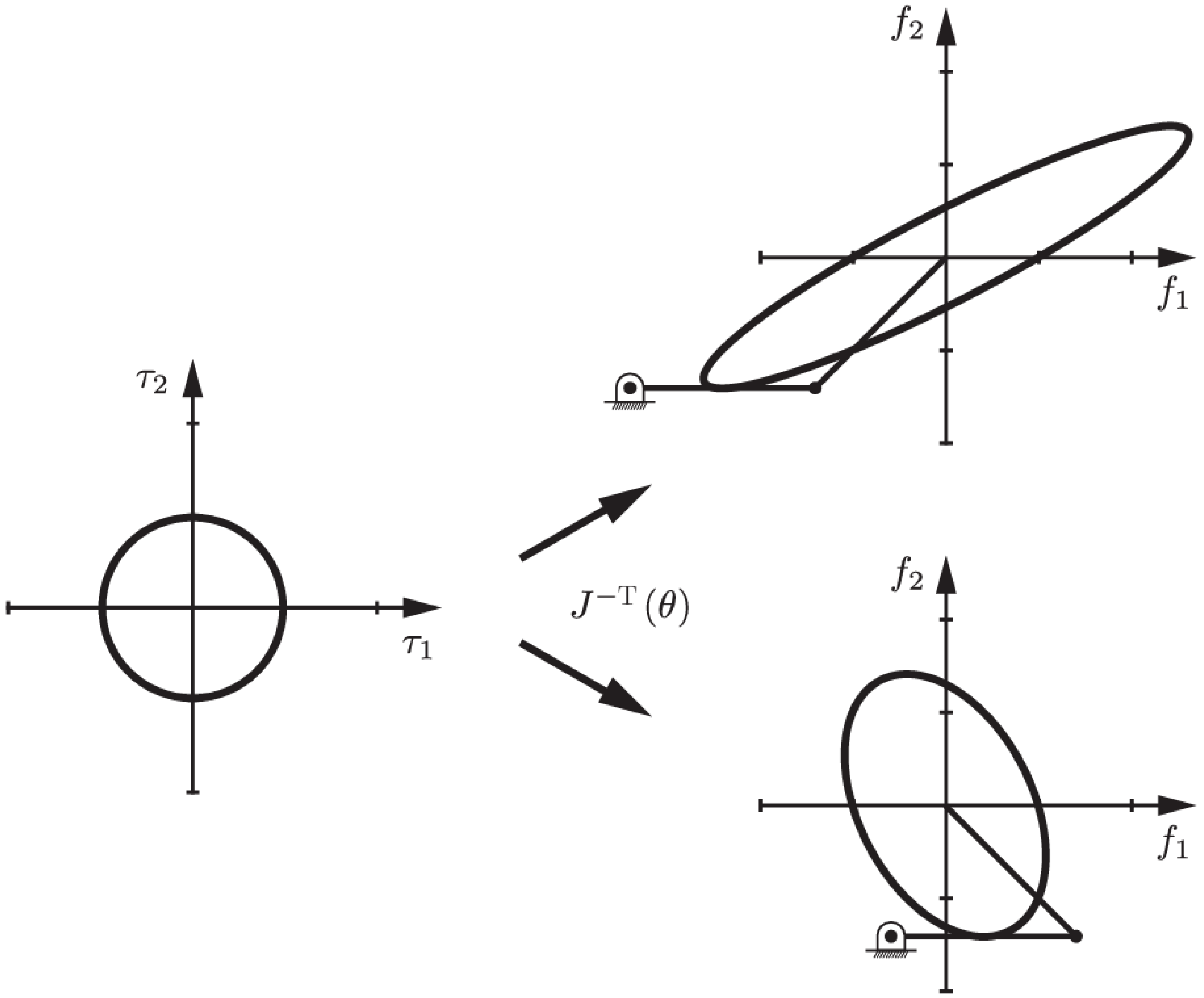


Figure 5.5: Force ellipsoids for two different postures of the 2R planar open chain.

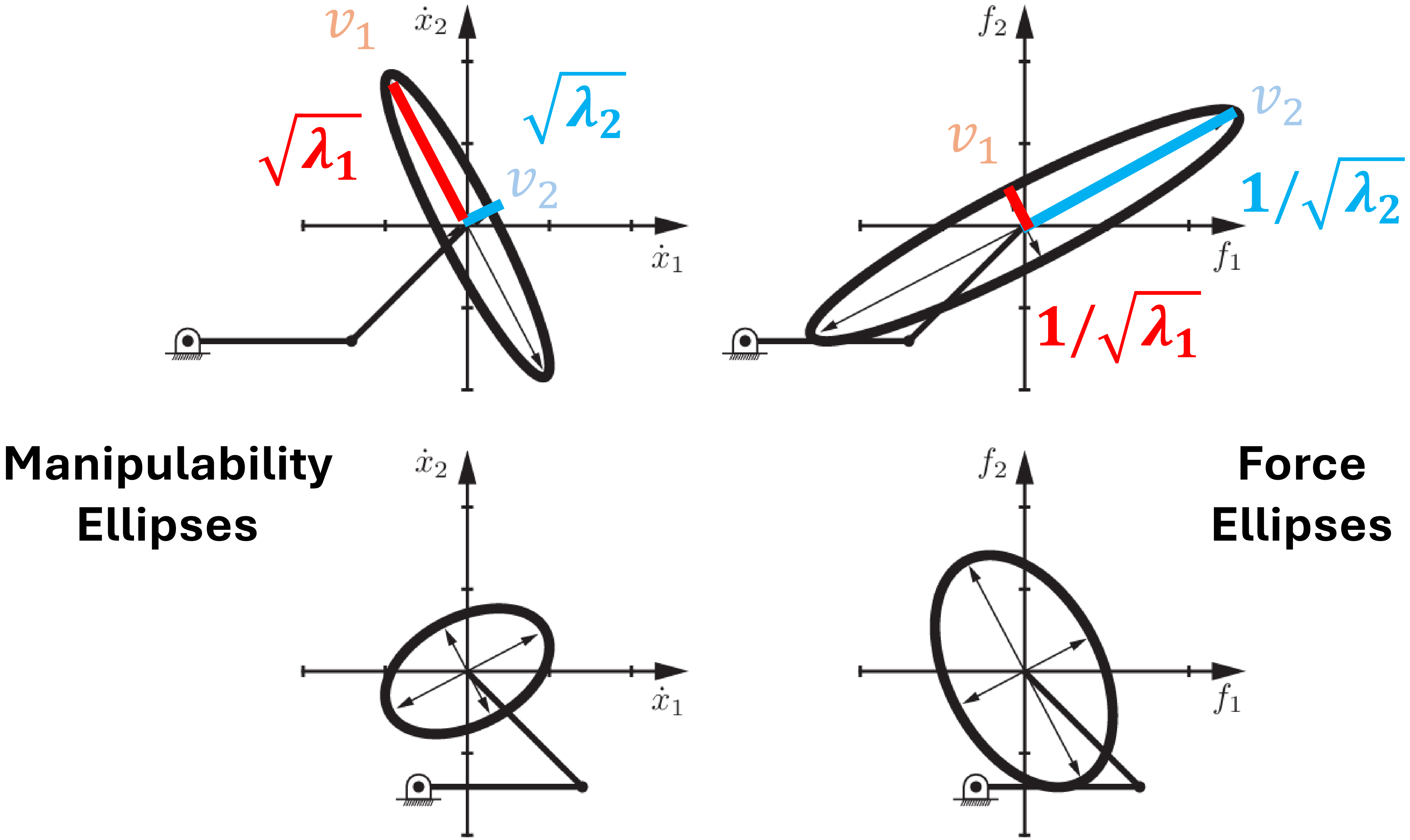


Figure 5.6: Left-hand column: Manipulability ellipsoids at two different arm configurations. Right-hand column: The force ellipsoids for the same two arm configurations.