Computer Animation

Lab 5 - A2 - Inverse Kinematics

COMP 477

UNIVERSITÉ Concordia UNIVERSITY

Preliminaries and Readings

Slides based on [Buss et al. 2004] ← I **highly** suggest you read this

Assignment 2

Implement 2D Inverse Kinematics with Obstacle Avoidance

- Forward Kinematics:
  - Going from local positions and rotations to global positions and rotations
- Inverse Kinematics:
  - Going from global positions to local positions and rotations

References
[Buss et al. 2004] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, 17(1-19):16, 2004.

Assignment 2

Goal:

Implement Inverse Kinematics with Obstacle Avoidance

Instructions:

https://moodle.concordia.ca/moodle/pluginfile.php/5166842/mod_resource/content/0/A2.pdf

Github:

https://github.com/tiperiu/COMP477_A2

Jacobian

How much does each joint position change with each angle change?

$$J(\boldsymbol{\theta}) = \left(\frac{\partial \mathbf{s}_i}{\partial \theta_j}\right)_{i,j}$$

Then, **FK** can be described as: $\dot{\vec{\mathbf{s}}} = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$

For **IK**, we wish to find a set of joint angles Δθ such that: $\boldsymbol{\theta} := \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

We then get: $\Delta\vec{\mathbf{s}} \approx J\,\Delta\boldsymbol{\theta}$

Jacobian - Calculating

When we change the angle of a rotational joint by a small δ, how much does the position of an end effector change?

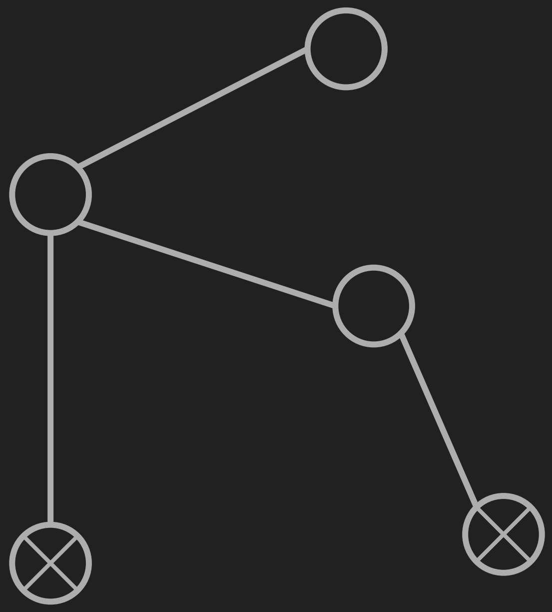
Rotational

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

Axis of Rotation

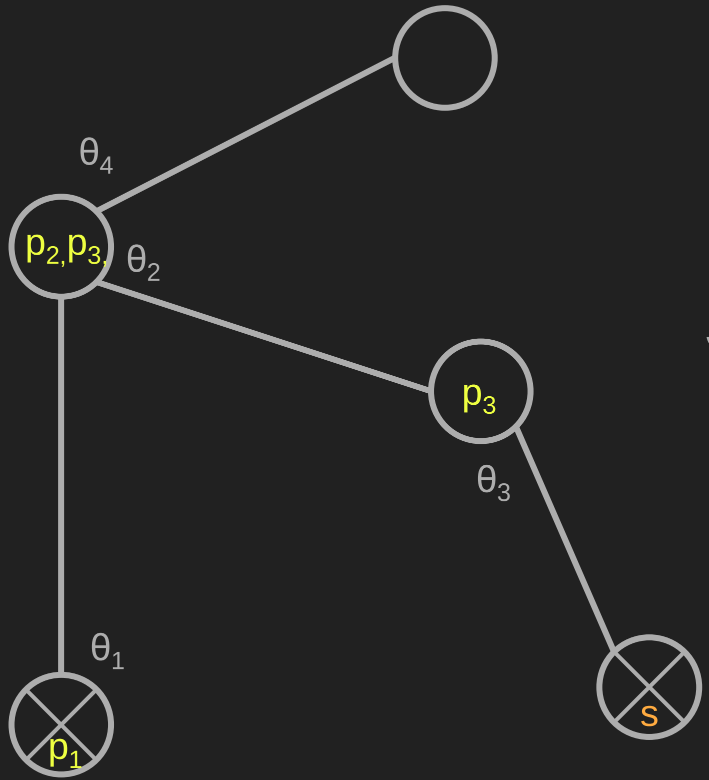
Position of end effector

Position of the joint



Jacobian - Calculating

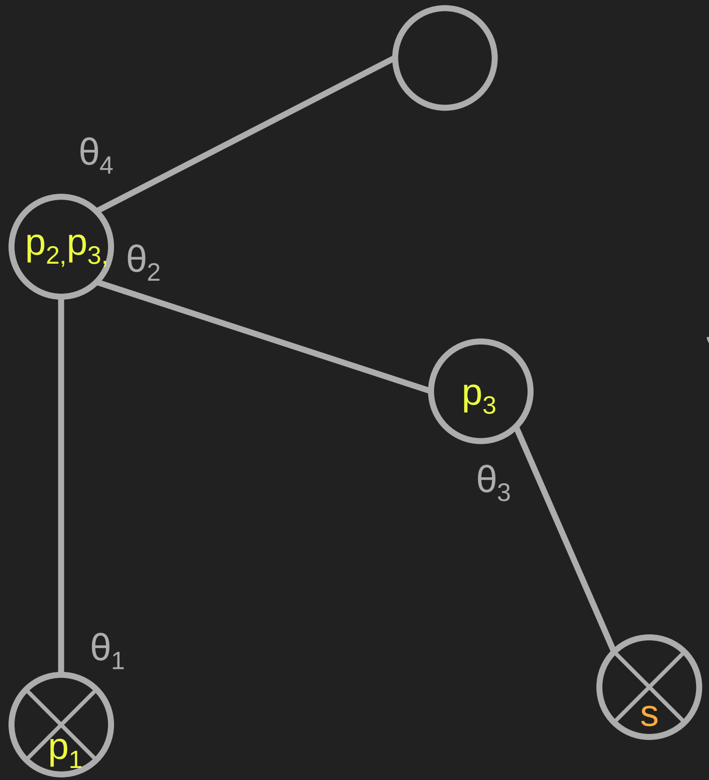
$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

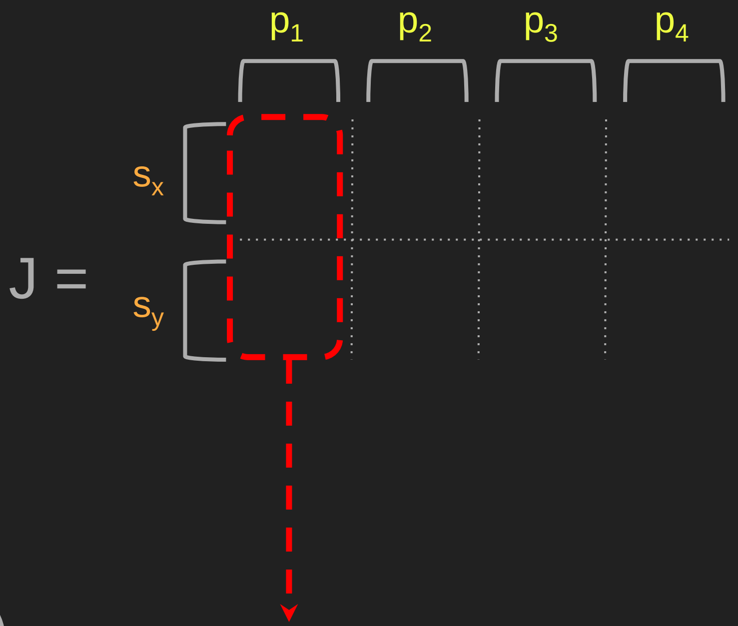


$$J = \begin{array}{c|cccc} & p_1 & p_2 & p_3 & p_4 \\ \hline s_x & & & & \\ s_y & & & & \end{array}$$

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

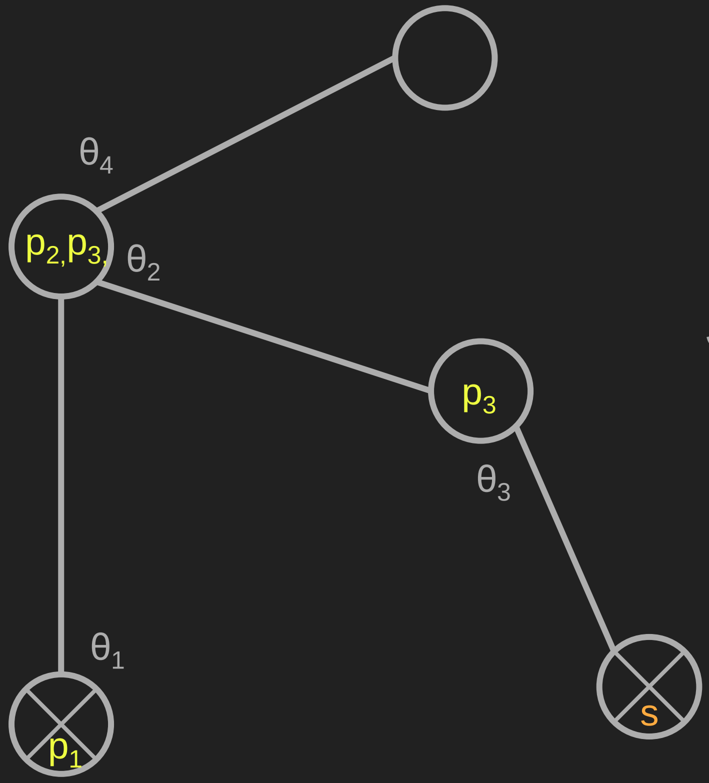


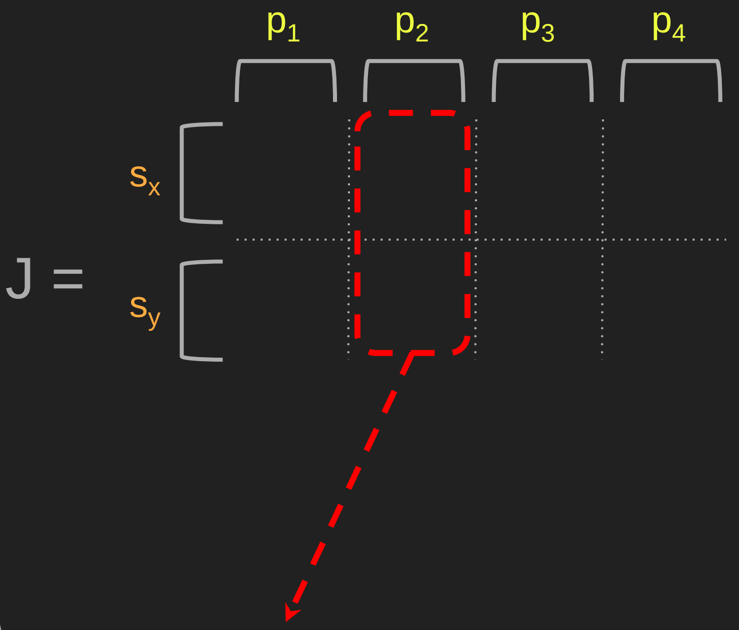


(0,0,1) x (s - p_1) ← Split into *x* and *y* components

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

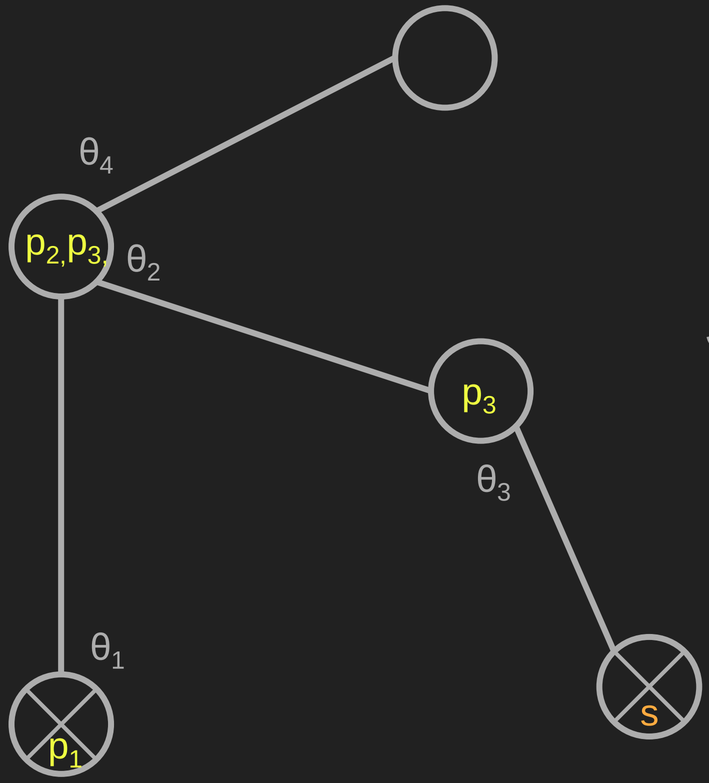




(0,0,1) x (s - p_2) ← Split into *x* and *y* components

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

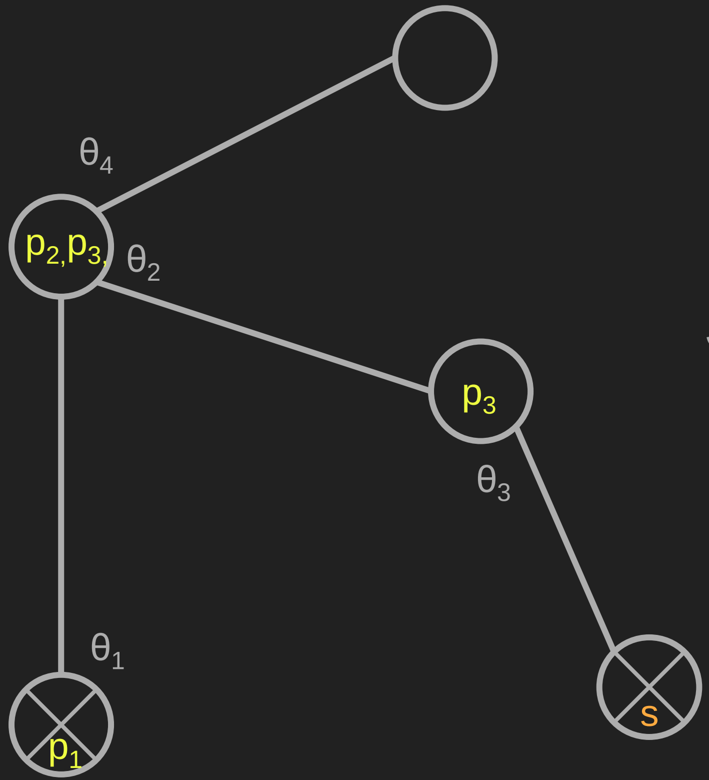


$$J = \begin{array}{c|cccc} & p_1 & p_2 & p_3 & p_4 \\ \hline s_x & & & & \\ s_y & & & & \end{array}$$

$(0,0,1) \times (s - p_3)$ ← Split into *x* and *y* components

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

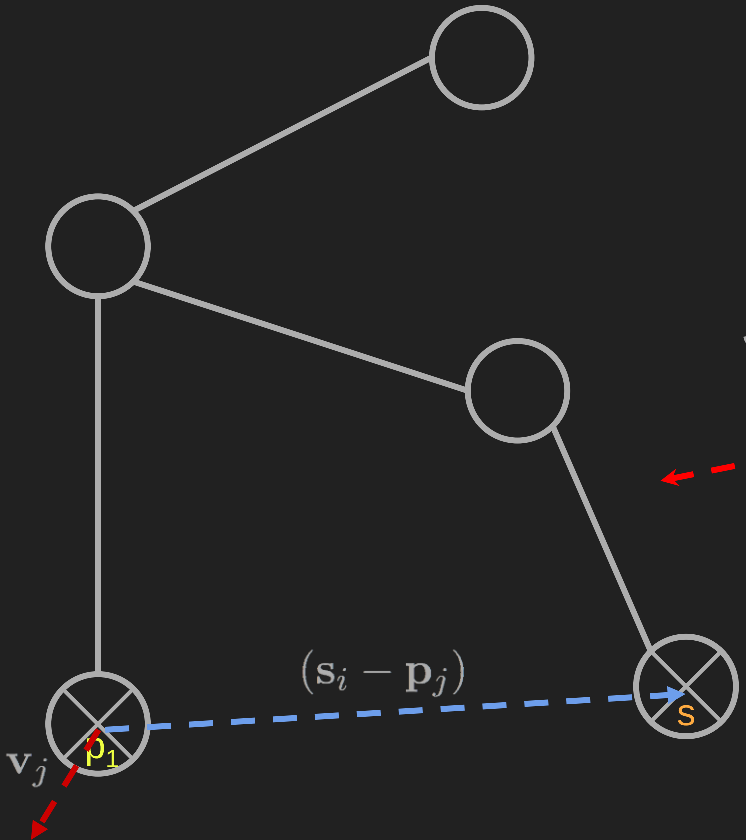


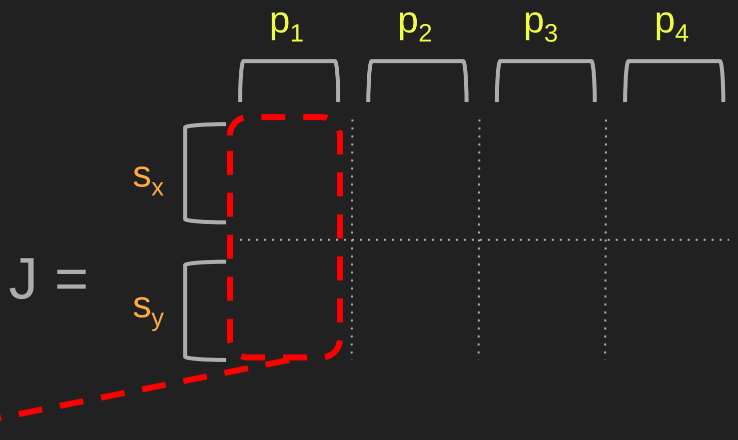
$$J = \begin{array}{c} \\ s_x \\ s_y \end{array} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \\ \hline & & & \boxed{} \\ & & & \boxed{} \end{array}$$

0 ← The rotation of joint 4 does not affect our end effector, thus the entries are 0. Generically, an entry is only non-zero if the given joint is a parent of the end effector.

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

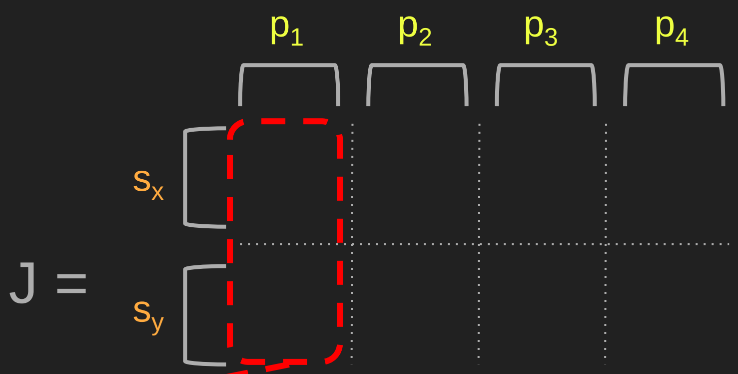


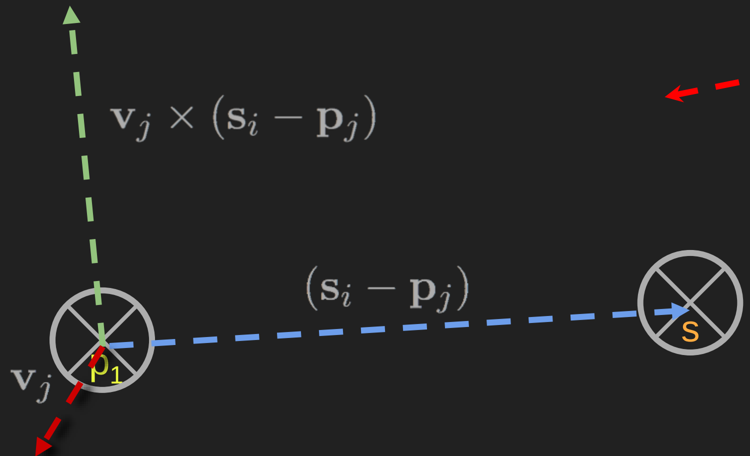


We can interpret this formula visually, it gives us a tangent in the direction of the joint's rotation!

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$

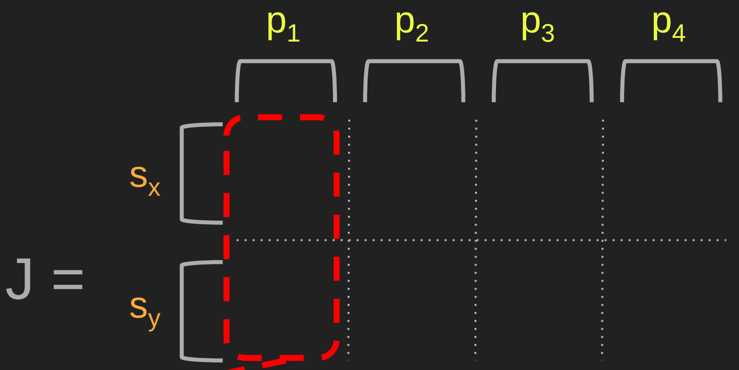


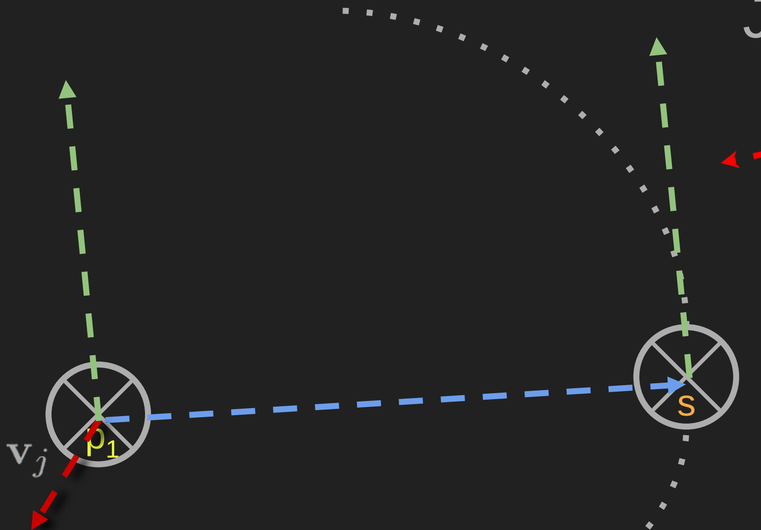


We can interpret this formula visually, it gives us a tangent in the direction of the joint's rotation!

Jacobian - Calculating

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j)$$





We can interpret this formula visually, it gives us a tangent in the direction of the joint's rotation!

Jacobian - Size

For IK, our jacobian will have the following size:

num_columns = 2 * num_end_effectors

num_rows = num_rotational_joints

For A2, you need only consider a single end effector at a time (the locked joint), and the number of rotational joints is equal to the number of joints in the hierarchy - 1

Jacobian - Solving

Now that we have our Jacobian, which describes how the end effector moves when we move a joint, it is sufficient to invert it to give us the opposite → by how much to move the joints to get the end effectors to a given target.

But wait...

Conditions for matrix invertibility:

- Square Matrix
- Non-zero determinant

Transpose Method

Find Δθ such that:

$$\Delta\theta = J^T \vec{\mathbf{e}}$$

Error term: vector from the end-effector to the target

Most “naive” method, can work ok for very simple applications, but it will be insufficient for our use case!

Pseudoinverse Method

Find Δθ such that:

$$\Delta\boldsymbol{\theta} = J^{\dagger}\vec{\mathbf{e}}$$

Compute the Moore-Penrose inverse of J:

$$\Delta\boldsymbol{\theta} = J^{T}(JJ^{T})^{-1}\vec{\mathbf{e}}$$

Better than the transpose method, but suffers from instabilities near singularities. System will oscillate near solution, resulting in poor performance. Still insufficient for our use case!

Damped Least Squares

Similar to previous mentioned pseudoinverse method, with one catch:

$$\Delta\boldsymbol{\theta} = J^T(JJ^T + \lambda^2 I)^{-1}\vec{\mathbf{e}}.$$

What is λ ?

Damping Factor

- Factor that ensures numerical stability
- Depends on details of your hierarchical structure
- Depending on implementation, values usually in the ballpark of 5 - 50

Obstacle Avoidance

Next Lab…

Meanwhile, here’s a resource to get you started:

https://www.researchgate.net/profile/Anthony_Maciejewski/publication/242483646_Obstacle_Avoidance_for_Kinematically_Redundant_Manipulators_in_Dynamically_Varying_Environments/links/542ae1ed0cf27e39fa9177a3.pdf