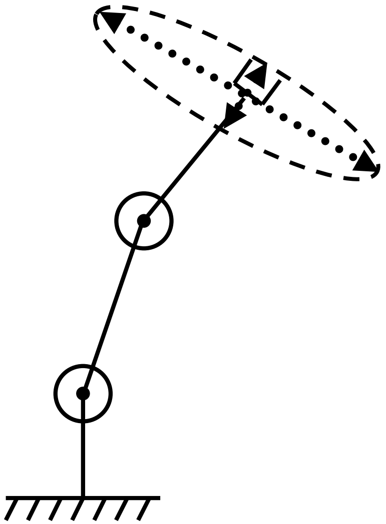
COMS W4733: Computational Aspects of Robotics

Lecture 6: Inverse Differential Kinematics



Instructor: Tony Dear

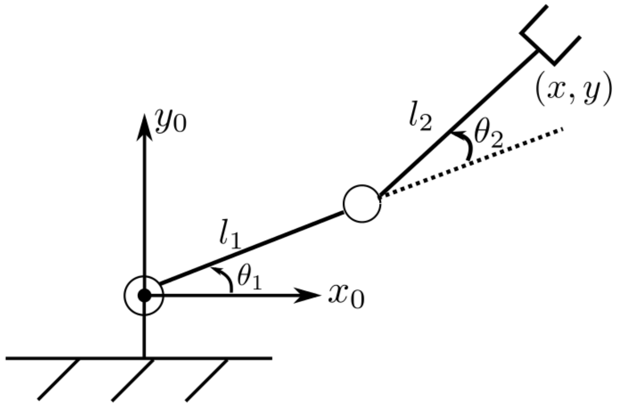
Review: Differential Kinematics

- Joint variables $\boldsymbol{q} = (q_1, \dots, q_n)^T \in$ **configuration space**
- End effector pose $\boldsymbol{x}_e \in$ **operational space**

- Differential kinematics: *Linear mapping* from joint velocities $\dot{\boldsymbol{q}}$ to end effector velocities $\dot{\boldsymbol{x}}_e$

$$\dot{\boldsymbol{x}}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

- **Jacobian**: Configuration-dependent matrix $\boldsymbol{J}(\boldsymbol{q})$
- 6 rows, one for each $\dot{\boldsymbol{x}}_e$ component (in 3D)
  - $\dot{\boldsymbol{x}}_e = (\dot{\boldsymbol{p}}_e^T, \boldsymbol{\omega}_e^T)^T = \left(\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z\right)^T$
- n columns, one for each joint



Review: The Jacobian

$$\boldsymbol{v}_e = \begin{pmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_P(\boldsymbol{q}) \\ \boldsymbol{J}_O(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

- Linear velocity Jacobian

$$\boldsymbol{J}_P(\boldsymbol{q}) = \begin{pmatrix} \frac{\partial p_{e,x}}{\partial q_1} & \cdots & \frac{\partial p_{e,x}}{\partial q_n} \\ \frac{\partial p_{e,y}}{\partial q_1} & \cdots & \frac{\partial p_{e,y}}{\partial q_n} \\ \frac{\partial p_{e,z}}{\partial q_1} & \cdots & \frac{\partial p_{e,z}}{\partial q_n} \end{pmatrix}$$

Or column by column:

$$[\boldsymbol{J}_{Pi}] = \begin{cases} [\boldsymbol{z}_{i-1}^0] & \text{prismatic} \\ [\boldsymbol{z}_{i-1}^0 \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1})] & \text{revolute} \end{cases}$$

- Angular velocity Jacobian

$$[\boldsymbol{J}_{Oi}] = \begin{cases} [\boldsymbol{0}] & \text{prismatic} \\ [\boldsymbol{z}_{i-1}^0] & \text{revolute} \end{cases}$$

The Inverse Problem

- Inverse position kinematics: Nonlinear, difficult to find closed-form solutions; even numerical solutions often involve some optimization or gradient descent

- Differential mapping between velocities is linear—can we simply invert it?
  - Solving for joint velocities will also help with joint configurations (next lecture)!

- If the Jacobian is square and nonsingular, then the problem is easy: $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\boldsymbol{v}_d$

- Number of joints must equal number of *desired workspace velocities* $\boldsymbol{v}_d$

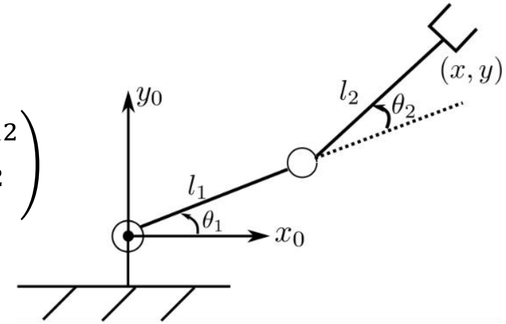
Square Jacobians

- RR arm is 3×2. Not square!
- We can only specify two desired velocities:

$$\dot{\boldsymbol{q}} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{pmatrix}^{-1} \begin{pmatrix} \dot{x}_d \\ \dot{y}_d \end{pmatrix}$$

$$\dot{\boldsymbol{q}} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \dot{x}_d \\ \omega_{z,d} \end{pmatrix}$$

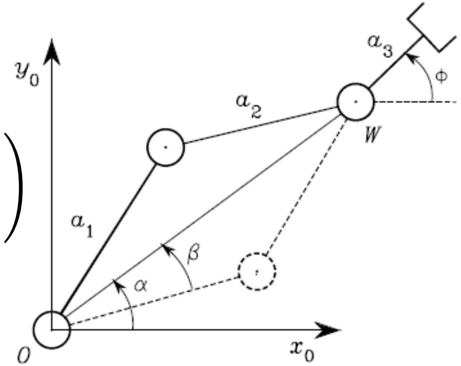
$$\boldsymbol{J}_{RR} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 1 & 1 \end{pmatrix}$$



- RRR arm is 3×3. We can fully specify $(\dot{x}, \dot{y}, \omega_z)^T$ and invert J

$$\boldsymbol{J}_{RRR} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} & -l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_2 c_{123} & l_2 c_{12} + l_2 c_{123} & l_2 c_{123} \\ 1 & 1 & 1 \end{pmatrix}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{RRR}{}^{-1} \begin{pmatrix} \dot{x}_d \\ \dot{y}_d \\ \omega_{z,d} \end{pmatrix}$$



Underconstrained Manipulators

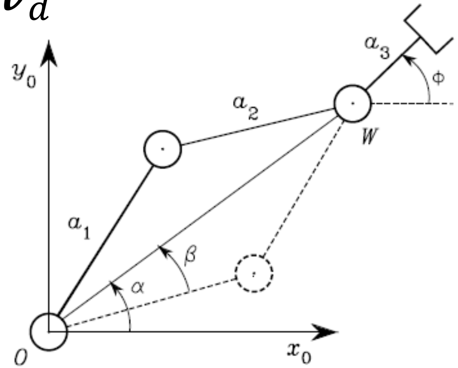
- *Underconstrained / redundant* manipulator: Fewer specifications than DOFs
- Occurs when more than 3 DOFs in 2D or more than 6 DOFs in 3D
- Also when fewer pose velocities are required

- Jacobian: Fewer rows (r) than columns (n)—more than one solution to IDK

$$\boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{v}_d \iff \dot{\boldsymbol{q}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}\boldsymbol{v}_d \iff \dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{v}_d$$

- $\boldsymbol{J}\boldsymbol{J}^T$ is $r \times r$ and square—invertible!
  - Non-singular iff $\boldsymbol{J}$ is non-singular
  - $\boldsymbol{W}$ is an invertible matrix as well

$$\begin{pmatrix}\dot{x}_d \\ \dot{y}_d\end{pmatrix} = \boldsymbol{J}_{RRR}\dot{\boldsymbol{q}}$$



Underconstrained Manipulators

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{v}_d$$

- Suppose $\boldsymbol{W}$ is symmetric and positive-definite
- Then the above minimizes a cost function in joint velocities: $g(\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{W}\dot{\boldsymbol{q}}$

- Ex: Suppose $(q_1, q_2, q_3)^T = (60°, -30°, 15°)^T$ and $\boldsymbol{v}_d = (\dot{x}_d, \dot{y}_d)^T = (1,1)^T$ m/s

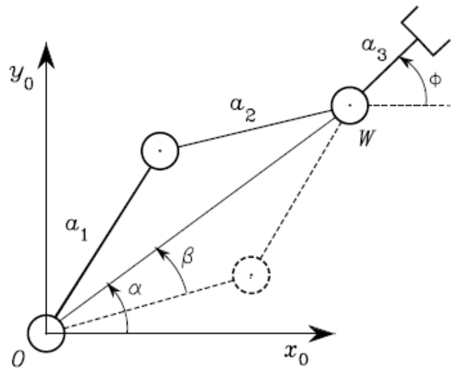
$$\boldsymbol{J}_{RRR} = \begin{pmatrix} -2.07 & -1.21 & -0.71 \\ 2.07 & 1.57 & 0.71 \end{pmatrix}$$

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad \dot{\boldsymbol{q}} = \begin{pmatrix} -3.59 \\ 5.56 \\ -0.41 \end{pmatrix}$$

Solution tries to have smaller $|\dot{\theta}_3|$

$$\boldsymbol{W} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \dot{\boldsymbol{q}} = \begin{pmatrix} -2.76 \\ 5.56 \\ -2.84 \end{pmatrix}$$

Solution tries to have smaller $|\dot{\theta}_1|$



Right Pseudo-inverse

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T)^{-1}\boldsymbol{v}_d$$

- For the special case $\boldsymbol{W} = \boldsymbol{I}$, all joint velocities are equally weighted

$$\boxed{\dot{\boldsymbol{q}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}\boldsymbol{v}_d = \boldsymbol{J}_r^+\boldsymbol{v}_d}$$

- *Right pseudo-inverse* of $\boldsymbol{J}$ defined s.t. $\boldsymbol{J}\boldsymbol{J}_r^+ = \boldsymbol{I}$

- **Moore-Penrose conditions**:
  - $J^+JJ^+ = J^+$
  - $JJ^+J = J$
  - $(JJ^+)^T = JJ^+$
  - $(J^+J)^T = J^+J$

Other useful properties:

- $(J^+)^+ = J$
- $(J^+)^T = (J^T)^+$

Jacobian Null Space

- For an underconstrained manipulator, the Jacobian maps from a higher-dimensional input (n joint DOF columns) to a lower-dimensional output (r desired velocity rows)

- A non-singular matrix has rank r (if $r \leq n$) and *null space* dimension $n - r$
  - Subspace of joint velocities $\dot{\boldsymbol{q}}$ that map to **zero** end effector velocities

- $(\boldsymbol{I} - \boldsymbol{J}_r^+ \boldsymbol{J})\dot{\boldsymbol{q}}_0$ lives in the null space of $\boldsymbol{J}$ because

$$\boldsymbol{J}(\boldsymbol{I} - \boldsymbol{J}_r^+ \boldsymbol{J})\dot{\boldsymbol{q}}_0 = (\boldsymbol{J} - \boldsymbol{J}\boldsymbol{J}_r^+ \boldsymbol{J})\dot{\boldsymbol{q}}_0 = (\boldsymbol{J} - \boldsymbol{J})\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$$

- Therefore we can add $(\boldsymbol{I} - \boldsymbol{J}_r^+ \boldsymbol{J})\dot{\boldsymbol{q}}_0$ to any joint velocity solution without changing the desired end effector velocity

Homogeneous Solution

$$\boxed{\dot{\boldsymbol{q}} = \boldsymbol{J}_r^+ \boldsymbol{v}_d + (\boldsymbol{I} - \boldsymbol{J}_r^+ \boldsymbol{J})\dot{\boldsymbol{q}}_0}$$

- The second term is the **homogeneous solution** and minimizes a new cost function

$$g'(\dot{\boldsymbol{q}}) = \frac{1}{2}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)^T(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_0)$$

- Simultaneously solve the IDK problem while returning a $\dot{\boldsymbol{q}}$ close to a specified $\dot{\boldsymbol{q}}_0$
- One idea: Choose $\dot{\boldsymbol{q}}_0$ to maximize secondary objective function:

$$\dot{\boldsymbol{q}}_0 = k_0 \left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T$$

- Ex: Distance from joint limits

$$w(\boldsymbol{q}) = \sum_{i=1}^{n} \left(\frac{q_i - \overline{q}_i}{q_{iM} - q_{im}}\right)^2$$

Joint i ranges from q_{im} to q_{iM};
middle value is $\overline{q}_i$

- Ex: Distance from obstacle

Overconstrained Manipulators

- *Overconstrained* manipulator: More specifications (rows) than DOFs (columns)
- Not enough DOFs to find exact joint velocity solutions

- Jacobian: Fewer columns than rows ($n < r$)
- Right pseudo-inverse undefined since $\boldsymbol{J}\boldsymbol{J}^T$ has rank $n < r$

- Instead, we define the *left pseudo-inverse*: $\boldsymbol{J}_l^+ = (\boldsymbol{J}^T\boldsymbol{J})^{-\mathbf{1}}\boldsymbol{J}^T$, s.t. $\boldsymbol{J}_l^+\boldsymbol{J} = \boldsymbol{I}$
- $\boldsymbol{J}^T\boldsymbol{J}$ is $n \times n$ and therefore full rank and invertible
- $\boldsymbol{J}_l^+$ also satisfies Moore-Penrose conditions

Left Pseudo-inverse

$$\boxed{\dot{\boldsymbol{q}} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{v}_d = \boldsymbol{J}_l^+\boldsymbol{v}_d}$$

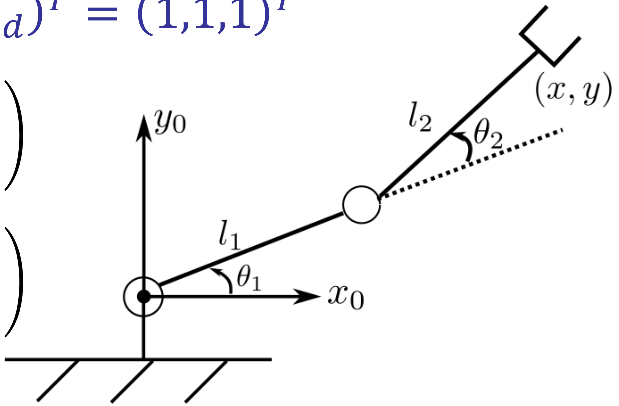
- Although we can't get exact solutions, the left pseudo-inverse tries to minimize the resulting error: $g(\dot{\boldsymbol{q}}, \boldsymbol{v}_d) = \frac{1}{2}(\boldsymbol{v}_d - \boldsymbol{J}\dot{\boldsymbol{q}})^T(\boldsymbol{v}_d - \boldsymbol{J}\dot{\boldsymbol{q}})$

- Ex: Suppose $(q_1, q_2)^T = (30°, 30°)^T$ and $\boldsymbol{v}_d = (\dot{x}_d, \dot{y}_d, \omega_d)^T = (1,1,1)^T$

$$\boldsymbol{J}_{RR} = \begin{pmatrix} -1.366 & -0.866 \\ 1.366 & 0.5 \\ 1 & 1 \end{pmatrix} \Rightarrow \dot{\boldsymbol{q}} = \begin{pmatrix} 0.146 \\ 0.107 \end{pmatrix} \Rightarrow \dot{\boldsymbol{x}} = \begin{pmatrix} -0.293 \\ 0.254 \\ 0.254 \end{pmatrix}$$

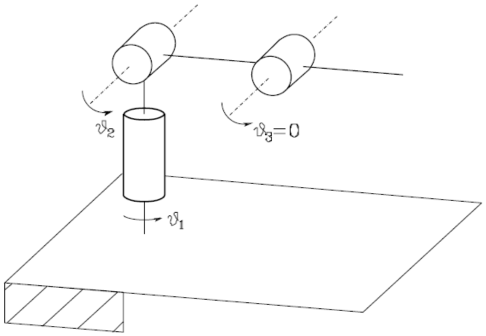
- What if $\boldsymbol{v}_d = (-2,2,2)^T$?

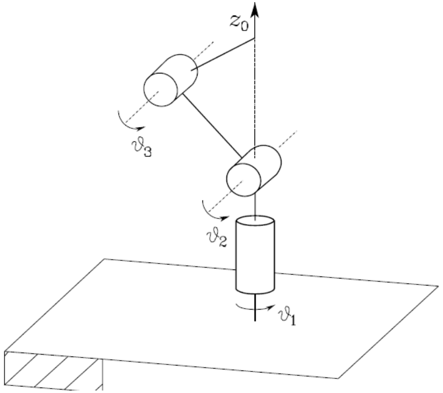
$$\dot{\boldsymbol{q}} = \begin{pmatrix} 1.093 \\ 0.8 \end{pmatrix} \Rightarrow \dot{\boldsymbol{x}} = \begin{pmatrix} -2.186 \\ 1.893 \\ 1.893 \end{pmatrix}$$



Singularities

- Recall: Singularities occur at configurations at workspace boundaries or when the robot loses DOFs
- Square Jacobians lose rank and cannot be inverted
- $\det(\boldsymbol{J})$ cannot be computed for non-square Jacobians
- If underconstrained ($r < n$), check $\det(\boldsymbol{J}\boldsymbol{J}^T)$
  - Both $\boldsymbol{J}$ and $\boldsymbol{J}\boldsymbol{J}^T$ should have full rank r; $\boldsymbol{J}\boldsymbol{J}^T$ is $r \times r$
- If overconstrained ($n < r$), check $\det(\boldsymbol{J}^T\boldsymbol{J})$
  - Both $\boldsymbol{J}$ and $\boldsymbol{J}^T\boldsymbol{J}$ should have full rank n; $\boldsymbol{J}^T\boldsymbol{J}$ is $n \times n$





Singular Value Decomposition

- We cannot compute either pseudo-inverse when the Jacobian is not full-rank
- Can we somehow ignore the "singular parts" and operate on the rest of the matrix?

- Recall the SVD of a matrix: $\boldsymbol{J} = \boldsymbol{U\Sigma V}^T$
  - $\boldsymbol{U}$ is $r \times r$**,** orthonormal (eigenvectors of $\boldsymbol{JJ}^T$)
  - $\boldsymbol{V}$ is $n \times n$**,** orthonormal (eigenvectors of $\boldsymbol{J}^T\boldsymbol{J}$)
  - $\boldsymbol{\Sigma}$ is $r \times n$**,** semi-diagonal matrix of *singular values*

$r < n$:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & \cdots & 0 \end{pmatrix}$$

$n < r$:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \sigma_n \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Inverting the SVD

- When we compute (either) pseudo-inverse, we're computing the following:

$$\boldsymbol{J}^{+} = \boldsymbol{V}\boldsymbol{\Sigma}^{+}\boldsymbol{U}^{T}$$

- $r < n$: $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & \cdots & 0 \end{pmatrix} \Rightarrow \boldsymbol{\Sigma}^{+} = \begin{pmatrix} 1/\sigma_1 & 0 & \cdots & 0 \\ 0 & 1/\sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 1/\sigma_r \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$

- $n < r$: $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \sigma_n \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \Rightarrow \boldsymbol{\Sigma}^{+} = \begin{pmatrix} 1/\sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1/\sigma_2 & \ddots & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sigma_n & \cdots & 0 \end{pmatrix}$

SVD at Singularities

- When the Jacobian is full-rank, the singular values matrix takes either of the two forms

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{D} & \boldsymbol{O} \end{pmatrix} \qquad\qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{D} \\ \boldsymbol{O} \end{pmatrix}$$

- If the Jacobian becomes singular, one or more singular values become zero and the matrix takes the form

$$\boldsymbol{\Sigma} = \begin{pmatrix} \widetilde{\boldsymbol{D}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix}$$

- The (pseudo-)inverse attempts to invert the zeros along the diagonal and fails!
- Even when singular values are near zero, inversion leads to large and unstable values

Damped Least-Squares

- Solution: Smooth or damp out the effects of the zero singular values
- For $r < n$, add a diagonal damping matrix to the matrix inverse term:

$$\boxed{\boldsymbol{J}^* = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + k^2\boldsymbol{I})^{-1}}$$

- Breaking down the **DLS pseudo-inverse** in SVD form:

$$\boldsymbol{J}^* = \boldsymbol{V}\boldsymbol{\Sigma}^*\boldsymbol{U}^T \qquad \boldsymbol{\Sigma}^* = \begin{pmatrix} \boldsymbol{D}^* \\ \boldsymbol{O} \end{pmatrix} \qquad \boldsymbol{D}^* = \mathrm{diag}(\sigma_i/(\sigma_i^2 + k^2))$$

- If $\sigma_i = 0$, inverse entry is also 0
- For $\sigma_i \ll k$, inverse entry gets smaller as k gets larger
- For $\sigma_i \gg k$, inverse entry is close to original $1/\sigma_i$

DLS Example

- RR arm at configuration $\boldsymbol{q} = (30°, 0°)^T$ is singular for $(\dot{x}, \dot{y})$

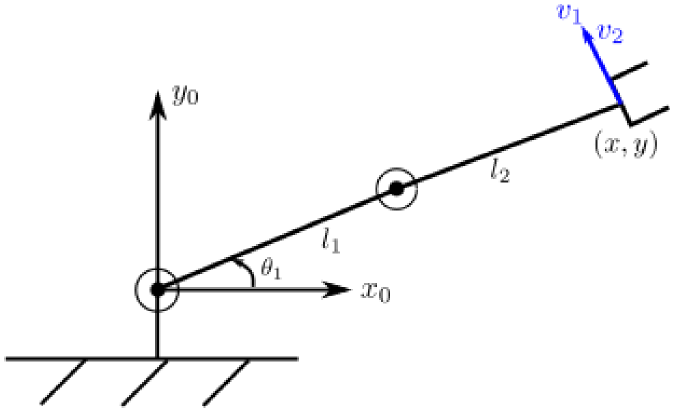
$$\boldsymbol{J}_{RR} = \boldsymbol{U\Sigma V}^T = \begin{pmatrix} -0.5 & \sqrt{3}/2 \\ \sqrt{3}/2 & 0.5 \end{pmatrix} \begin{pmatrix} 2.236 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0.894 & 0.447 \\ -0.447 & 0.894 \end{pmatrix}$$

$$\boldsymbol{J}_{RR} = \begin{pmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{pmatrix} = \begin{pmatrix} -1 & -0.5 \\ \sqrt{3} & \sqrt{3}/2 \end{pmatrix}$$

- Suppose we want to achieve $(\dot{x}, \dot{y})^T = \left(-1, \sqrt{3}\right)^T$

- Large $k = 10$: $\dot{\boldsymbol{q}} = \boldsymbol{J}_{RR}^* \begin{pmatrix} -1 \\ \sqrt{3} \end{pmatrix} = \begin{pmatrix} 0.267 \\ 0.133 \end{pmatrix}$

- Small $k = 10^{-4}$: $\dot{\boldsymbol{q}} = \boldsymbol{J}_{RR}^* \begin{pmatrix} -1 \\ \sqrt{3} \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix}$



Summary

- IDK: Solve for joint velocities to achieve desired workspace velocities
- When Jacobian is non-singular and square, a simple inverse is sufficient

- If Jacobian is underconstrained ($r < n$), use the right pseudo-inverse
  - Minimizes joint velocities, can also weight individual joints
  - Homogeneous solution characterized by null space, since there are infinite solutions

- Otherwise there may be no exact solutions!
- If Jacobian is overconstrained ($n < r$), use the left pseudo-inverse
- If Jacobian is singular, use the damped least-squares inverse