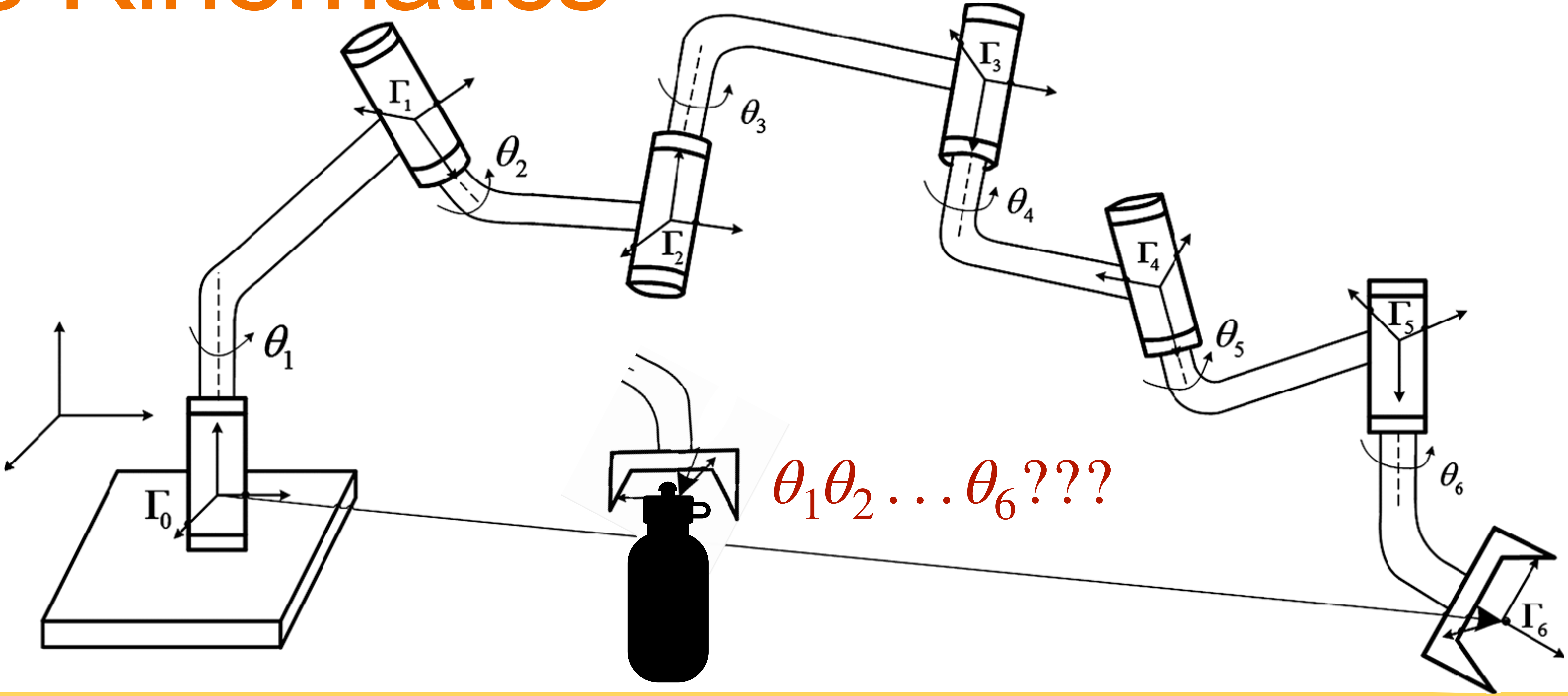
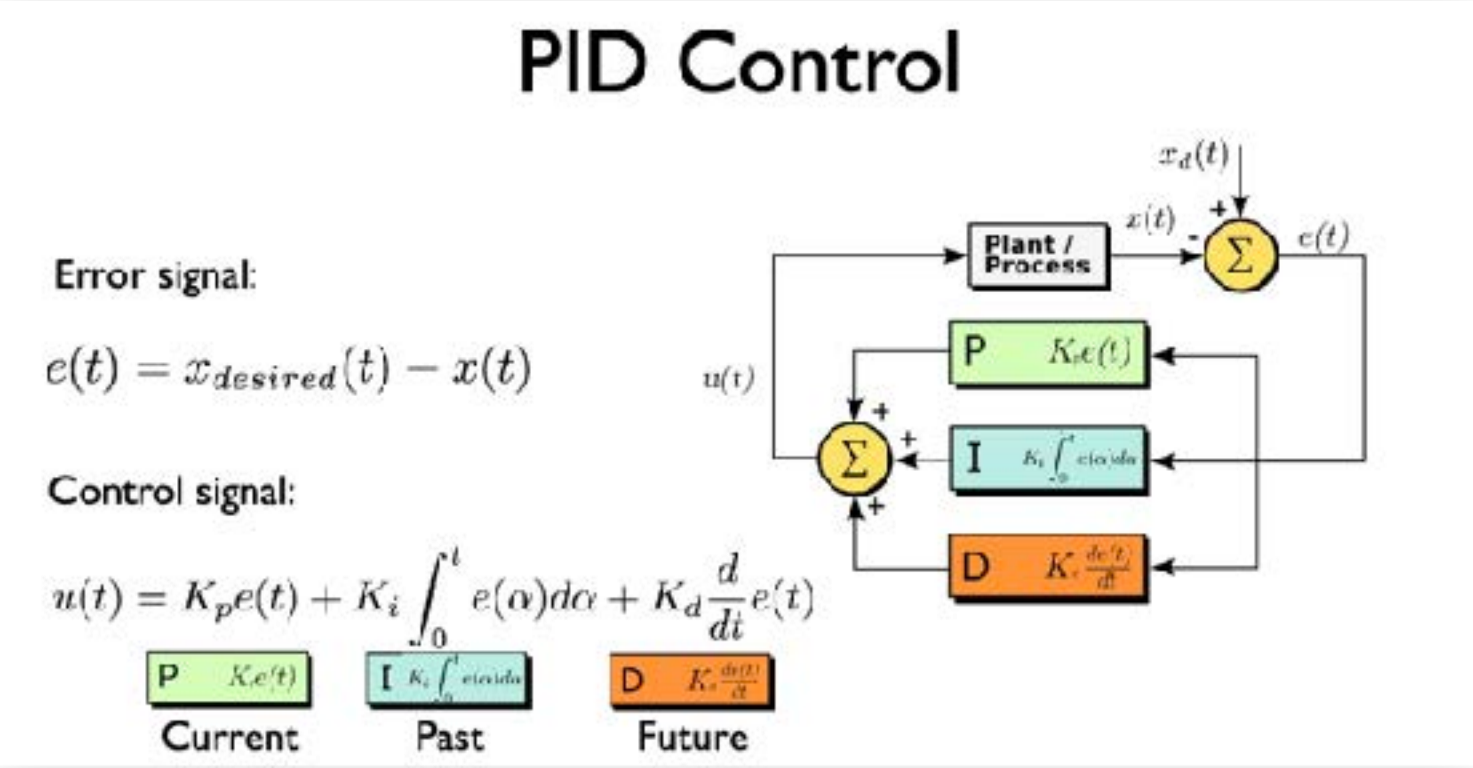
Lecture 08
Manipulation - III
Inverse Kinematics

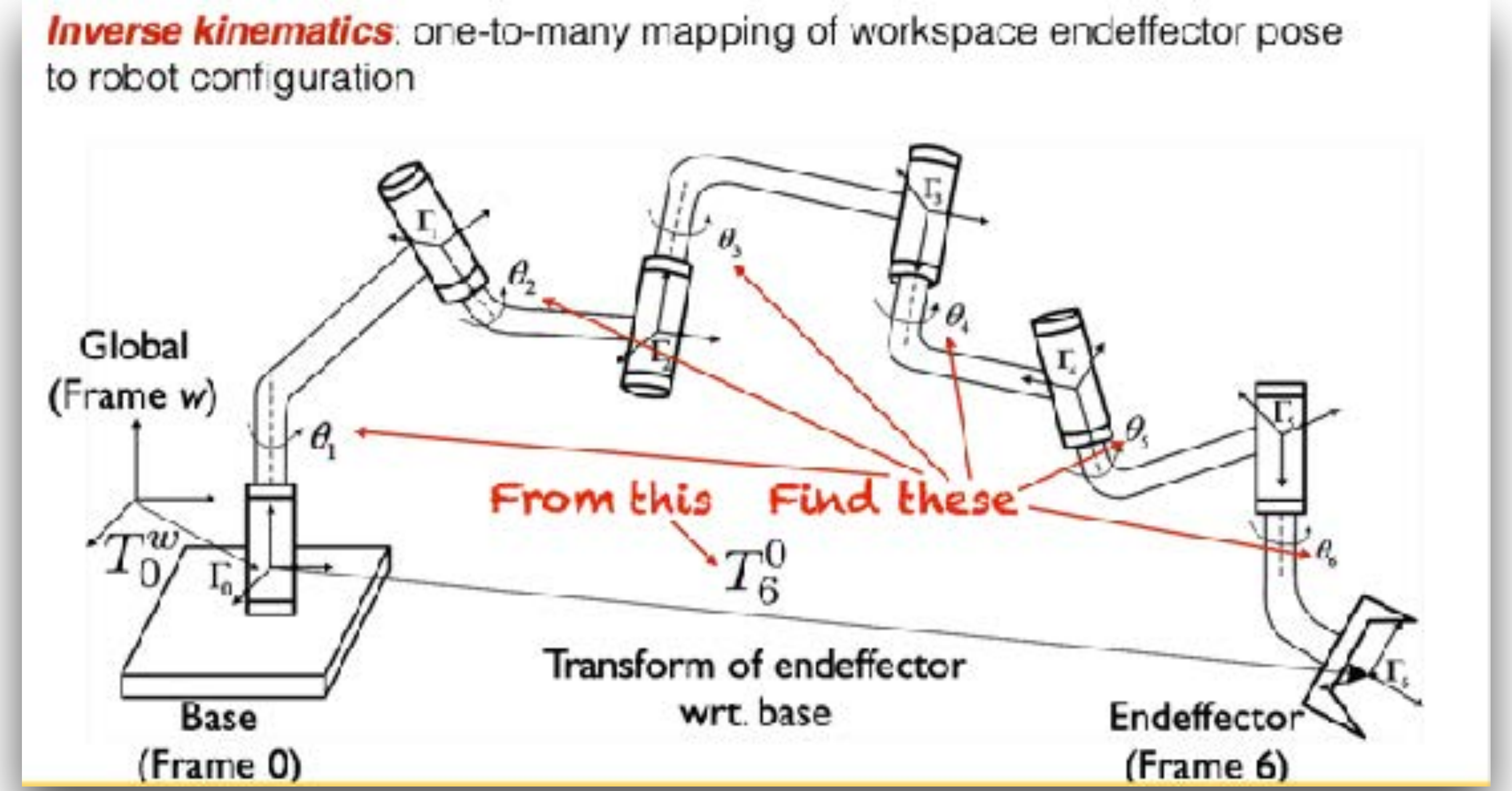


Course Logistics

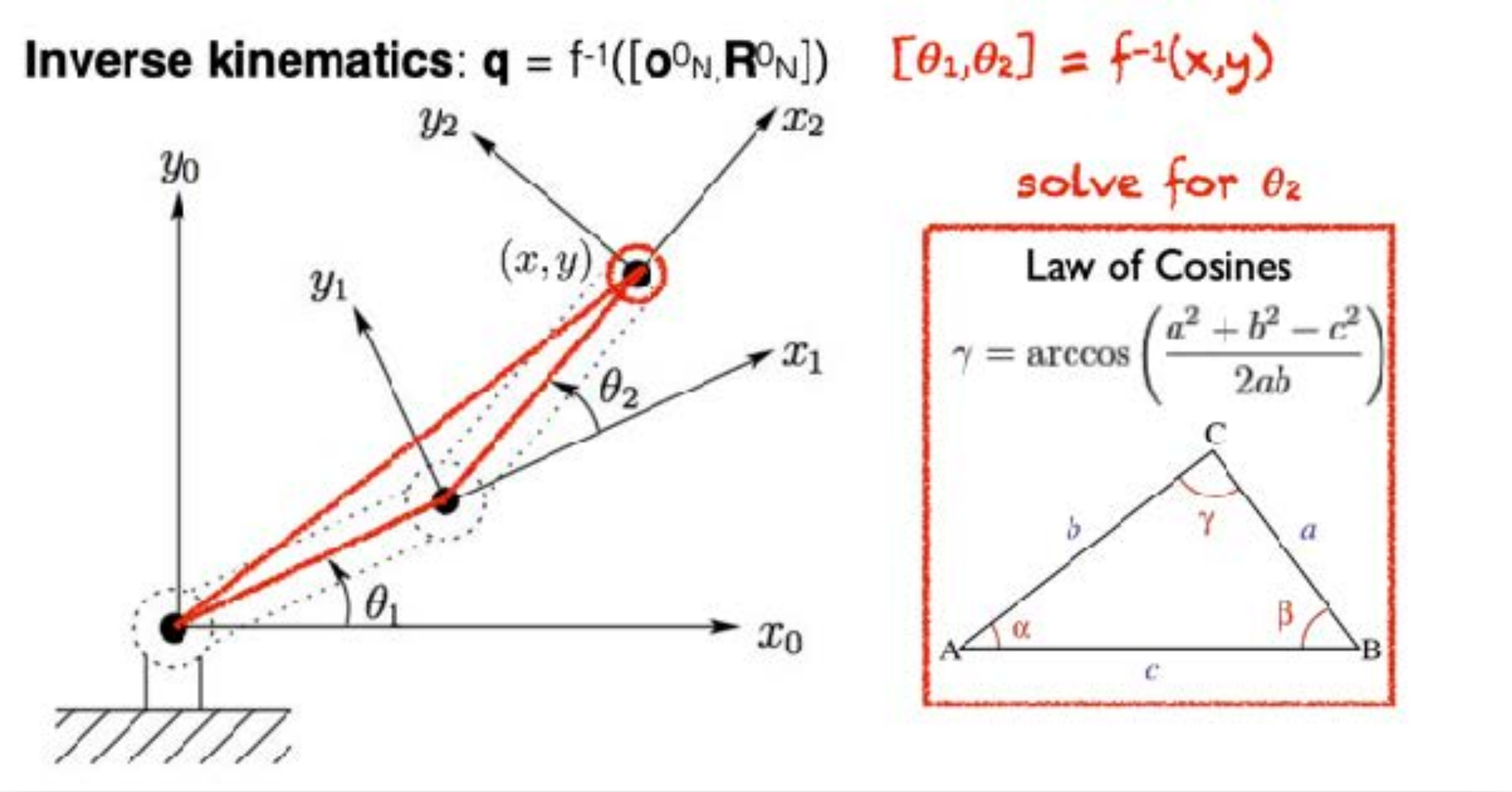
- Project 3 was posted on 02/07 and will be due 02/15 (~~not 02/14~~).
- Project 4 will be posted on 02/14 and will be due 02/28 (*yes 2 weeks*).
- Quiz 4 will be posted tomorrow at **noon** *(based on requests on Ed)* and will be due on Wed at noon.

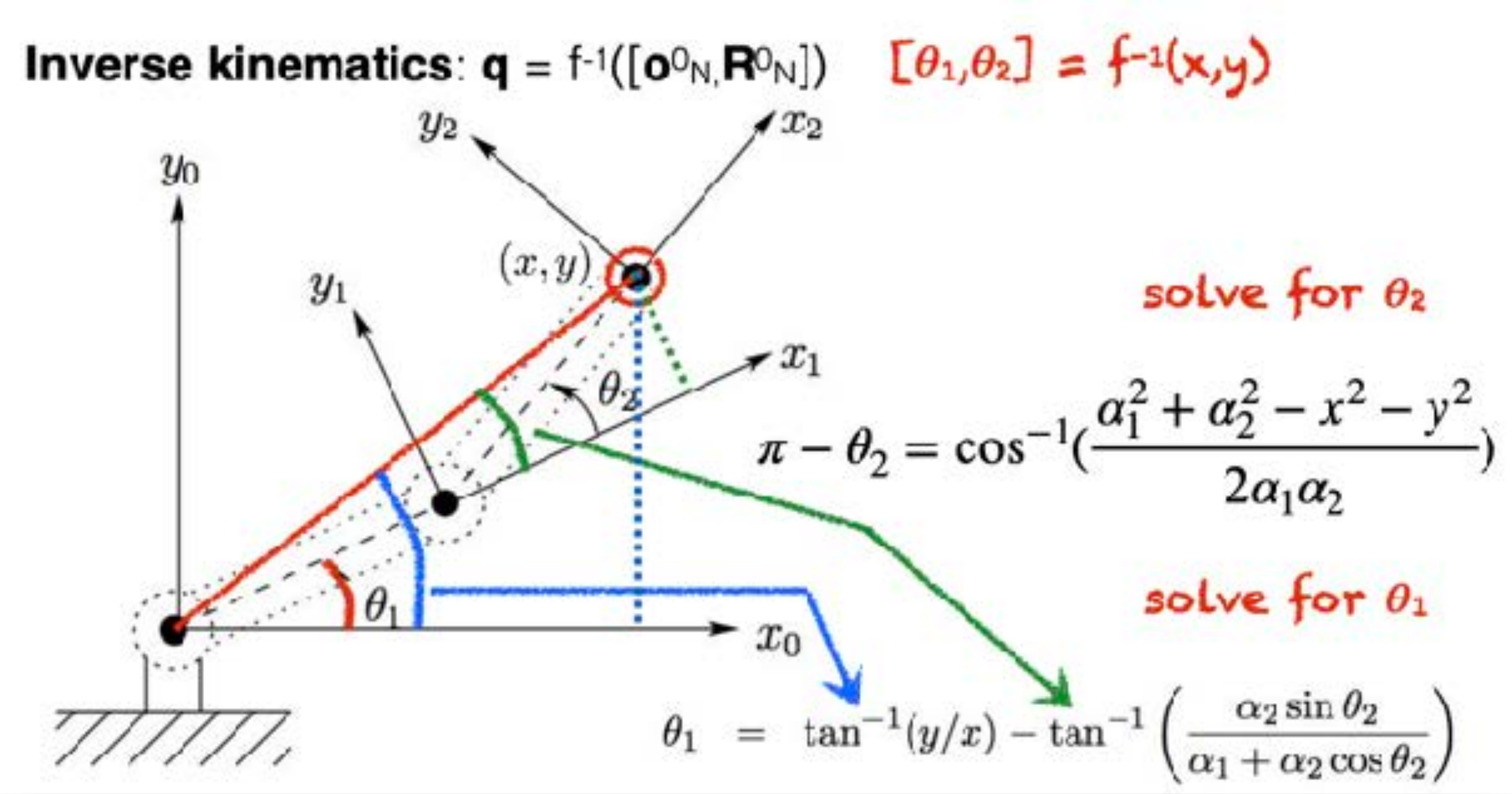
Previously

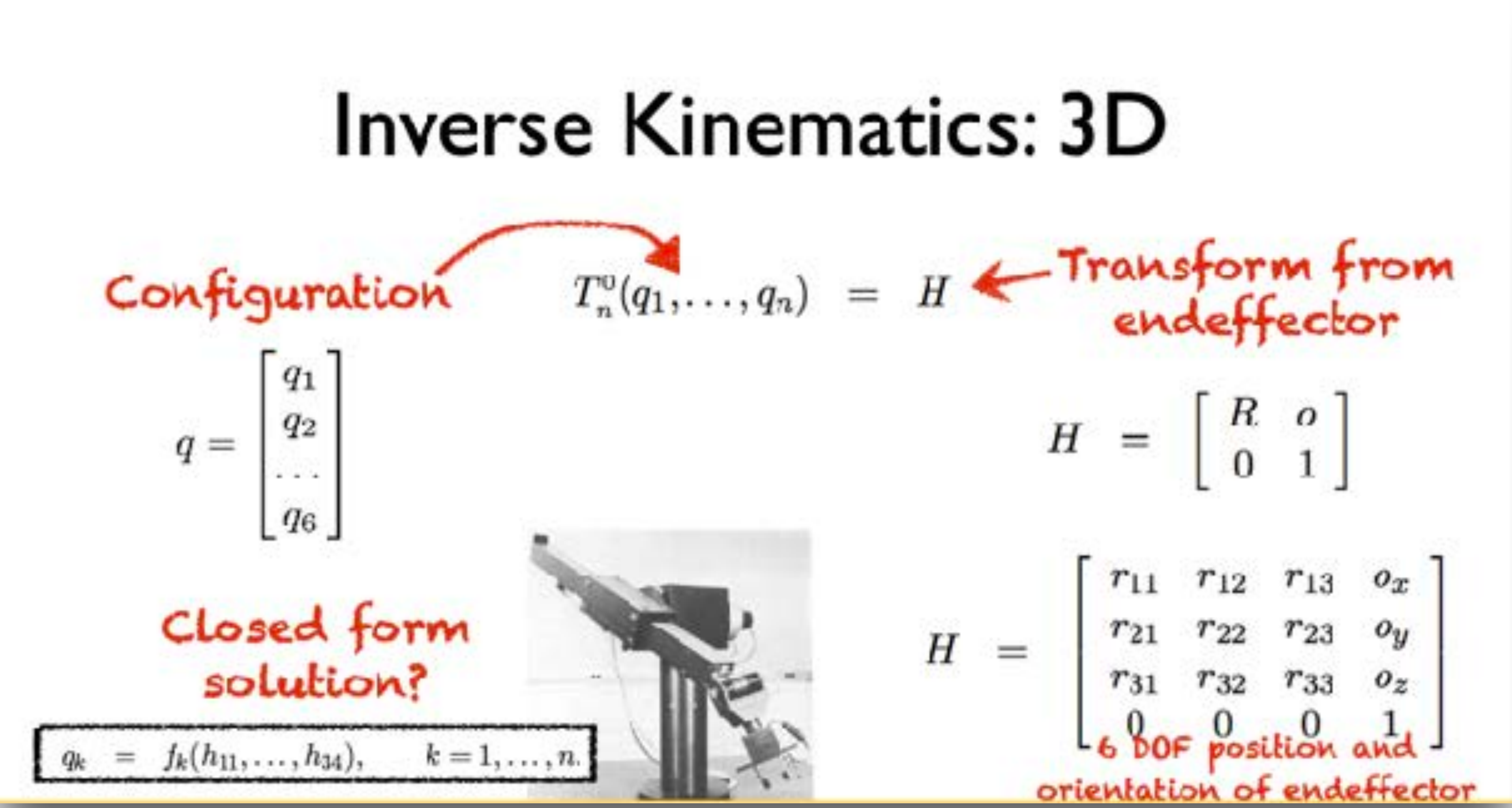




Inverse kinematics: how to solve for $q = \{\theta_1, \ldots, \theta_N\}$ from T^0_N?







Closed form solution?

$$q_k = f_k(h_{11}, \ldots, h_{34}), \quad k = 1, \ldots, n.$$

Why Closed Form?

- Advantages
  - Speed: IK solution computed in constant time
  - Predictability: consistency in selecting satisfying IK solution
- Disadvantage
  - Generality: general form for arbitrary kinematics difficult to express

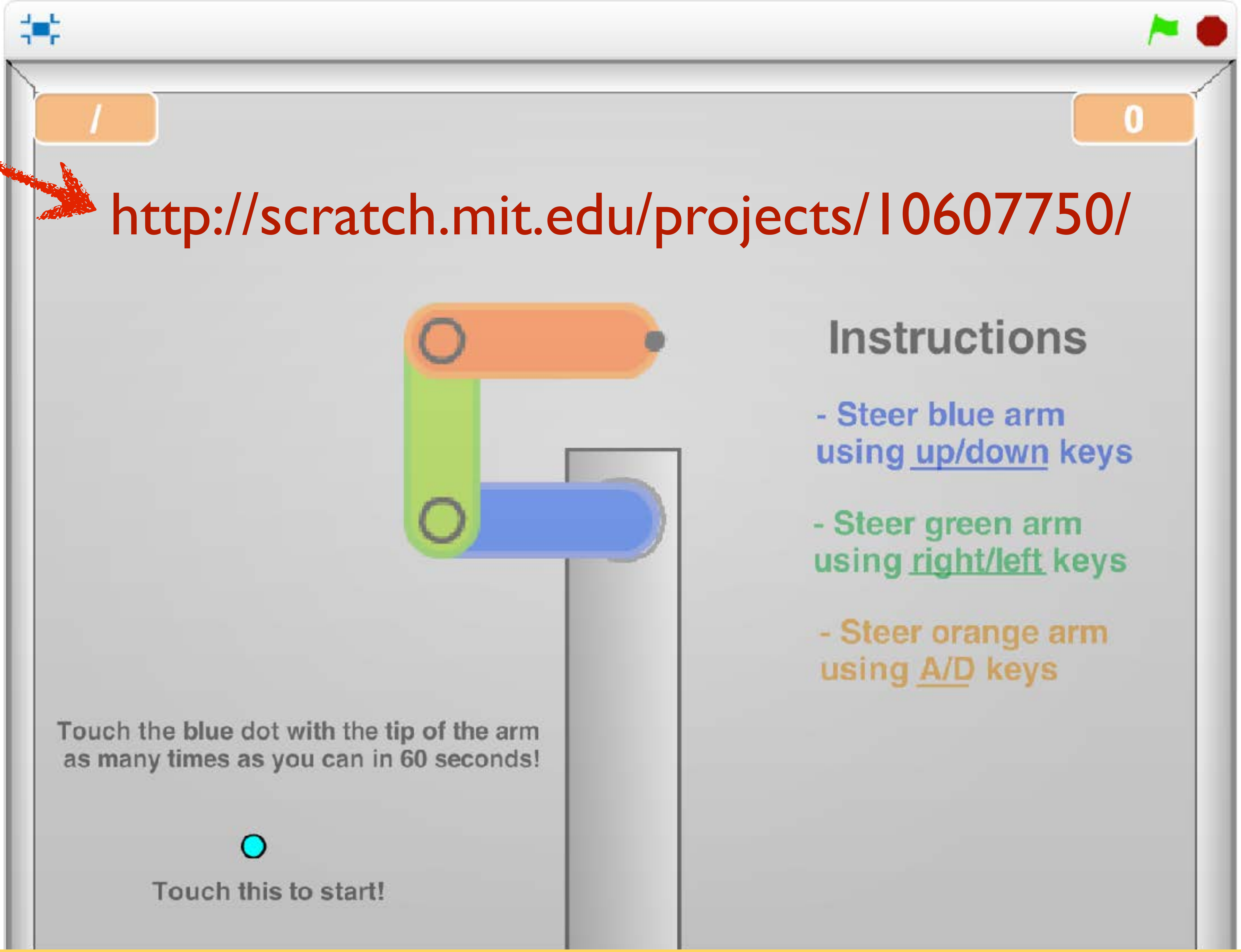
Inverse Kinematics: 2 possibilites

- **Closed-form solution**: geometrically infer satisfying configuration
  - *Speed*: solution often computed in constant time
  - *Predictability*: solution is selected in a consistent manner
- **Solve by optimization**: minimize error of endeffector to desired pose
  - often some form of Gradient Descent (a la Jacobian Transpose)
  - *Generality*: same solver can be used for many different robots

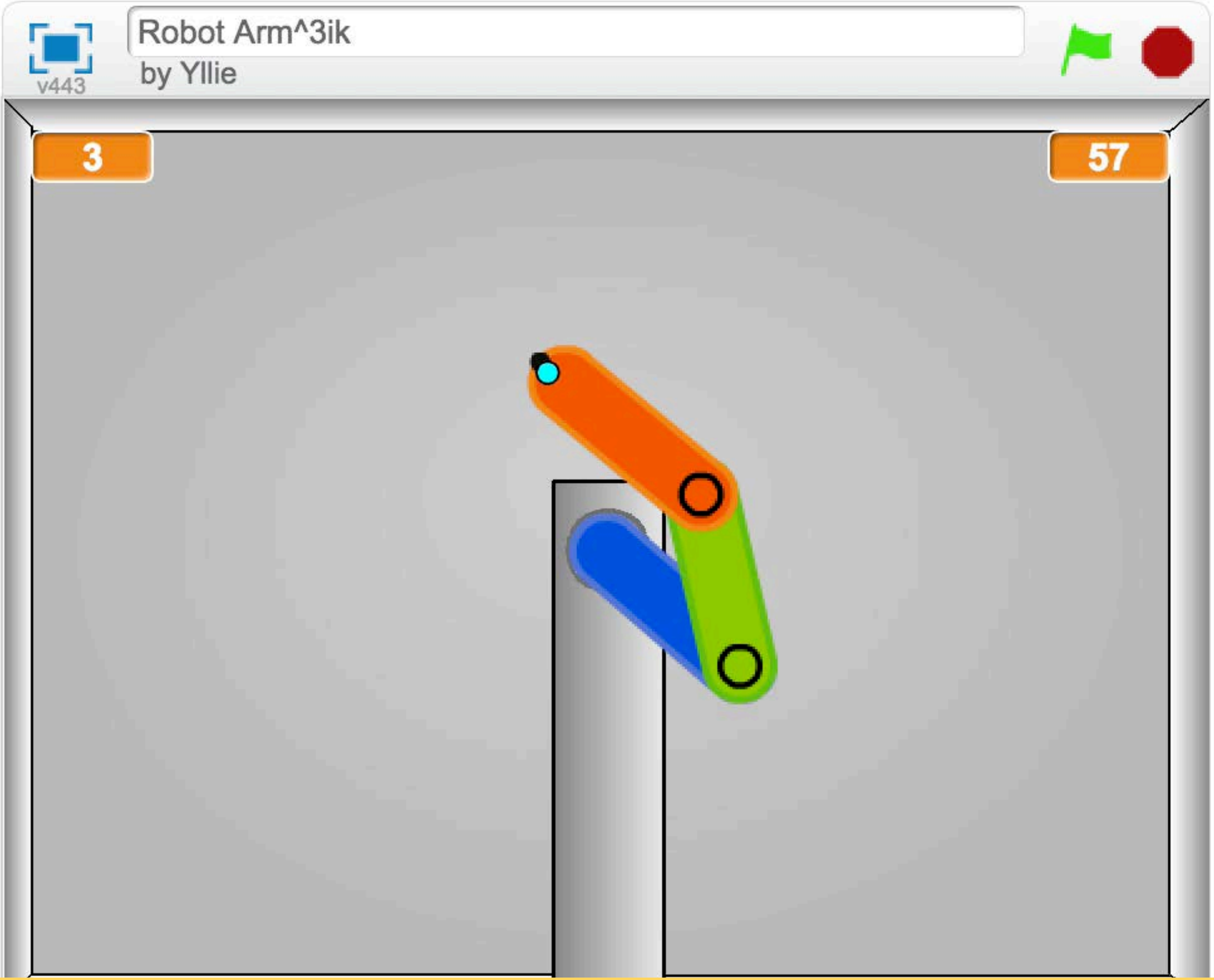
Anyone tried this?

Try this

http://scratch.mit.edu/projects/10607750/

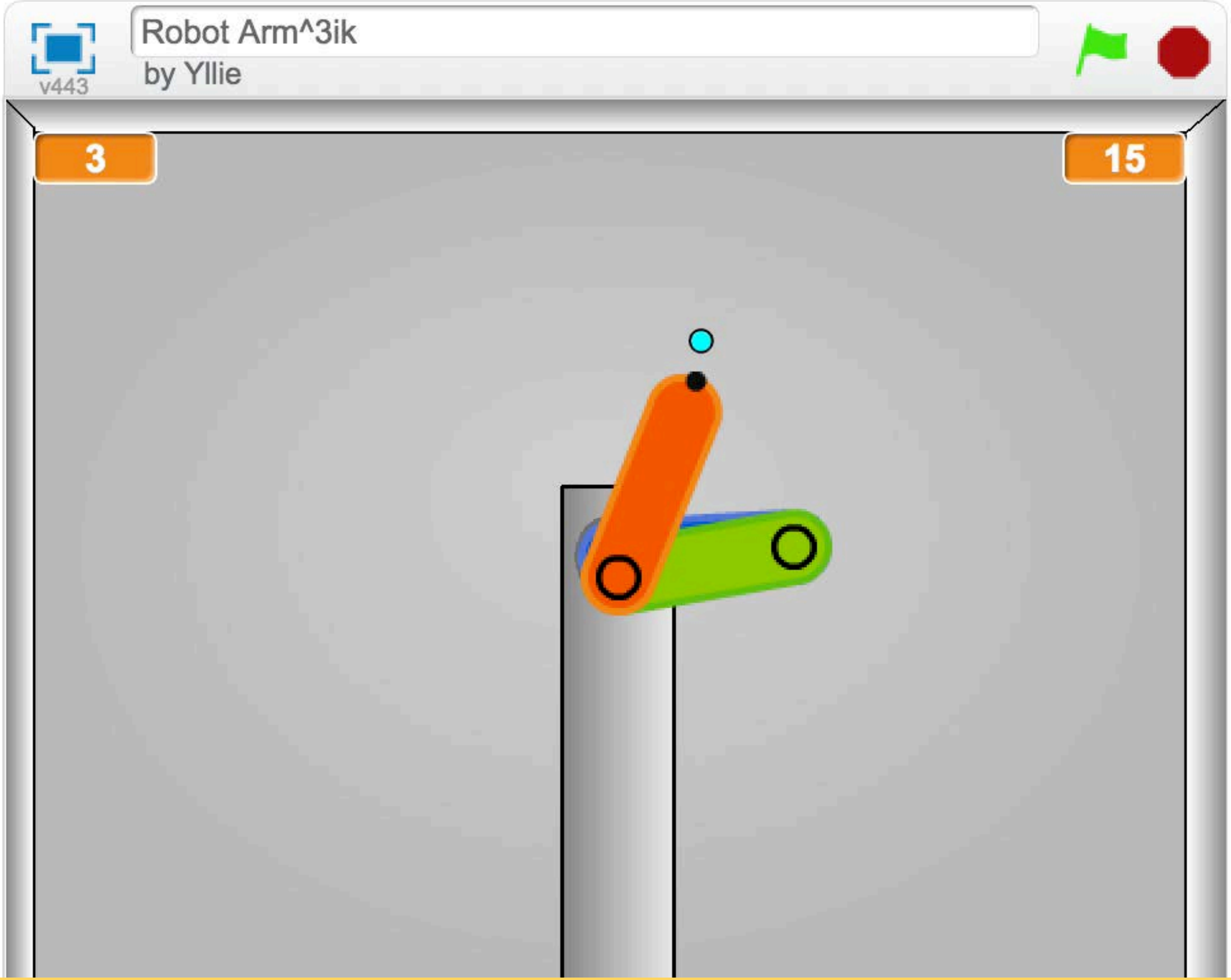


Aggressively tuned IK



By Dr. Jenkins

Conservatively tuned IK

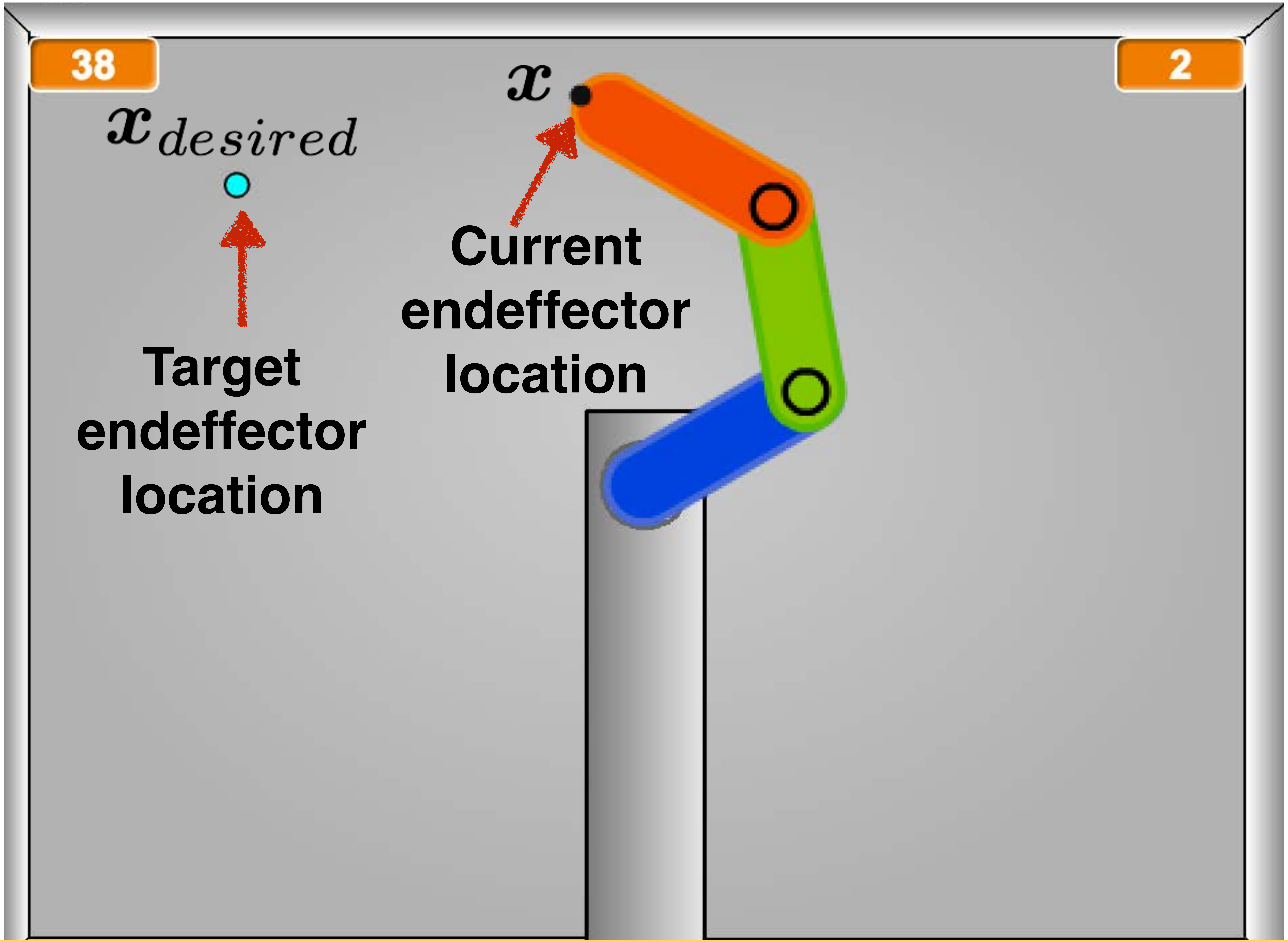


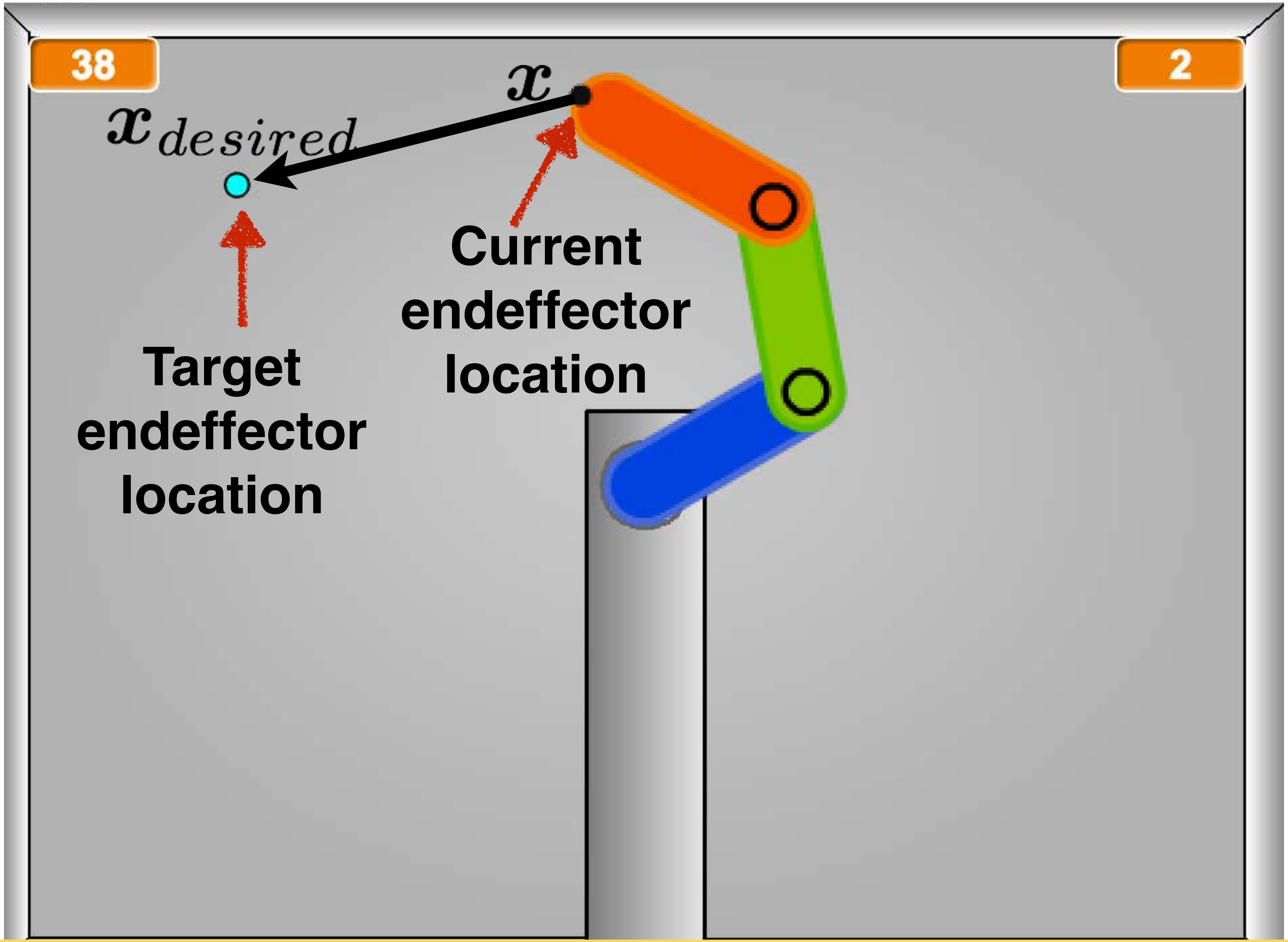
By Dr. Jenkins

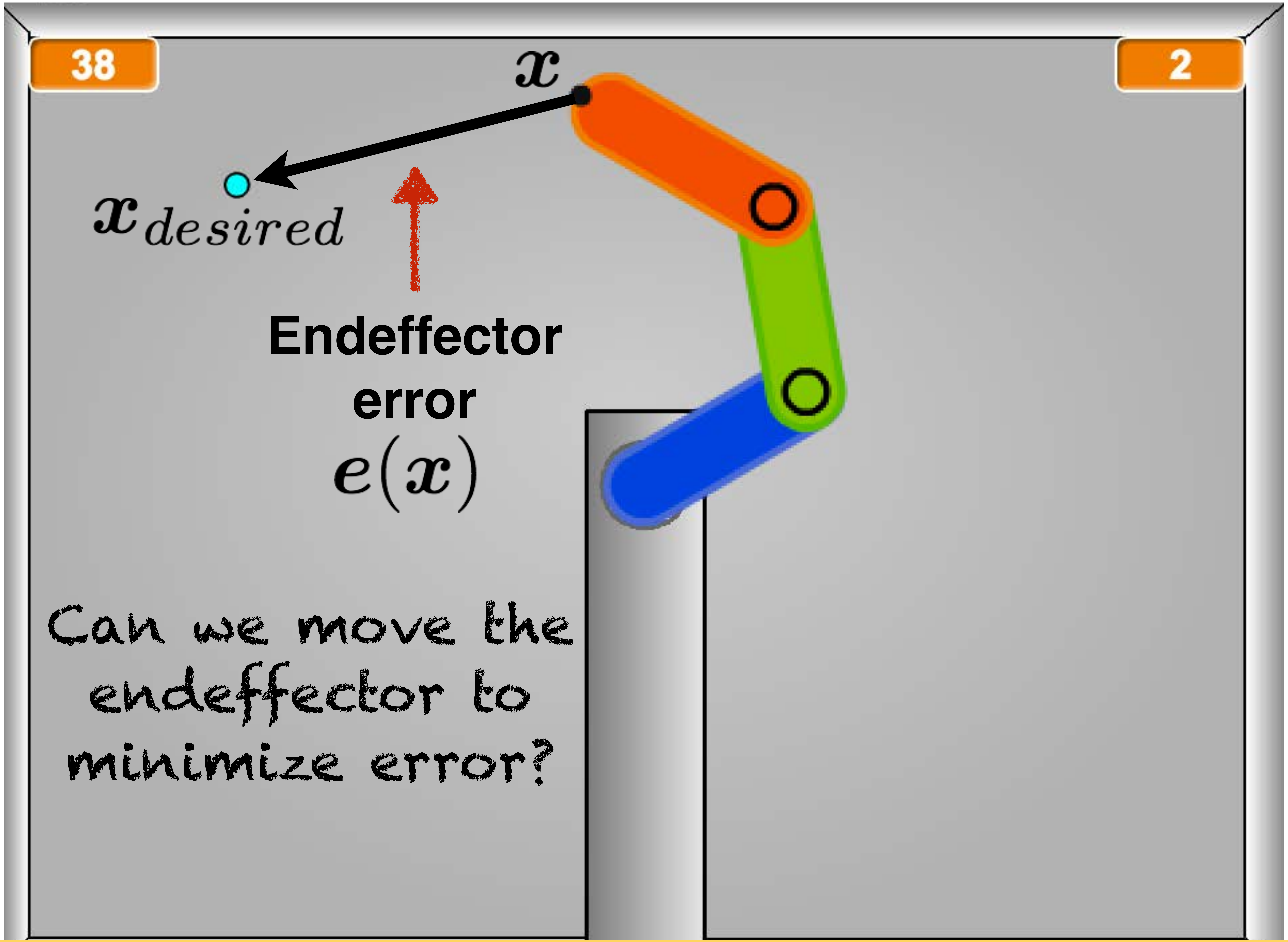
How to programmatically do this?

How to programmatically do this?

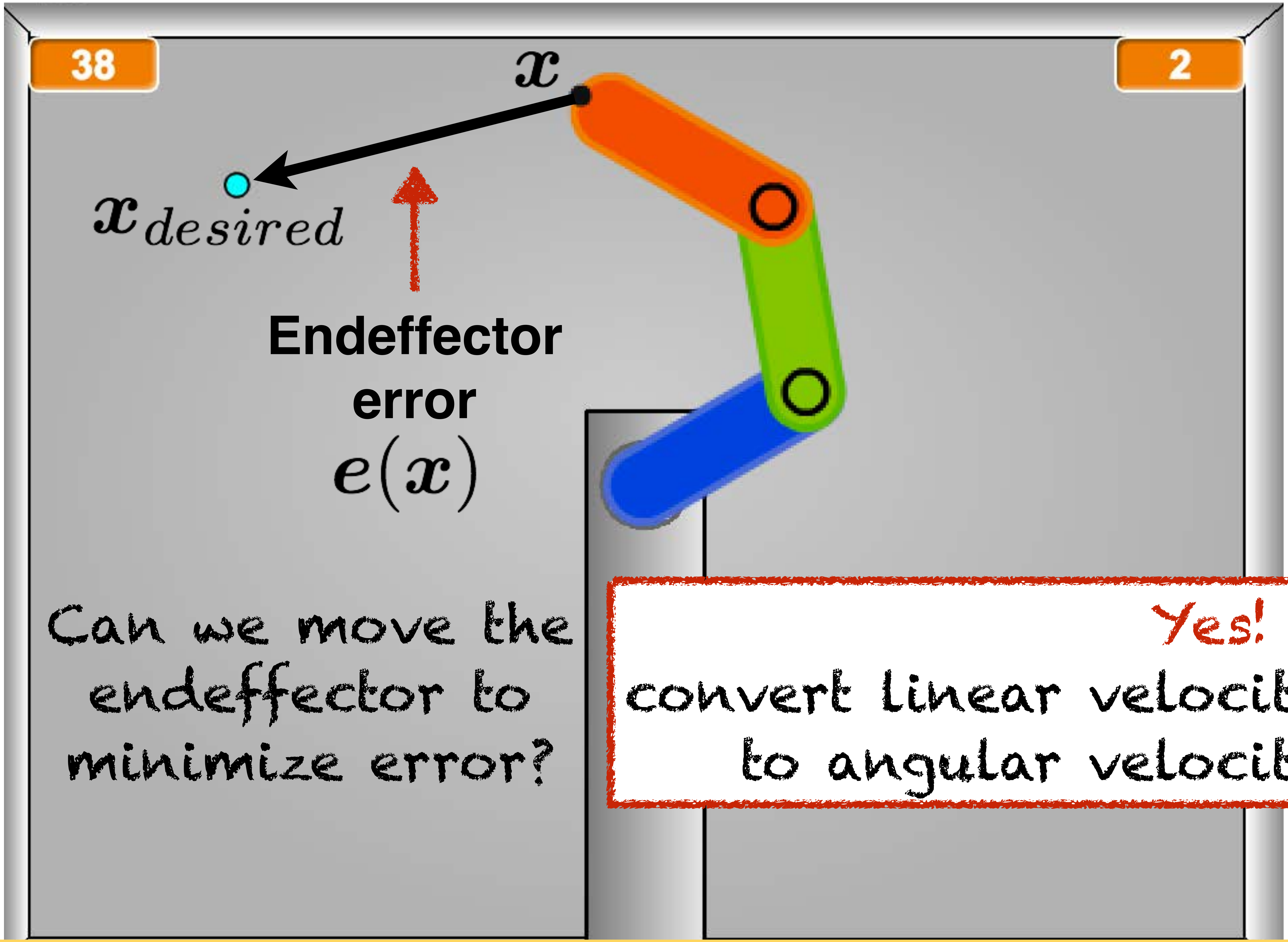
Jacobian Transpose


38
2
$x_{desired}$
x
Target endeffector location
Current endeffector location


38
2
x
$x_{desired}$
Current endeffector location
Target endeffector location



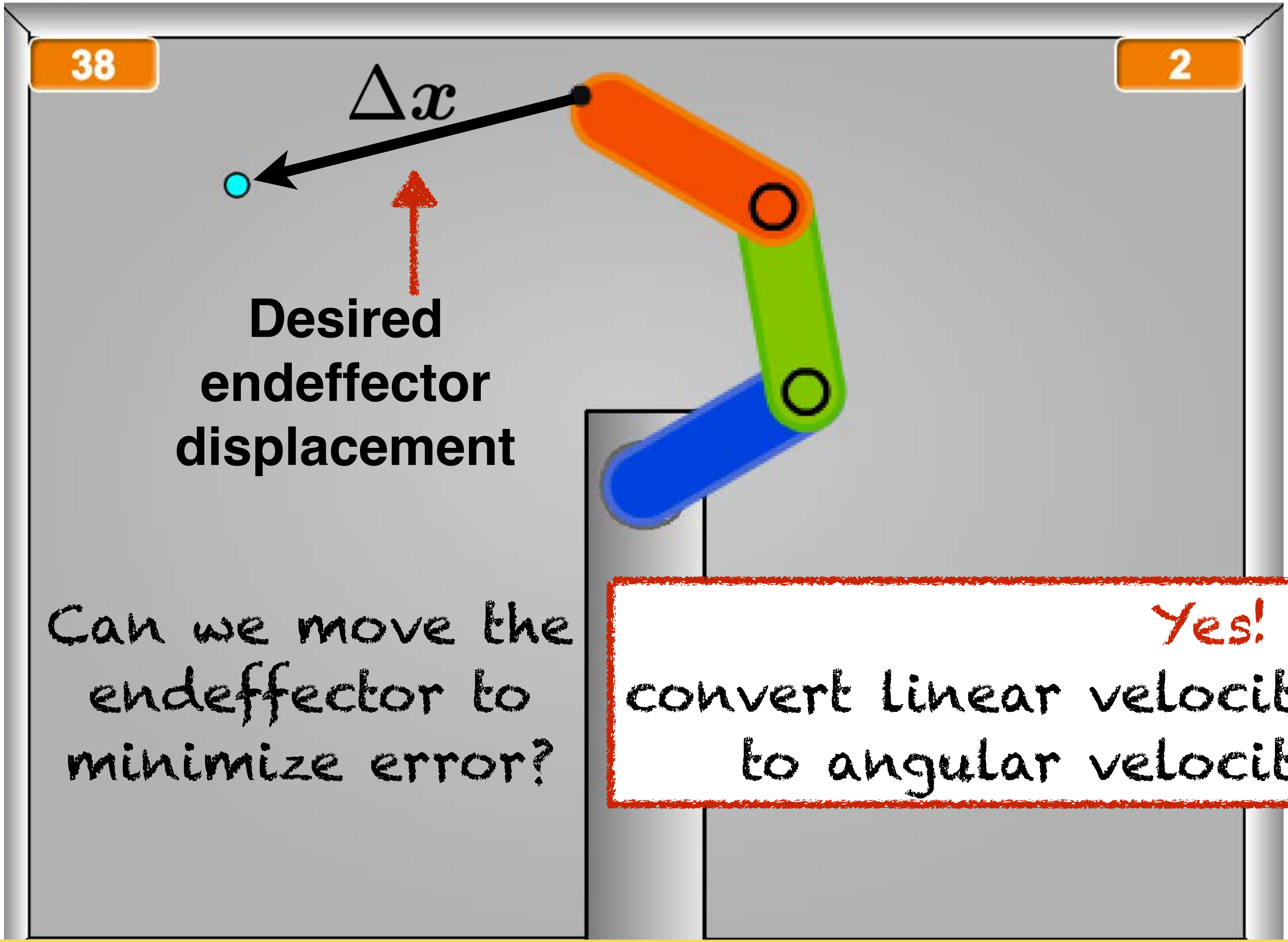
Slide borrowed from Michigan Robotics autorob.org



Can we move the endeffector to minimize error?

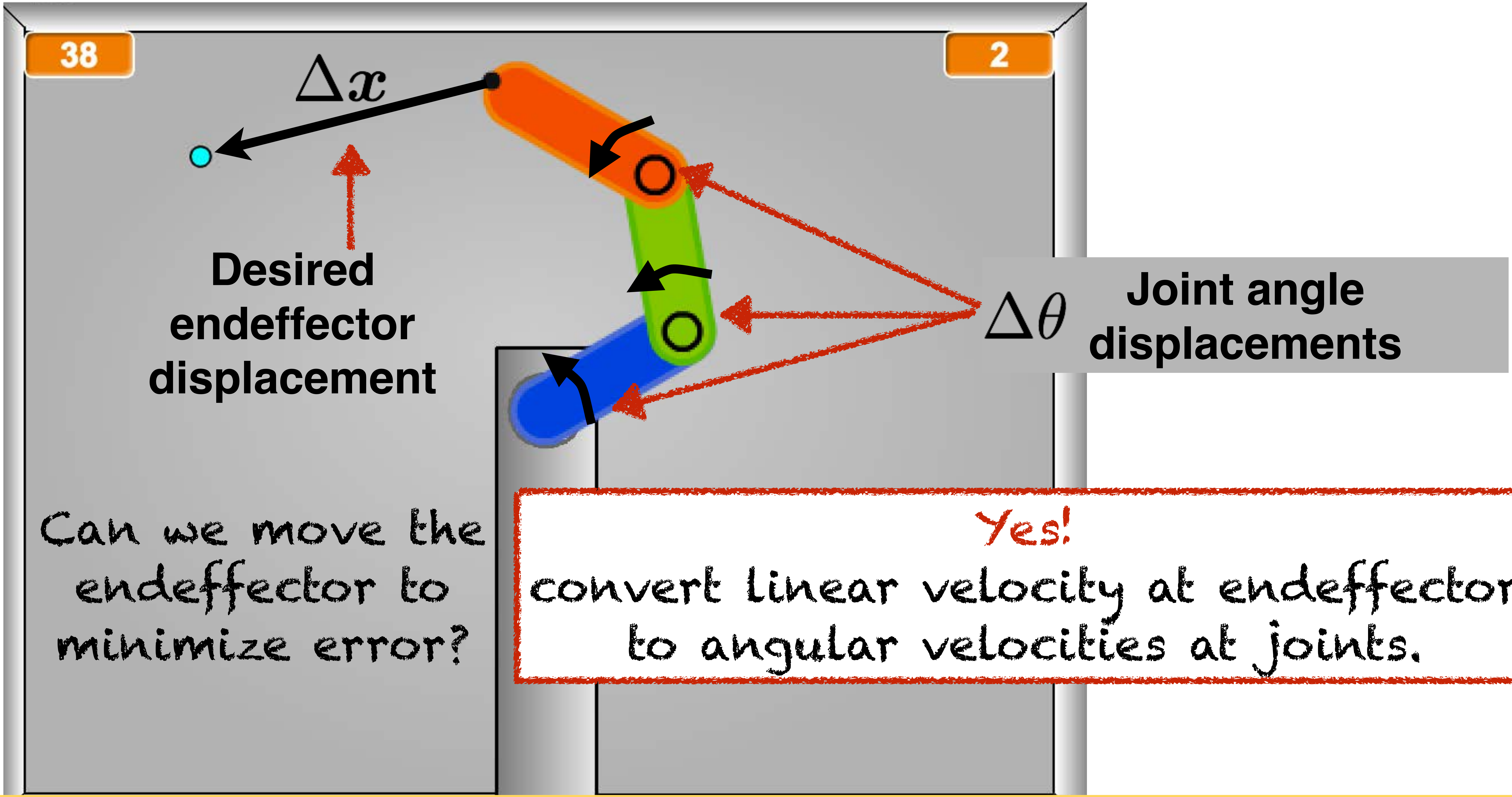
Yes!

convert linear velocity at endeffector to angular velocities at joints.



Can we move the endeffector to minimize error?

Yes!
convert linear velocity at endeffector to angular velocities at joints.



Can we move the endeffector to minimize error?

Yes!
convert linear velocity at endeffector to angular velocities at joints.

How are linear and angular velocity related?

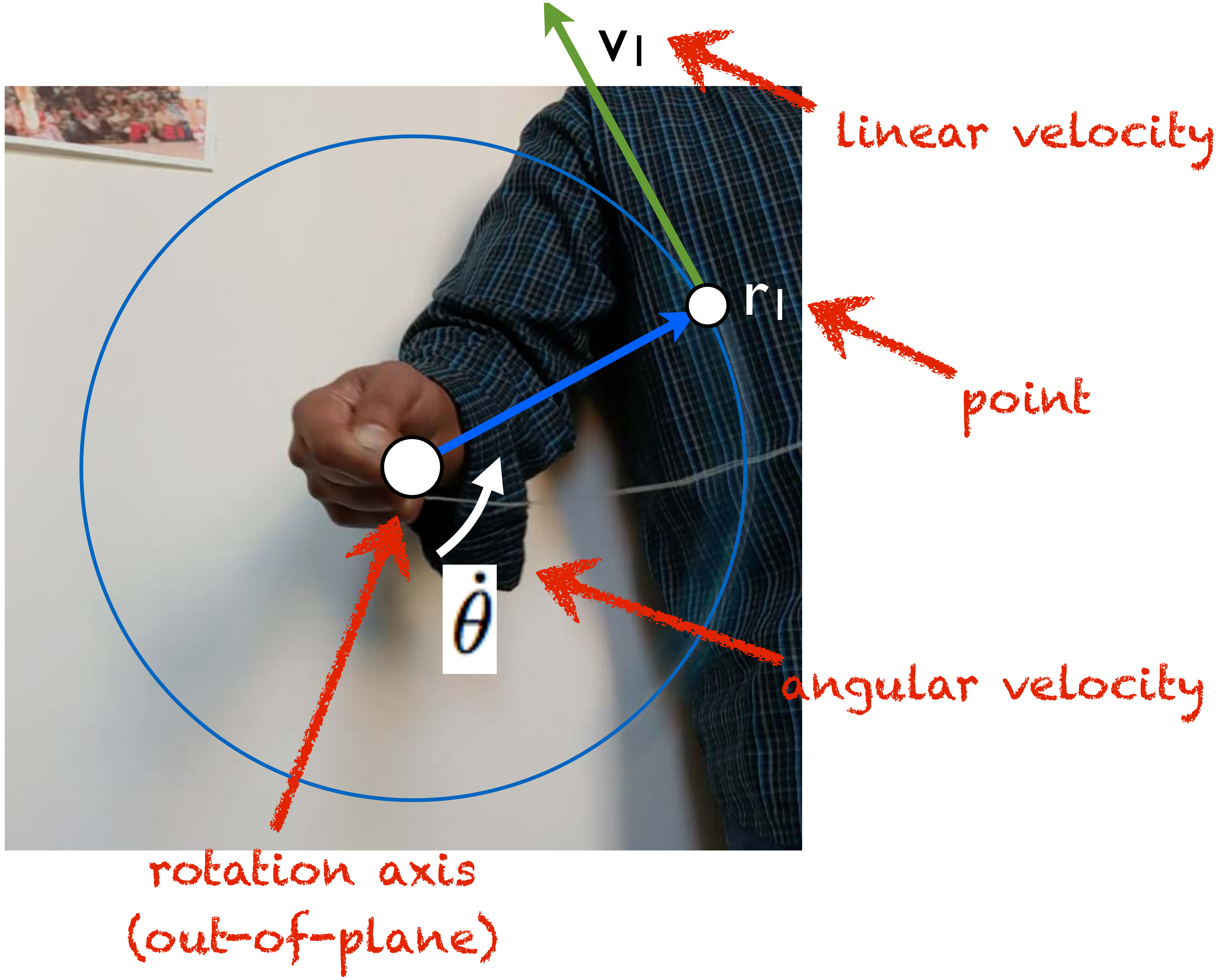
How are linear and angular velocity related?

Consider the velocity of a point

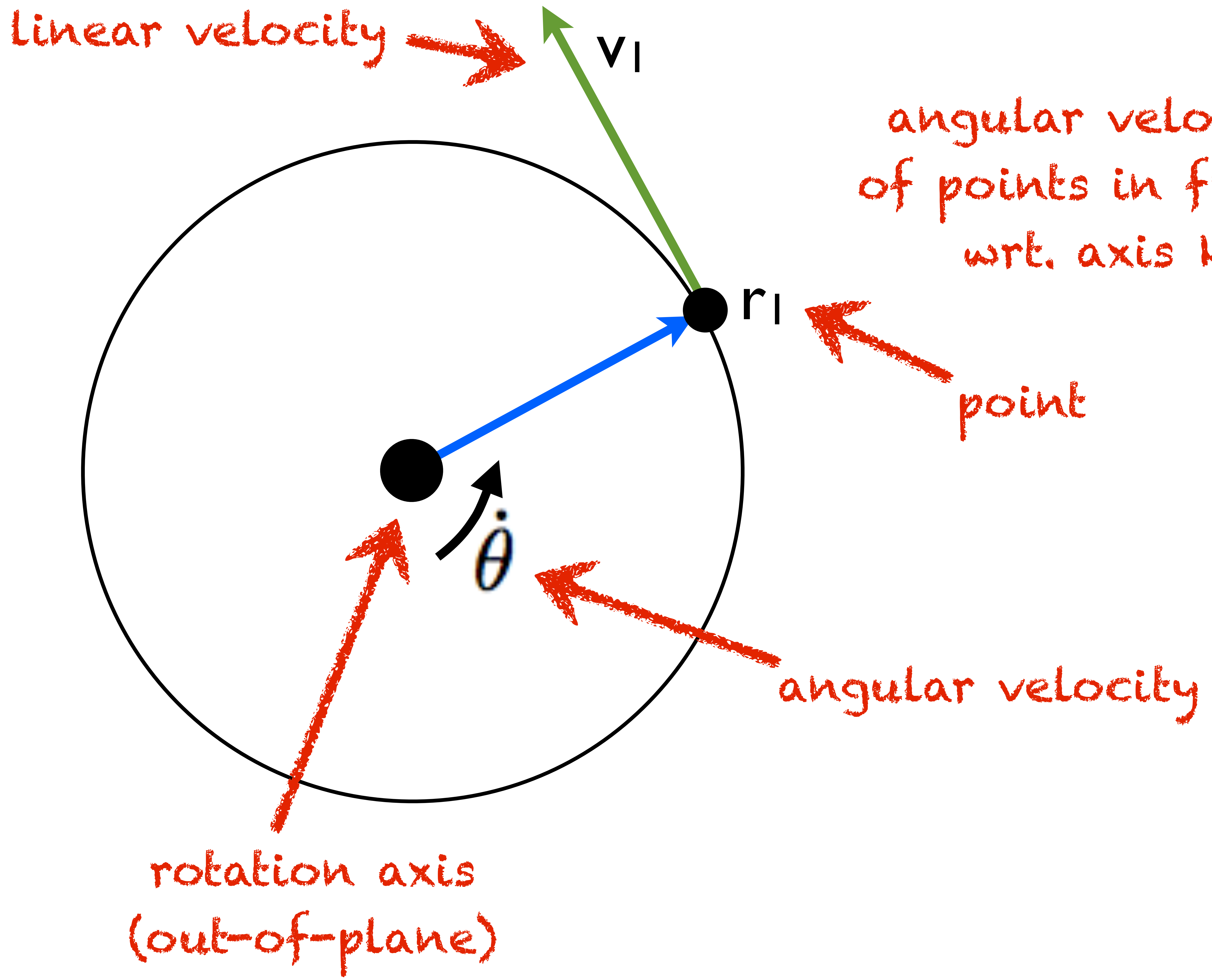
Consider the velocity of a point

Consider the velocity of a point

Velocity of Point Rotating in Fixed Frame



Velocity of Point Rotating in Fixed Frame



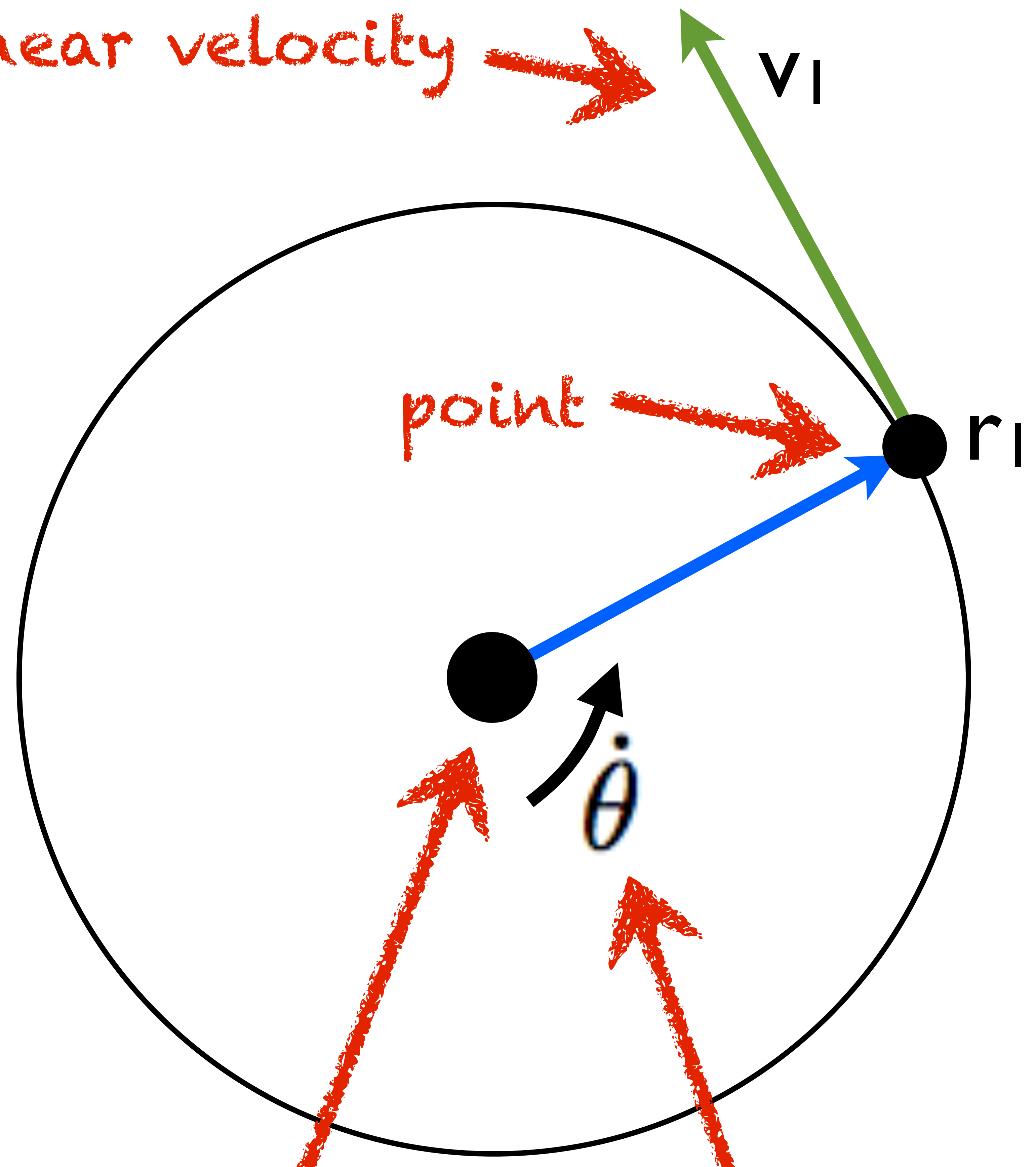
$$\omega = \dot{\theta}\boldsymbol{k}$$

angular velocity of points in frame wrt. axis k

rotation speed of frame

rotation axis

Velocity of Point Rotating in Fixed Frame



$$\omega = \dot{\theta}\boldsymbol{k}$$

angular velocity of points in frame wrt. axis k

rotation speed of frame

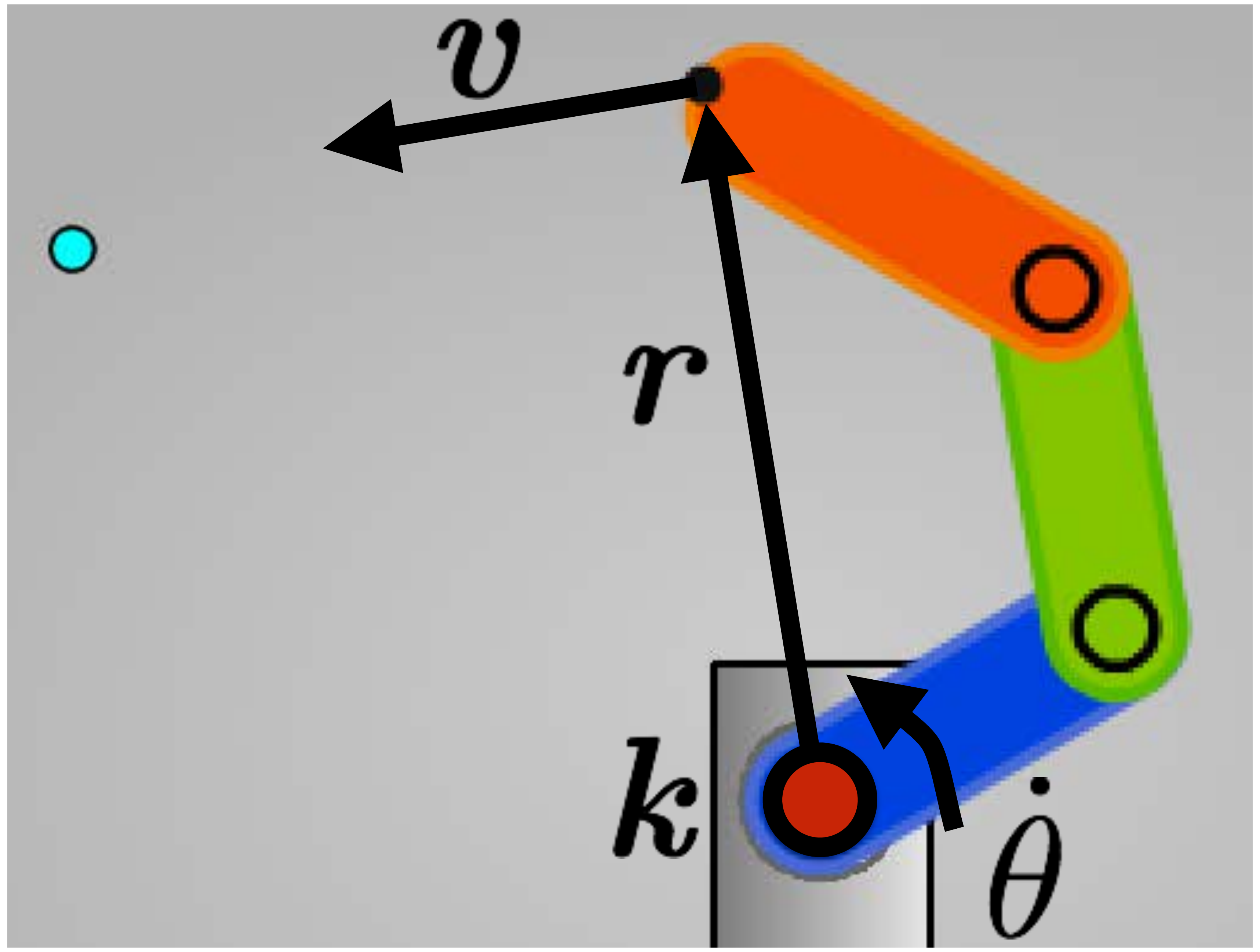
rotation axis

$$\boldsymbol{v} = \omega \times \boldsymbol{r}$$

linear velocity of points in frame wrt. axis k

vector to point in frame

Velocity of Point Rotating in Fixed Frame



$$\omega = \dot{\theta}\boldsymbol{k}$$

angular velocity of points in frame wrt. axis k

rotation speed of frame

rotation axis

$$\boldsymbol{v} = \omega \times \boldsymbol{r}$$

linear velocity of points in frame wrt. axis k

vector to point in frame

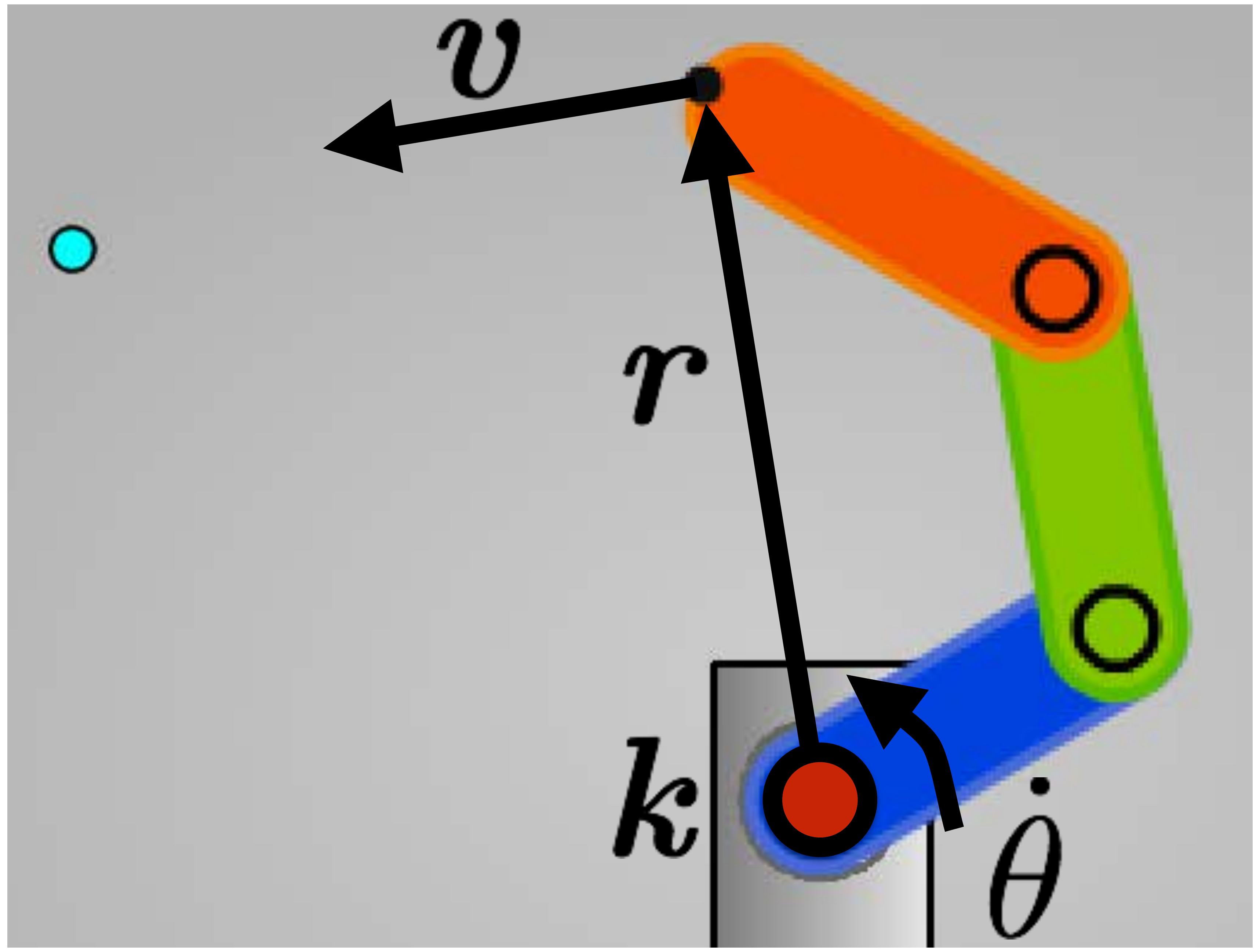
$$\boldsymbol{v} = \dot{\theta}\boldsymbol{k} \times \boldsymbol{r}$$

endeffector linear velocity

joint rotation axis

vector from joint origin to endeffector

Slide borrowed from Michigan Robotics autorob.org



$$\boldsymbol{v} = \dot{\theta}\boldsymbol{k} \times \boldsymbol{r}$$

endeffector linear velocity

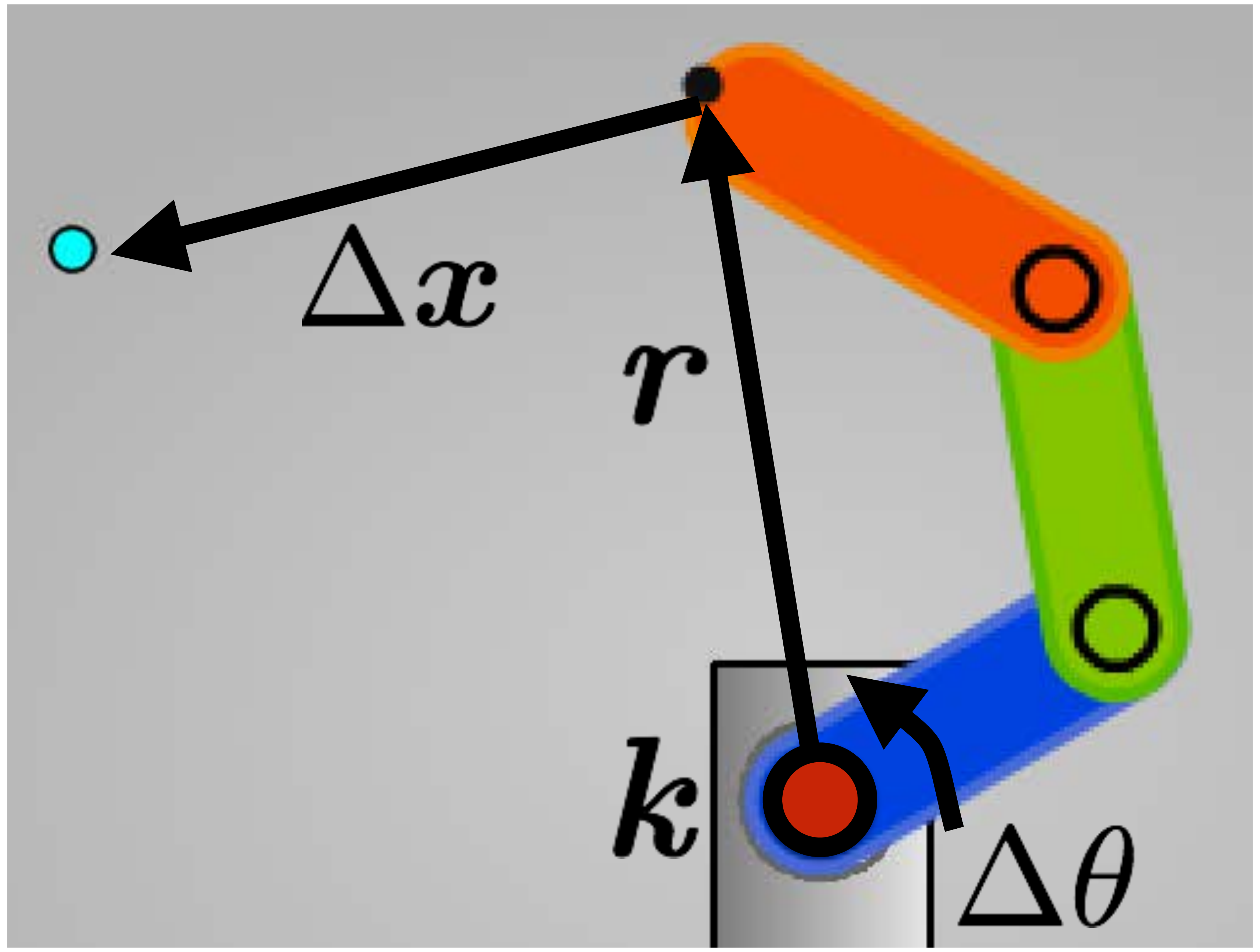
joint rotation axis

vector from joint origin to endeffector

This is not what we wanted.

Why?

Jacobian Transpose



vector from joint origin to endeffector

$$\boldsymbol{v} = \dot{\theta}\boldsymbol{k} \times \boldsymbol{r}$$

endeffector linear velocity

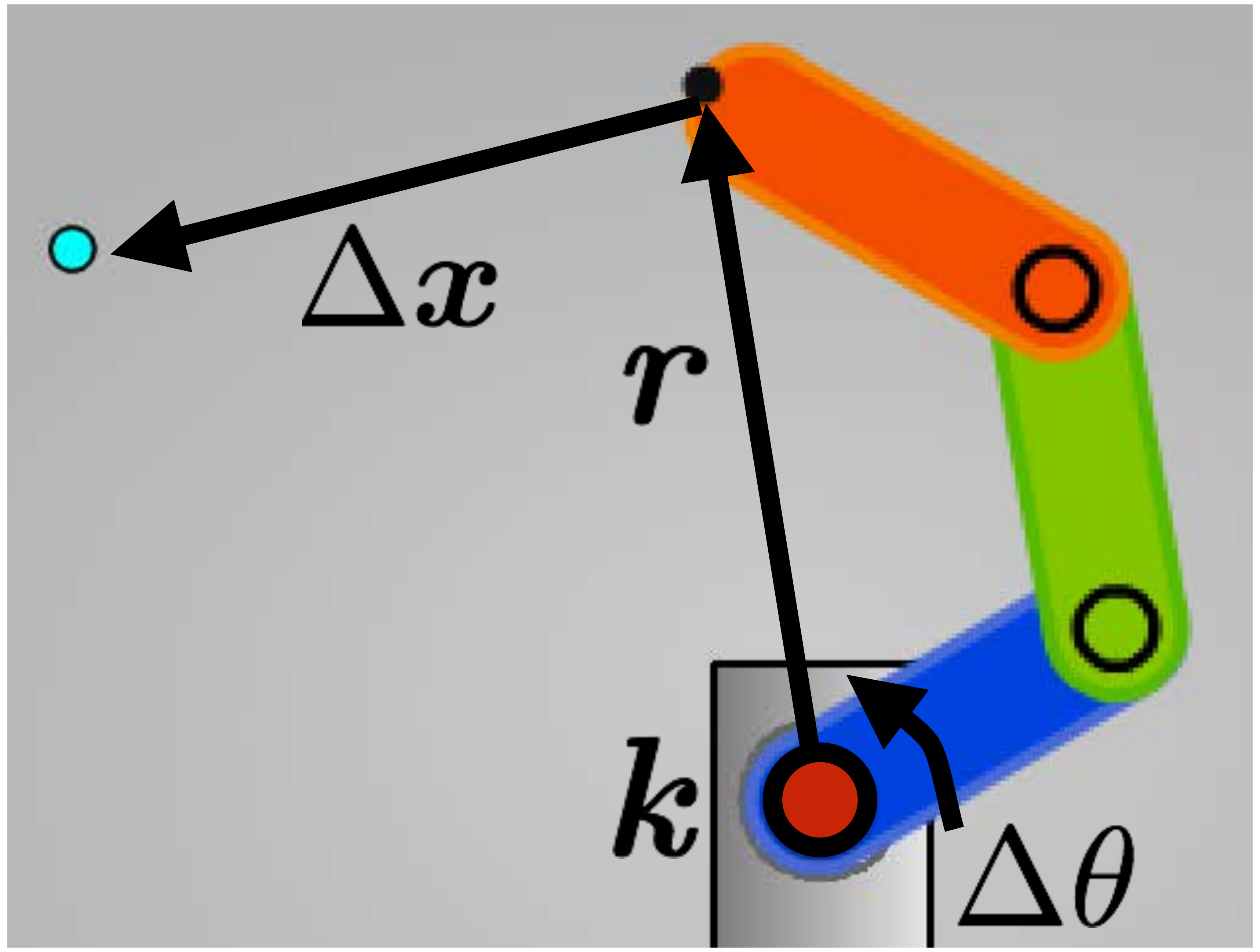
joint rotation axis

This is not what we wanted.

How to obtain joint angular velocity from endeffector linear velocity?

[redacted]

Jacobian Transpose



$$v = \dot{\theta} k \times r$$

vector from joint origin to endeffector

endeffector linear velocity

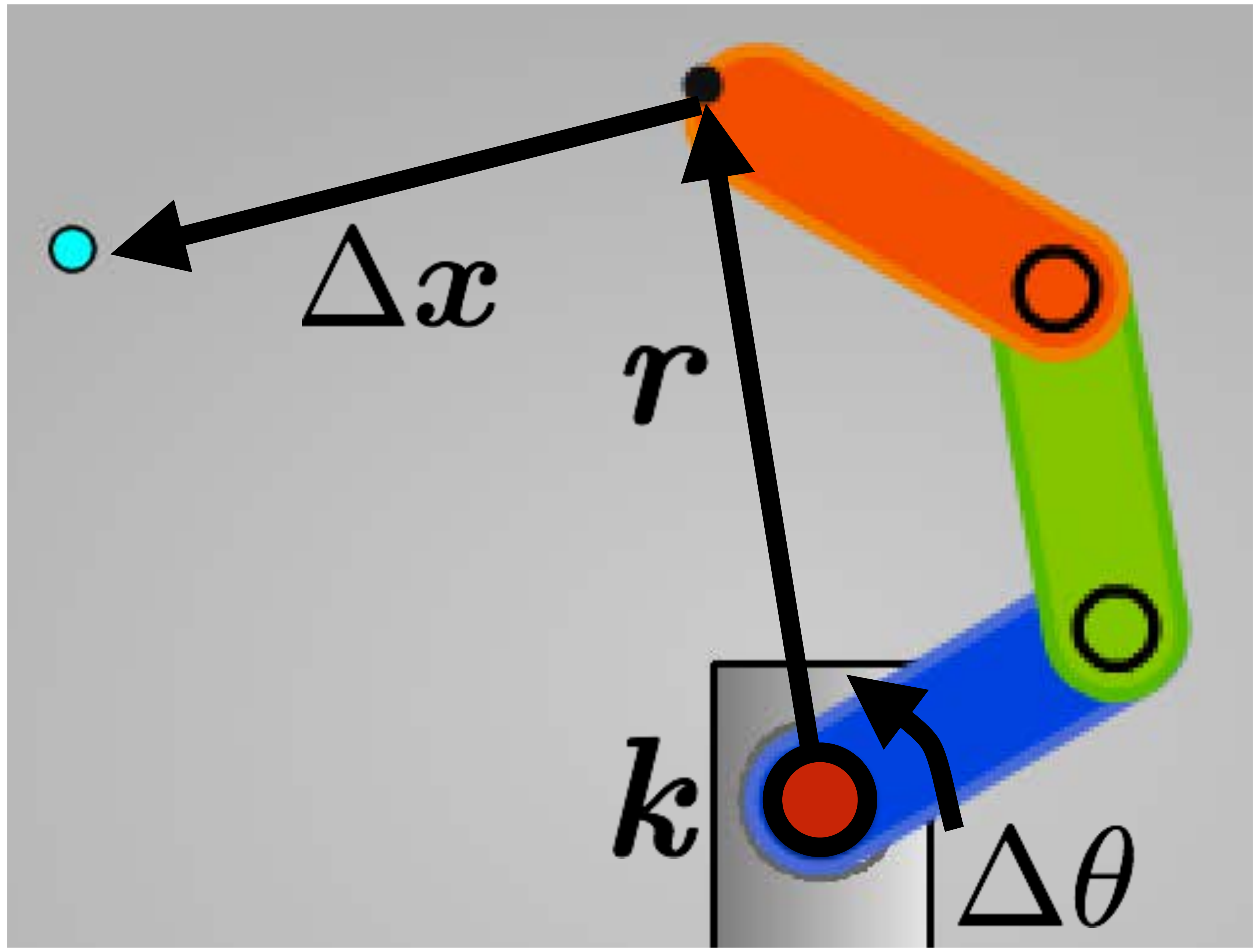
joint rotation axis

This is not what we wanted.

How to obtain joint angular velocity from endeffector linear velocity?

$$\Delta\theta = (k \times r)^T \Delta x$$

Jacobian Transpose



joint rotation axis

vector from joint origin to endeffector

$$\Delta\theta = (\underbrace{\boldsymbol{k} \times \boldsymbol{r}})^T \Delta x$$

Angular displacement for joint i

Jacobian for joint i

desired endeffector displacement

Procedure (for each joint):

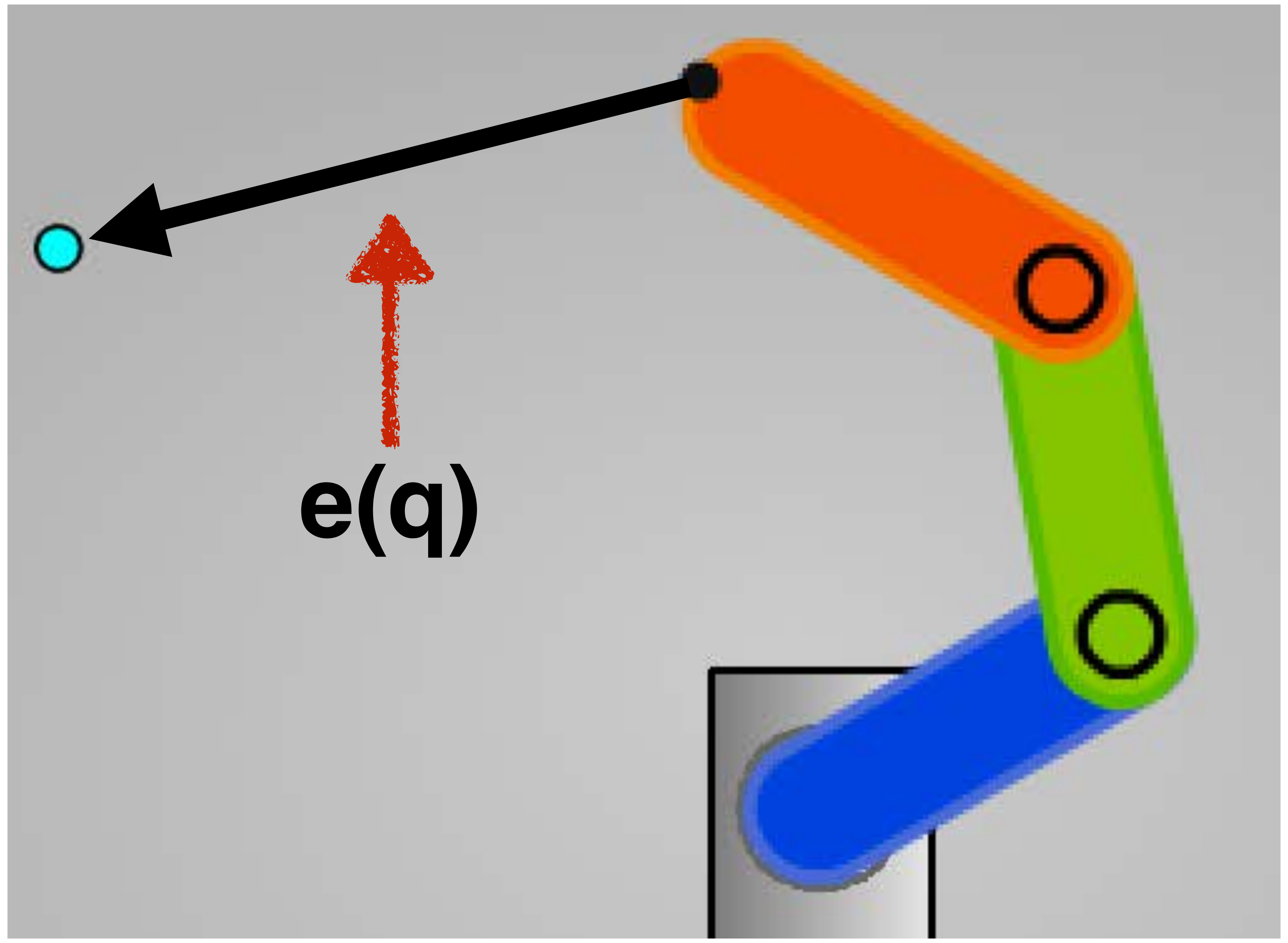
1) Compute Jacobian
2) Update joint angles using Jacobian transpose
3) Repeat forever (or until error minimized)

IK as Error Minimization

IK as Error Minimization

Gradient Descent Optimization

Inverse kinematics as error minimization

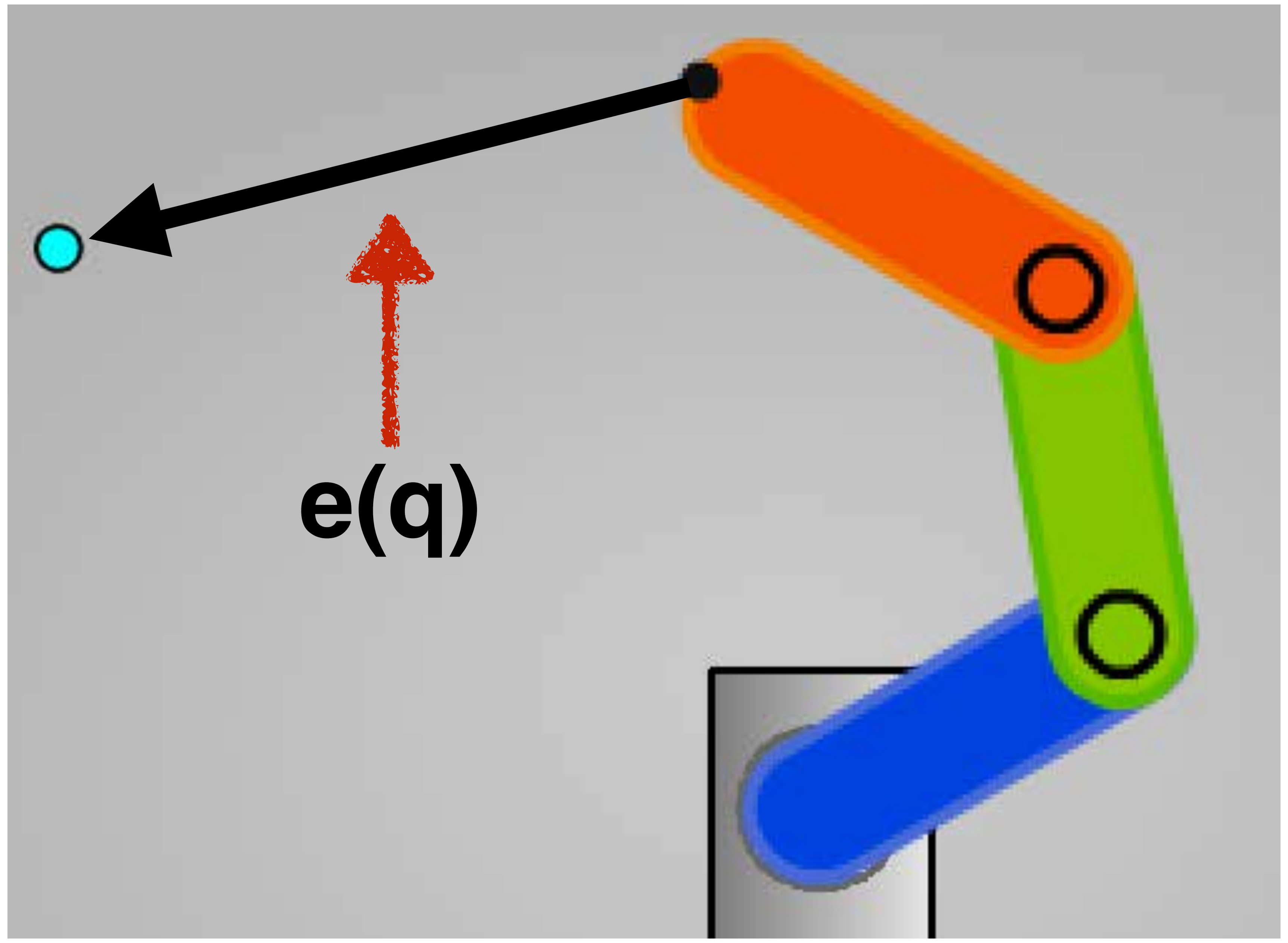


Define error function $e(\boldsymbol{q})$ as difference between current and desired endeffector poses

Error function parameterized by robot configuration $\boldsymbol{q}$

Find global minimum of $e(\boldsymbol{q})$: $\operatorname{argmin}_{\boldsymbol{q}}\, e(\boldsymbol{q})$

Inverse kinematics as error minimization



Define error function $e(\boldsymbol{q})$ as difference between current and desired endeffector poses

Error function parameterized by robot configuration $\boldsymbol{q}$

Find global minimum of $e(\boldsymbol{q})$: $\mathrm{argmin}_{\boldsymbol{q}}\ e(\boldsymbol{q})$

How could we find $\mathrm{argmin}_{\boldsymbol{q}}\ e(\boldsymbol{q})$ if we knew $e(\boldsymbol{q})$ in closed form?

Example: Find global minimum of function

$$f(x) = 3(x-2)^2$$

Take derivative

[redacted]

Solve for *x* where derivative is zero

[redacted]

Verify

[redacted]

Example: Find global minimum of function

$$f(x) = 3(x-2)^2$$

Take derivative

$$\frac{df}{dx} = 6(x-2)1$$

Solve for *x* where derivative is zero

[redacted]

Verify [redacted]

Example: Find global minimum of function

$$f(x) = 3(x-2)^2$$

Take derivative

$$\frac{df}{dx} = 6(x-2)1$$

Solve for *x* where derivative is zero

$$6(x-2) = 0$$

$$6x - 12 = 0$$

$$x = 2$$

Verify [redacted]

Example: Find global minimum of function

$$f(x) = 3(x-2)^2$$

Take derivative

$$\frac{df}{dx} = 6(x-2)1$$

Solve for *x* where derivative is zero

$$6(x-2) = 0$$

$$6x - 12 = 0$$

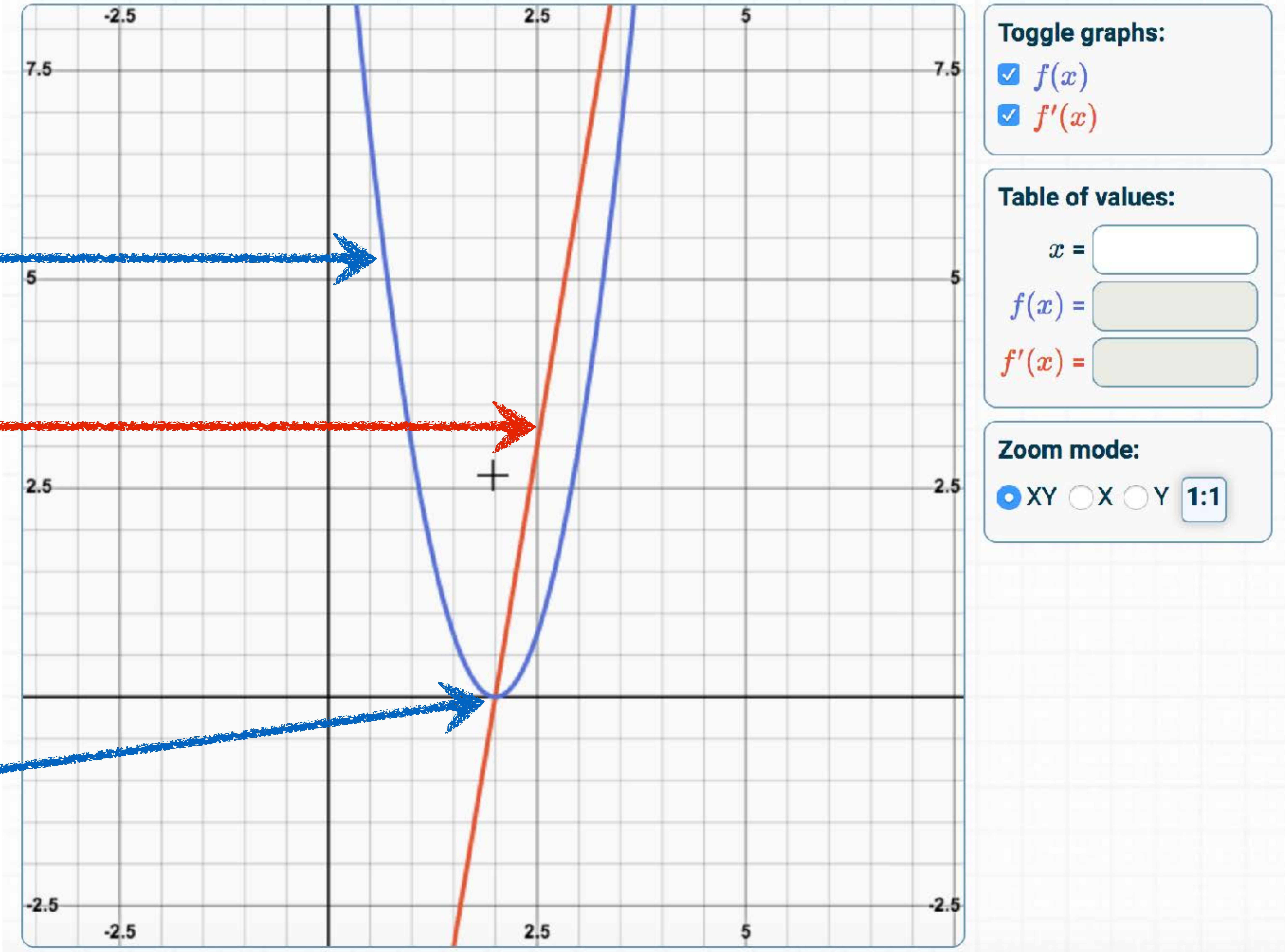
$$x = 2$$

Verify $$f(2) = 3((2)-2)^2 = 0$$

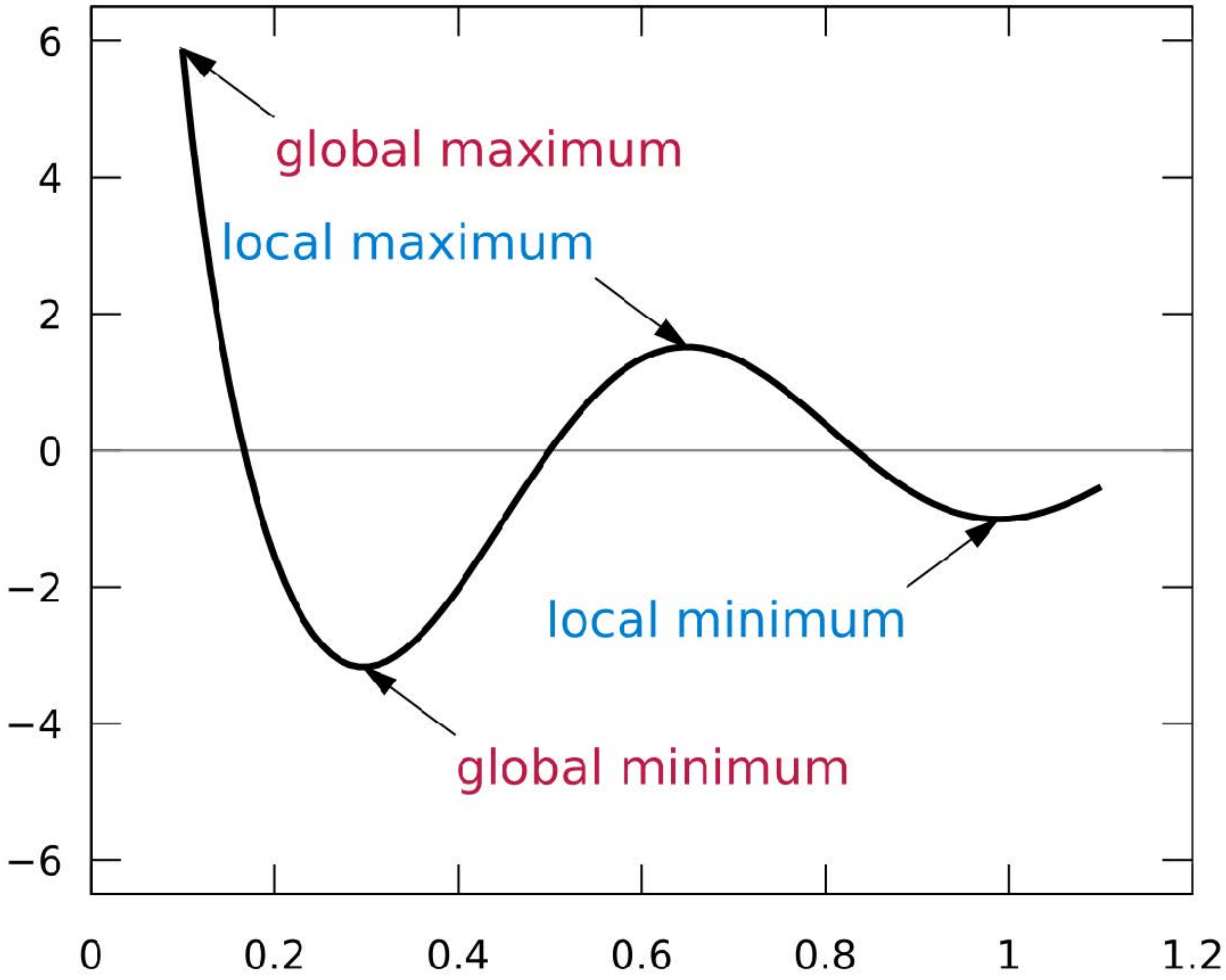
$$f(x) = 3(x-2)^2$$

$$\frac{df}{dx} = 6(x-2)1$$

$$f(2) = 3((2)-2)^2 = 0$$



Example: $\cos(3\pi x)/x$, $0.1 \leq x \leq 1.1$



Slide borrowed from Michigan Robotics autorob.org

derivative-calculator.net

= Check your own answe

= Export the expression (

commendation

Calculus for Dummies (2nd Edition)

An extremely well-written book for students taking Calculus for the first time as well as those who need a refresher. This book makes you realize that Calculus isn't that tough after all. → *to the book*

YOUR INPUT:

$f(x) =$

$$\frac{\cos(3\pi x)}{x}$$

Simplify | Roots/zeros

FIRST DERIVATIVE:

$\frac{\mathrm{d}}{\mathrm{d}x}[f(x)] = f'(x) =$

$$-\frac{3\pi\sin(3\pi x)}{x} - \frac{\cos(3\pi x)}{x^2}$$

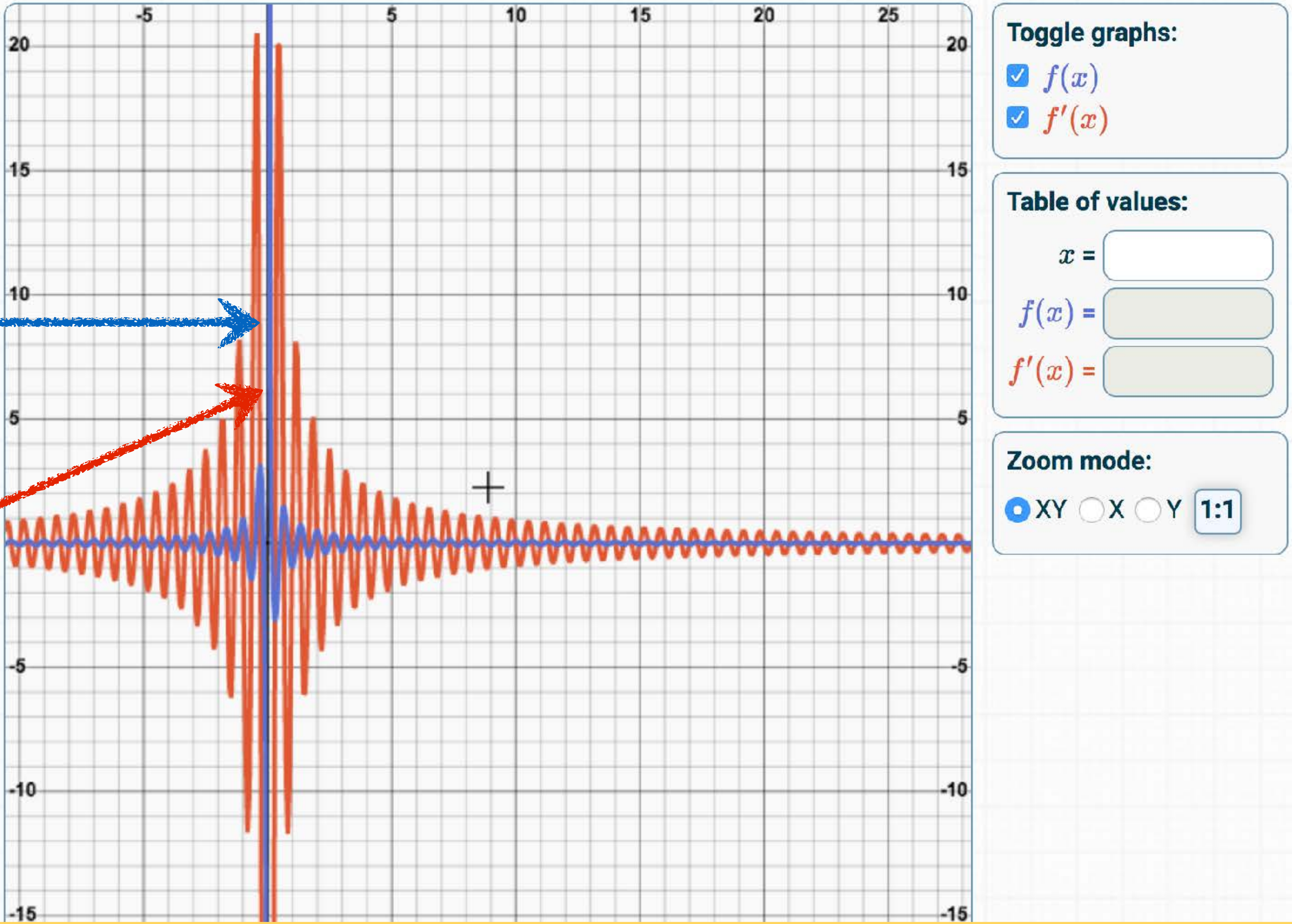
Simplify/rewrite:

$$-\frac{3\pi x\sin(3\pi x) + \cos(3\pi x)}{x^2}$$

Simplify | Show steps | Roots/zeros

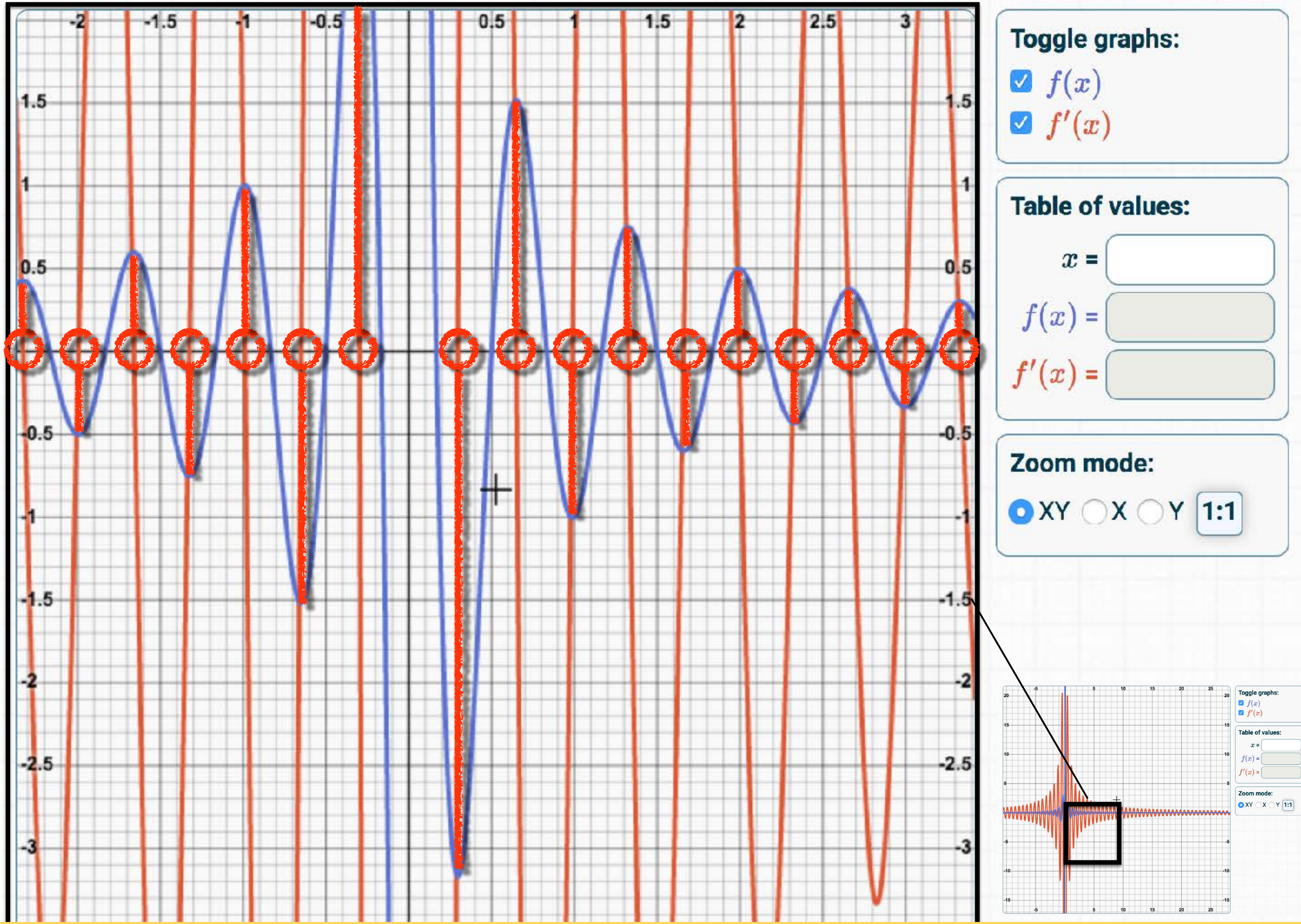
$$\frac{\cos(3\pi x)}{x}$$

$$-\frac{3\pi x \sin(3\pi x) + \cos(3\pi x)}{x^2}$$

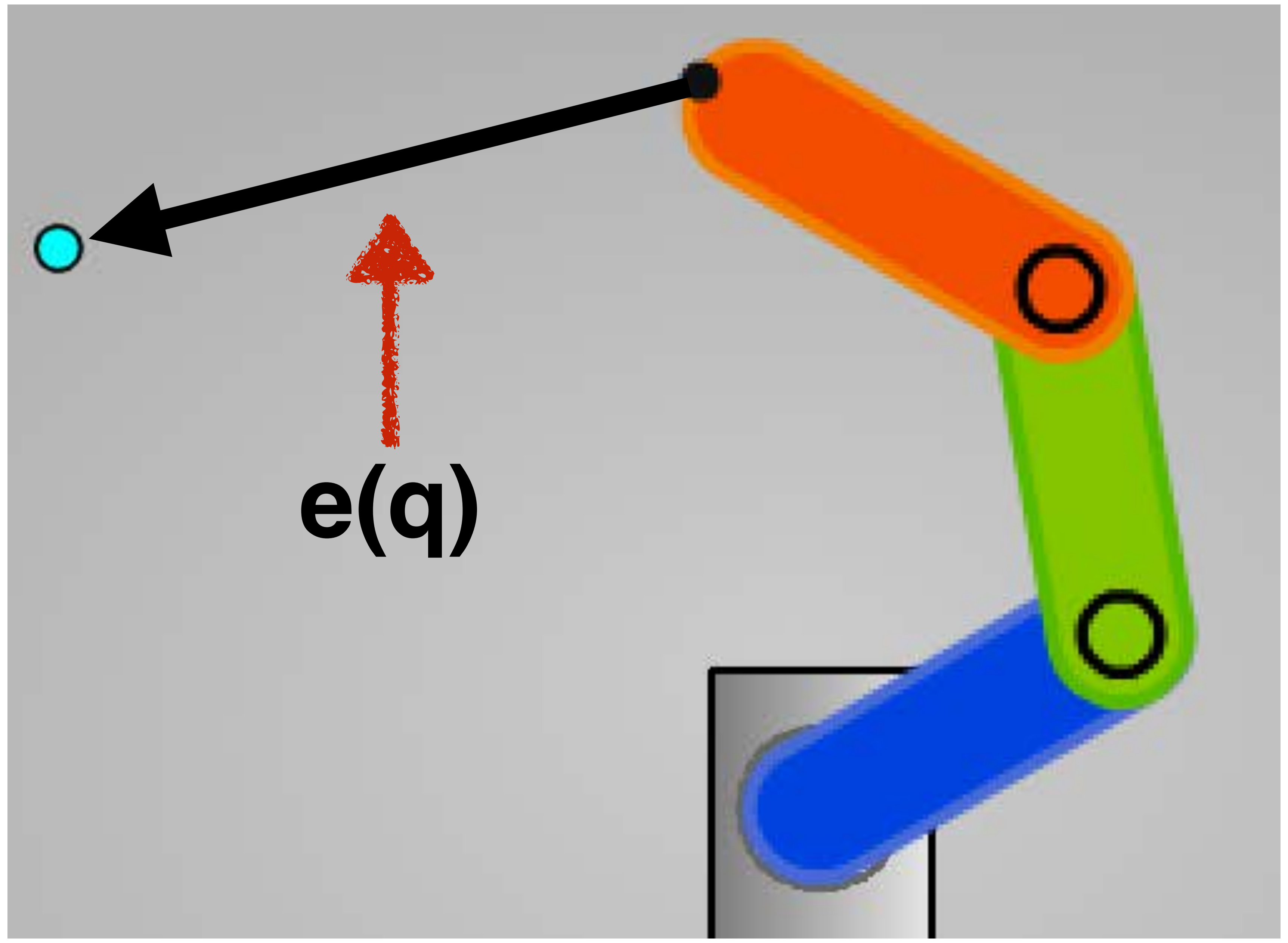


Every zero crossing of *f'(x)* is an optimum

Every zero crossing of $f'(x)$ is an optimum



Inverse kinematics as error minimization



Define error function $e(\boldsymbol{q})$ as difference between current and desired endeffector poses

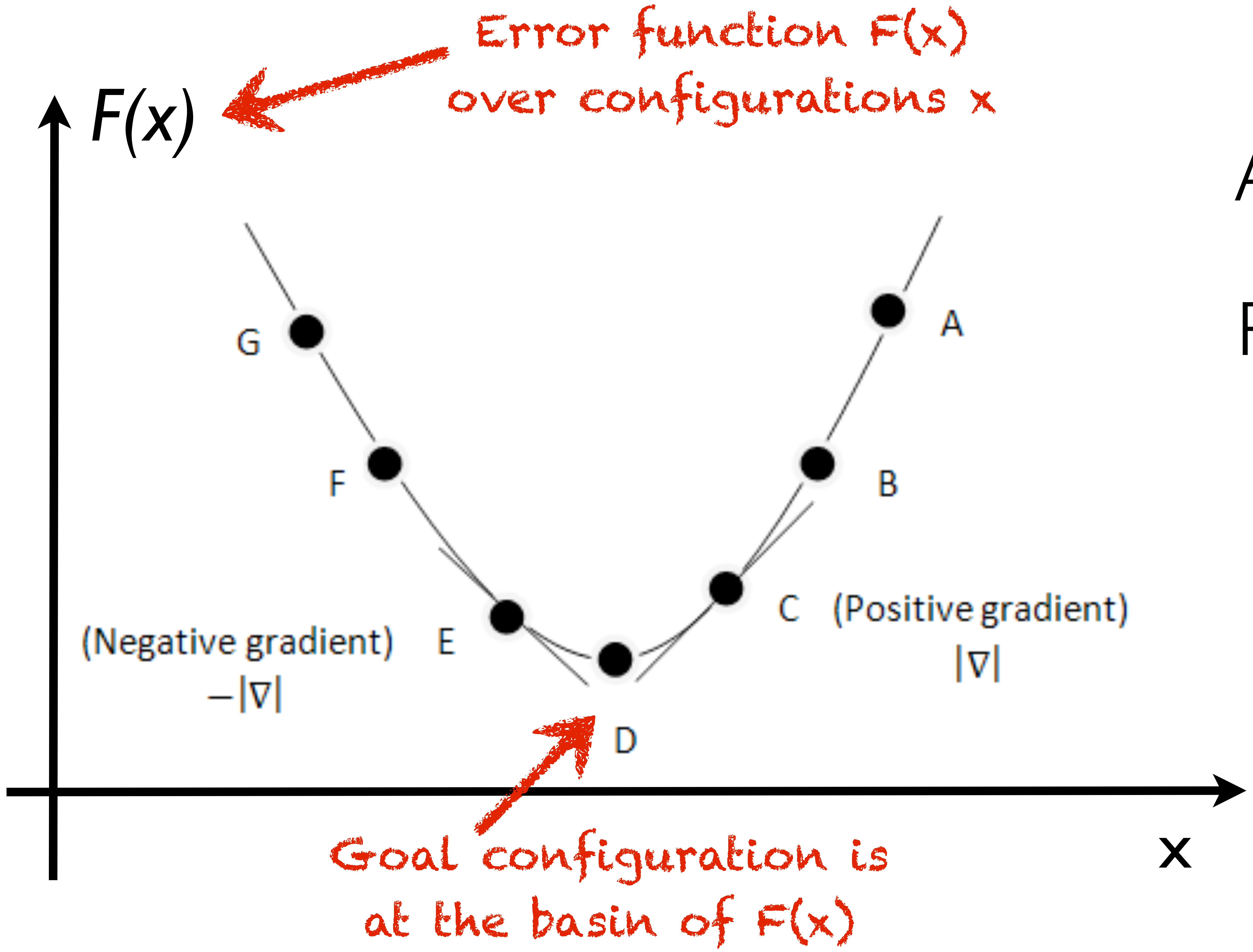
Error function parameterized by robot configuration $\boldsymbol{q}$

Find global minimum of $e(\boldsymbol{q})$,
or, $\mathrm{argmin}_{\boldsymbol{q}}\ e(\boldsymbol{q})$

But, do we know $e(\boldsymbol{q})$ in closed form?

Gradient descent

From Wikipedia, the free encyclopedia



Assign initial solution guess $\boldsymbol{x_0}$

Repeat $\boldsymbol{x_{i+1}} = \boldsymbol{x_i} - \gamma_i \, \nabla F(\boldsymbol{x_i})$

until $||\boldsymbol{x_i} - \boldsymbol{x_{i-1}}||$ is “small”

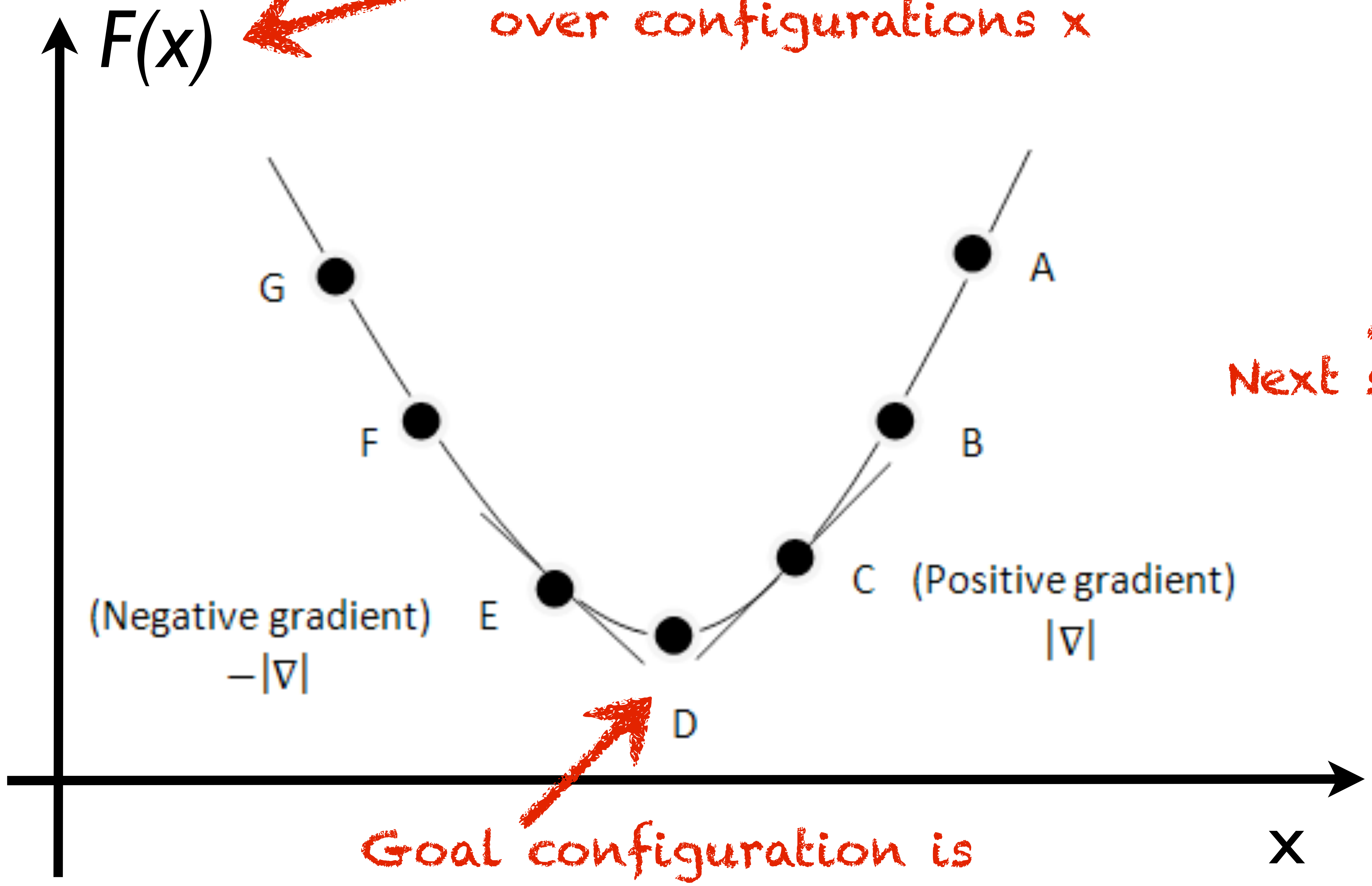
Gradient descent

From Wikipedia, the free encyclopedia

Gradient descent

From Wikipedia, the free encyclopedia

Error function F(x)
over configurations x



Goal configuration is
at the basin of F(x)

Current solution

"Learning rate"

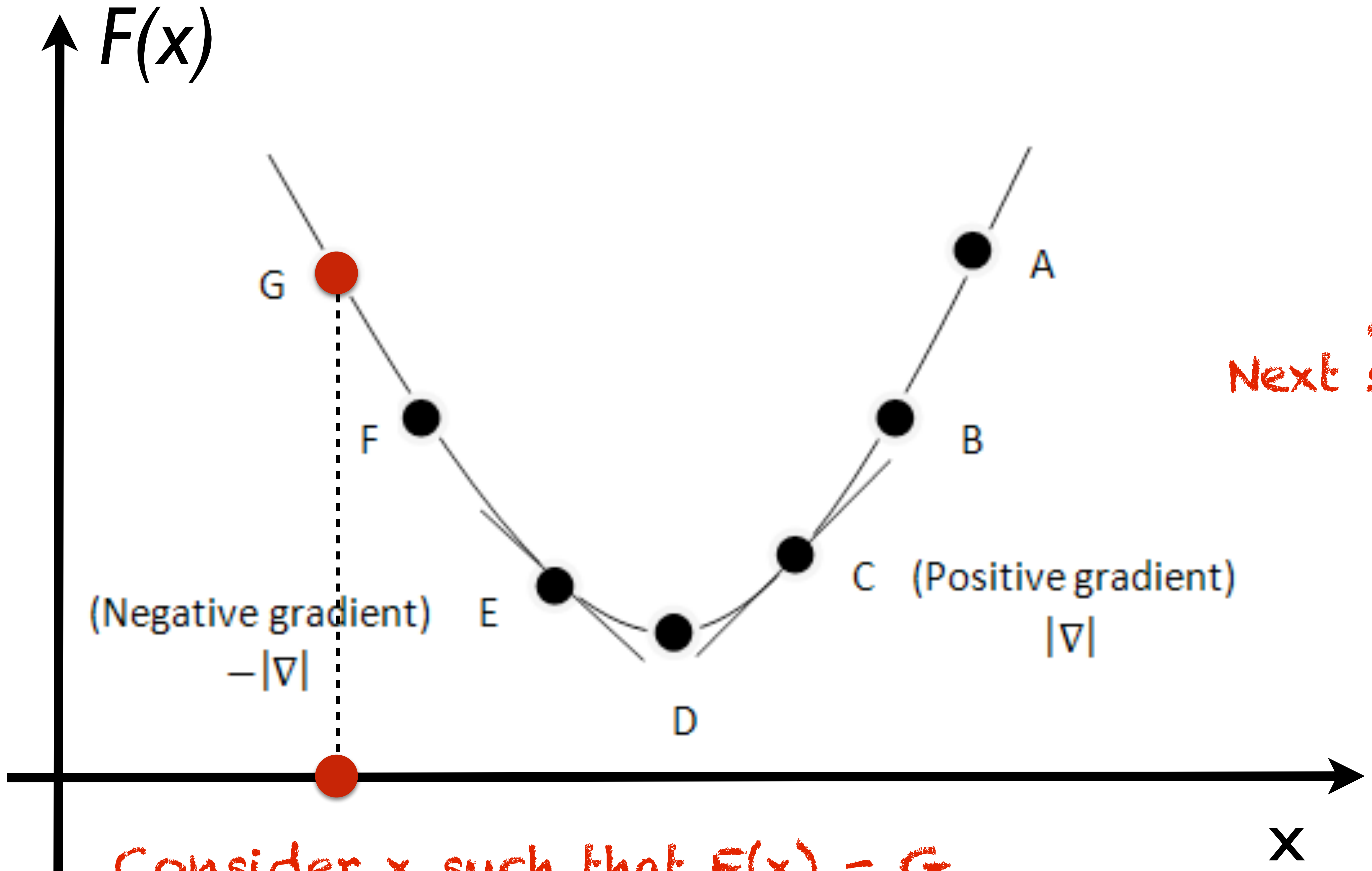
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \nabla F(\mathbf{x}_i)$$

Next solution

Derivative assumed to be direction
of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



Consider x such that F(x) = G

Current solution

"Learning rate"

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla F(\mathbf{x}_i)$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

Gradient descent

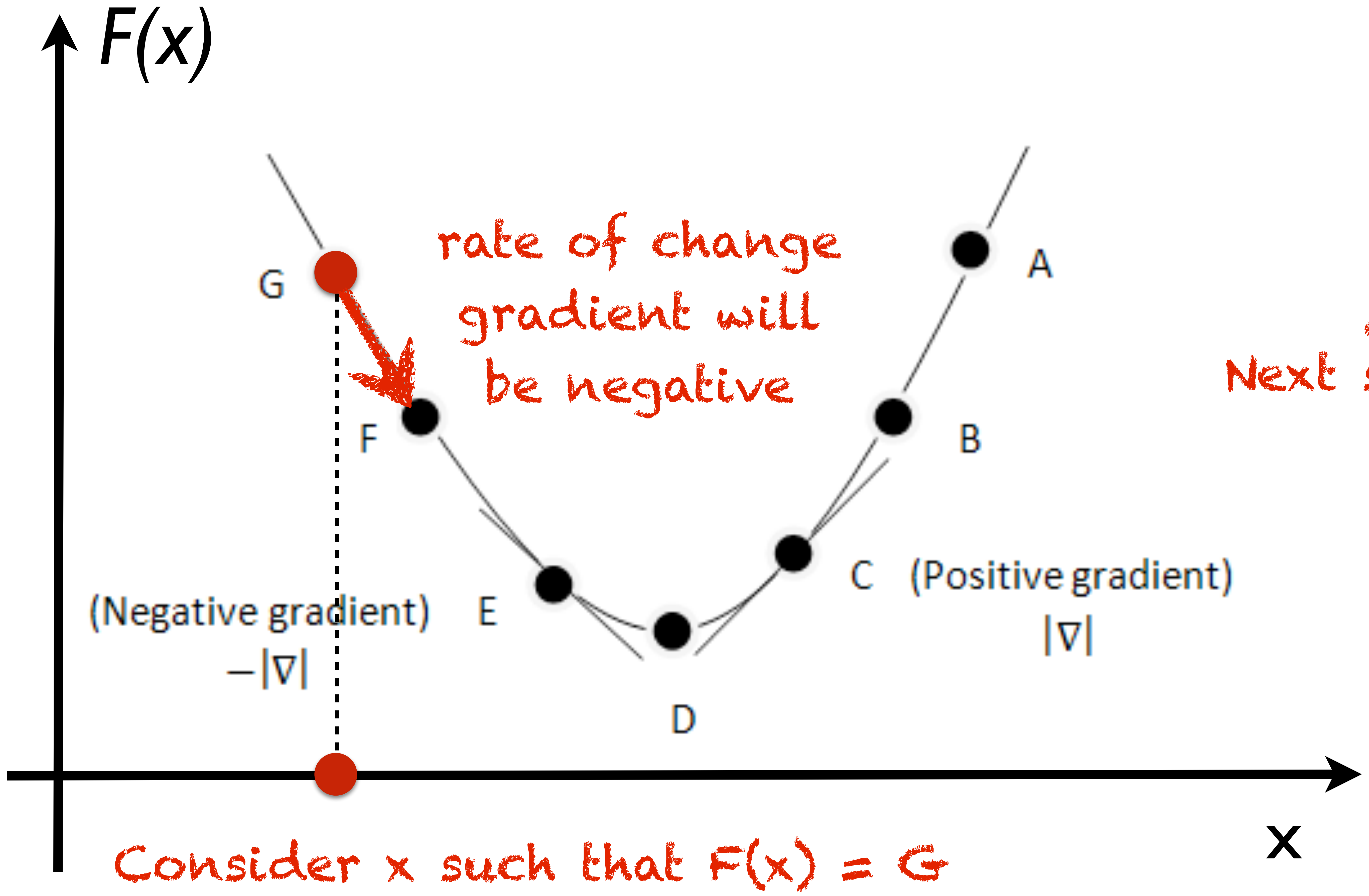
From Wikipedia, the free encyclopedia

Current solution "Learning rate"

$$x_{i+1} = x_i - \gamma_i \nabla F(x_i)$$

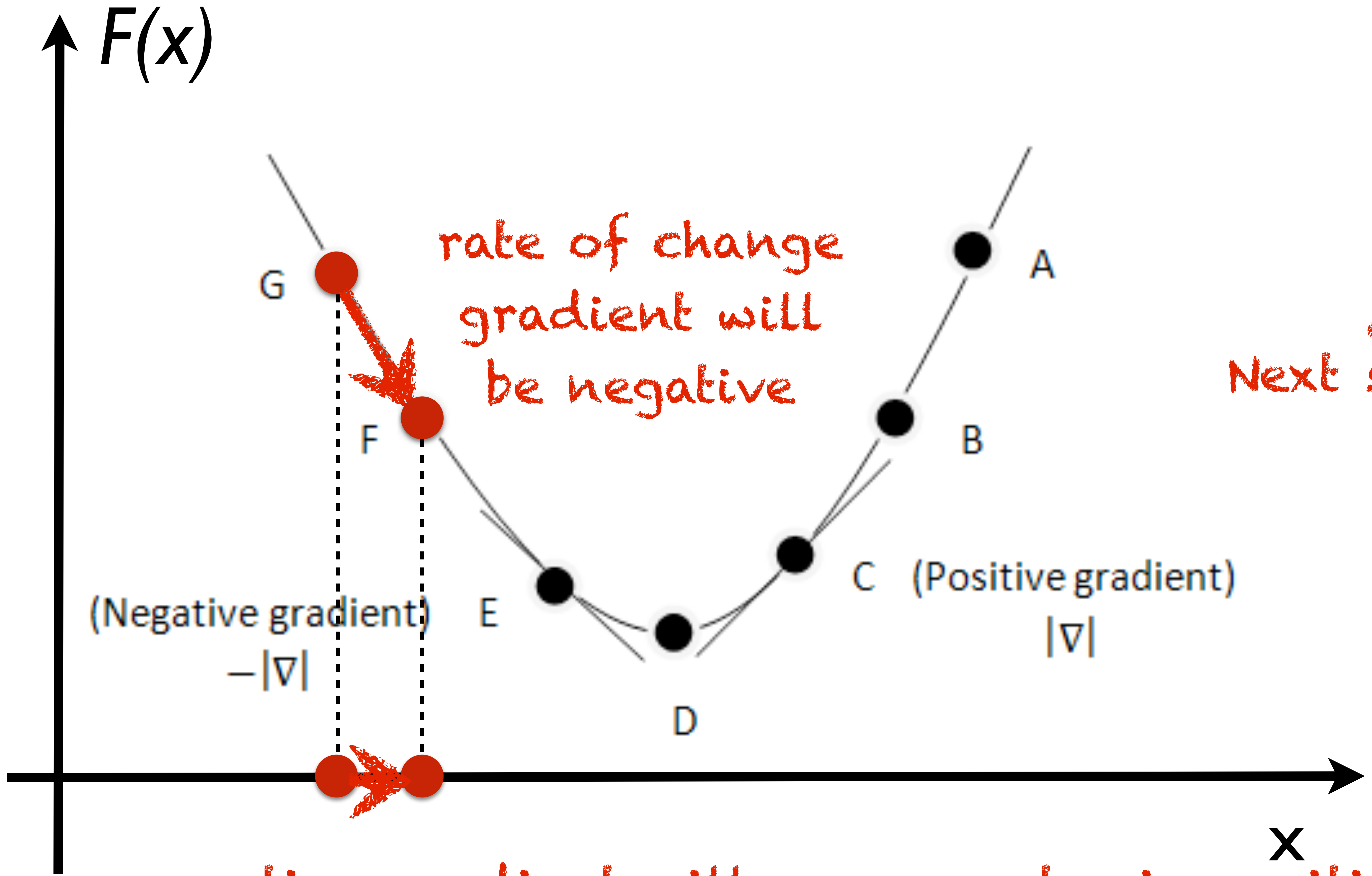
Next solution

Derivative assumed to be direction of steepest ascent away from goal



Gradient descent

From Wikipedia, the free encyclopedia



negative gradient will move next x in positive direction

Current solution

"Learning rate"

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \underline{\nabla F(\mathbf{x}_i)}$$

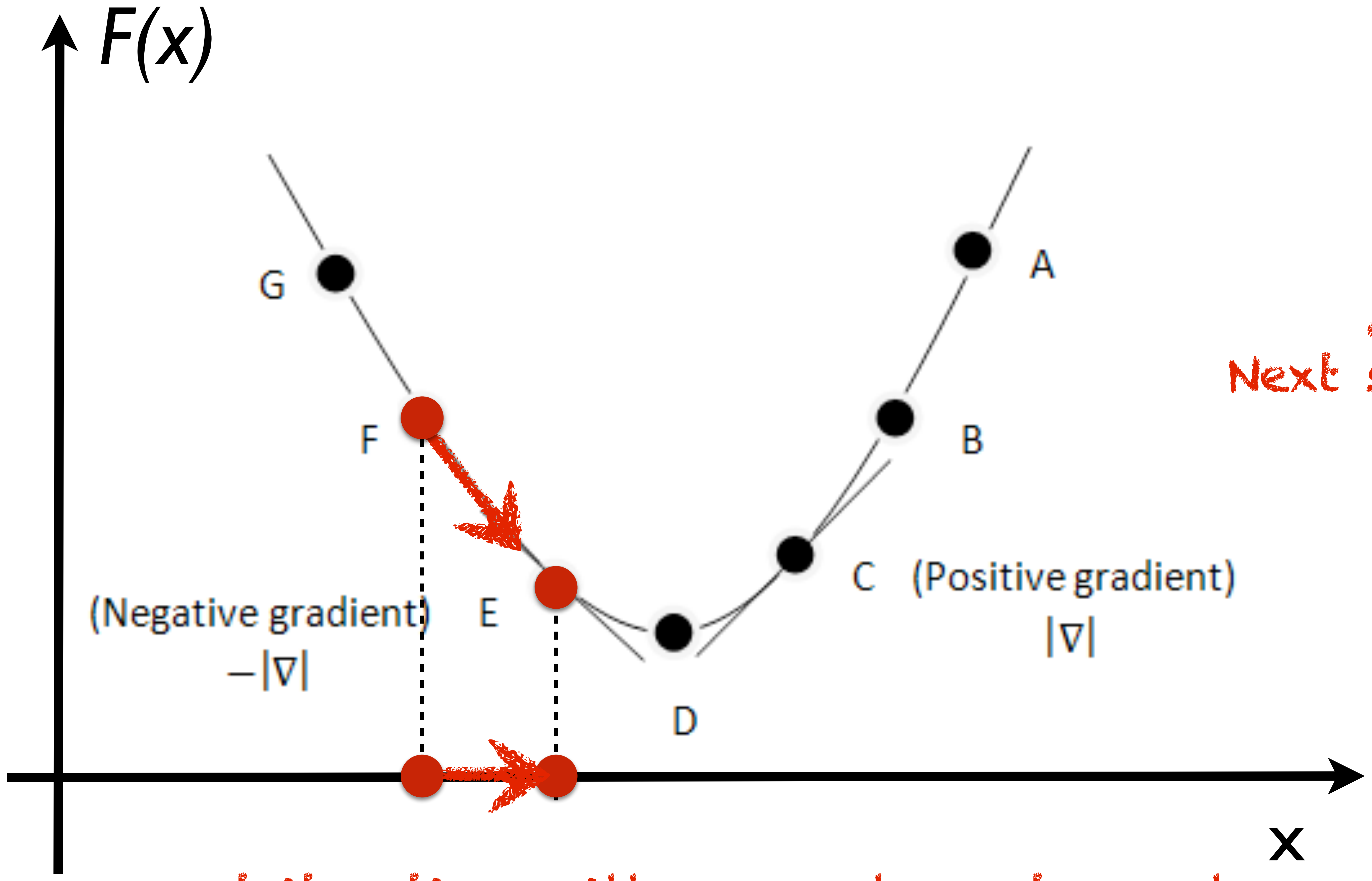
Next solution

Derivative assumed to be direction of steepest ascent away from goal

Slide borrowed from Michigan Robotics autorob.org

Gradient descent

From Wikipedia, the free encyclopedia



next iteration will move closer to goal

Current solution

"Learning rate"

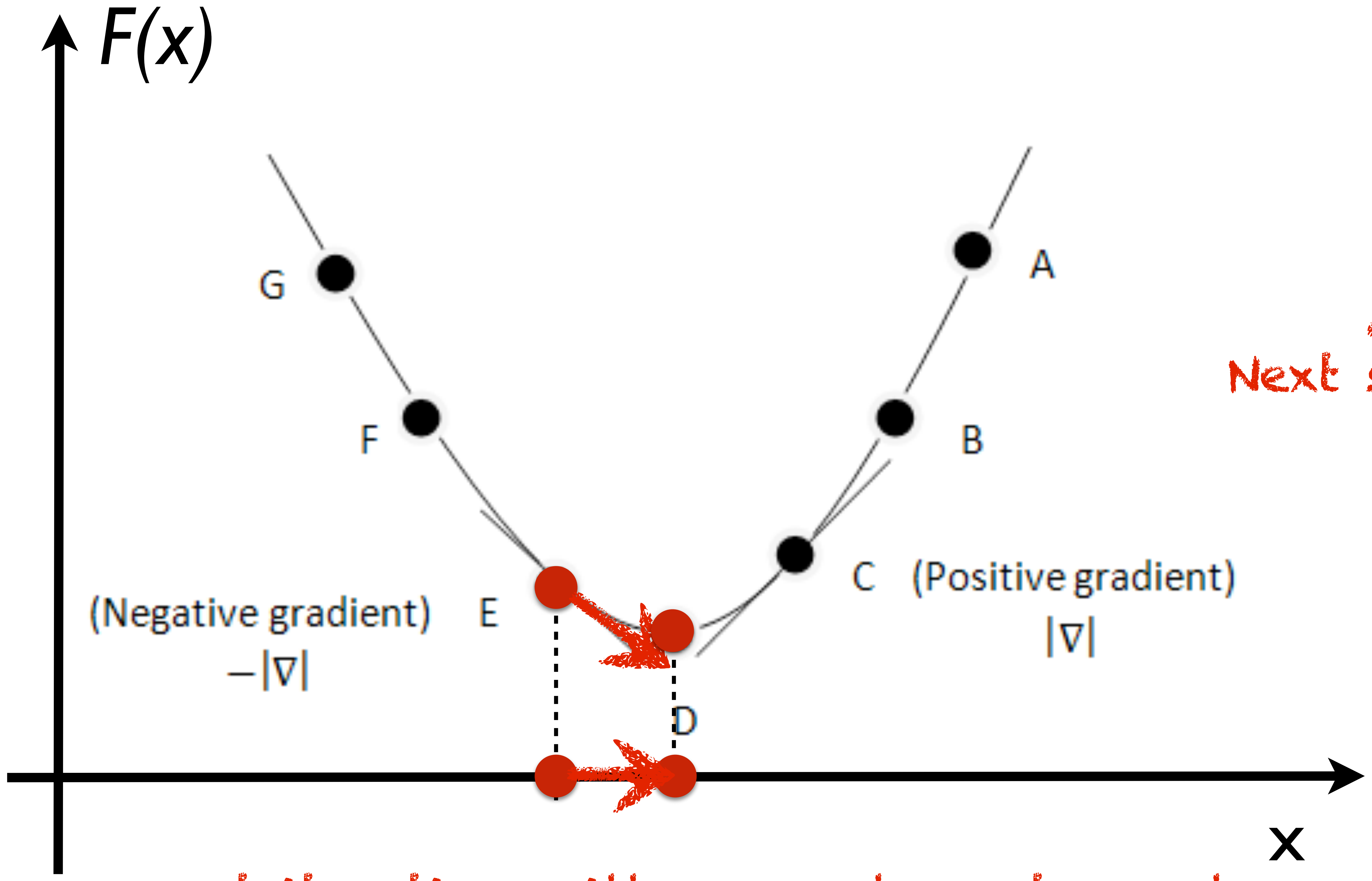
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla F(\mathbf{x}_i)$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



next iteration will move closer to goal

Current solution

"Learning rate"

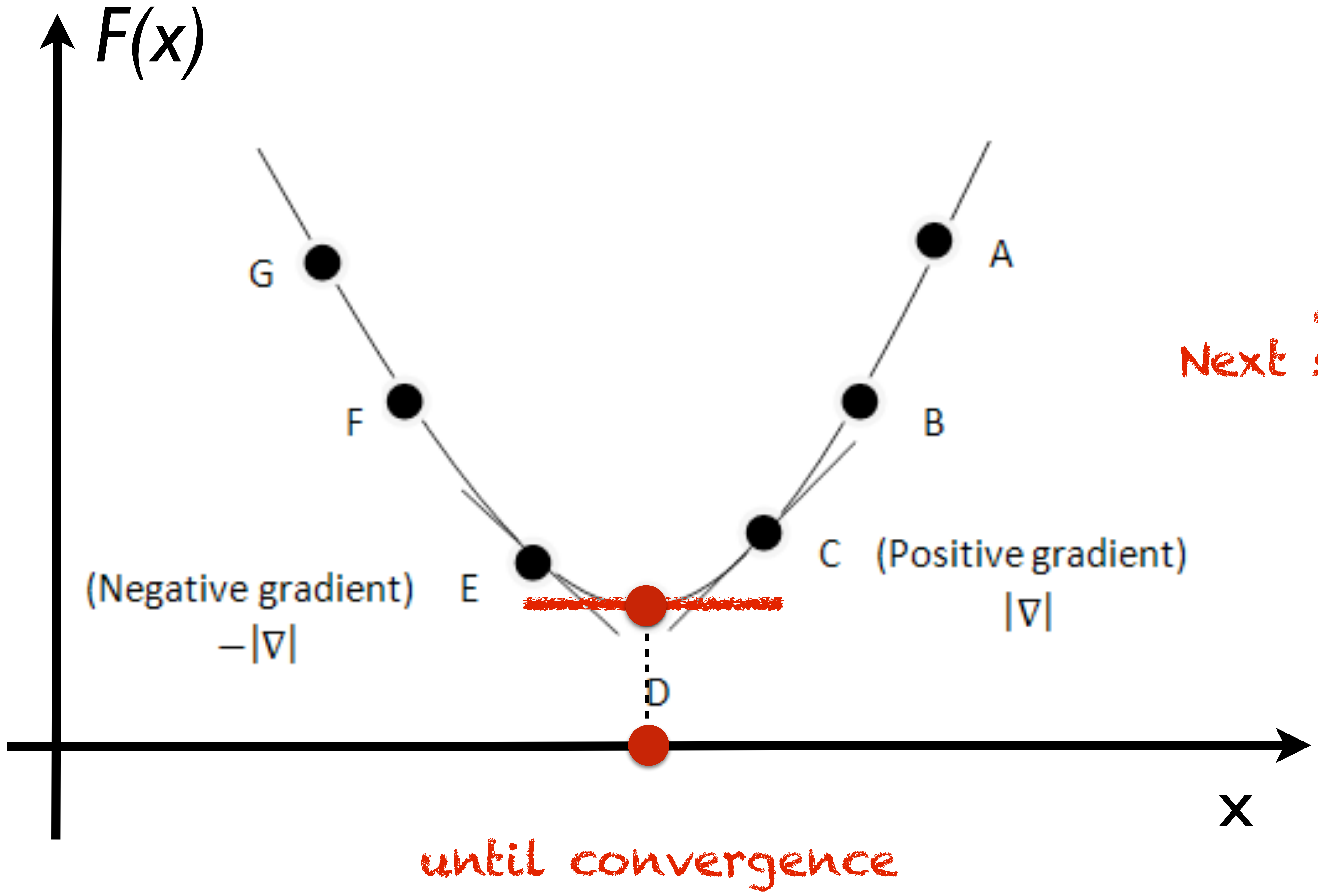
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \underline{\nabla F(\mathbf{x}_i)}$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



Current solution

"Learning rate"

$$x_{i+1} = x_i - \gamma_i \nabla F(x_i)$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

What is the derivative of robot configuration?

rate of change of
the endeffector
with respect to

What is the ~~derivative of~~ robot configuration?

What is the derivative of robot configuration?

Geometric Jacobian

3D N-link arm

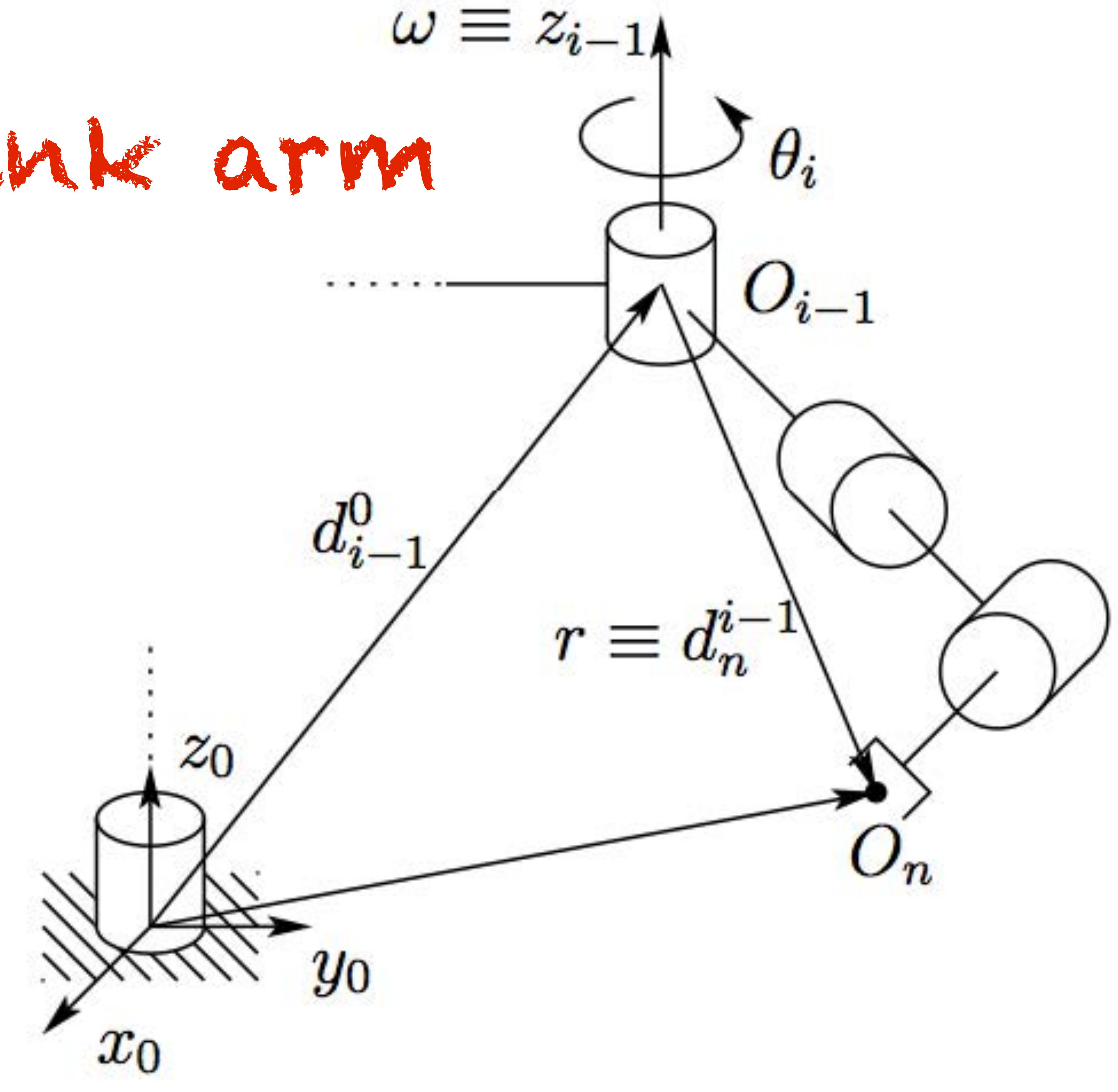


Figure 5.1: Motion of the end-effector due to link i.

Each column transforms velocity at the endeffector to velocity at a DOF

The Geometric Jacobian

A 6xN matrix

$$\boldsymbol{J} = [J_1 J_2 \cdots J_n]$$

assuming forward kinematics:

$$\boldsymbol{x} = f(\boldsymbol{q})$$

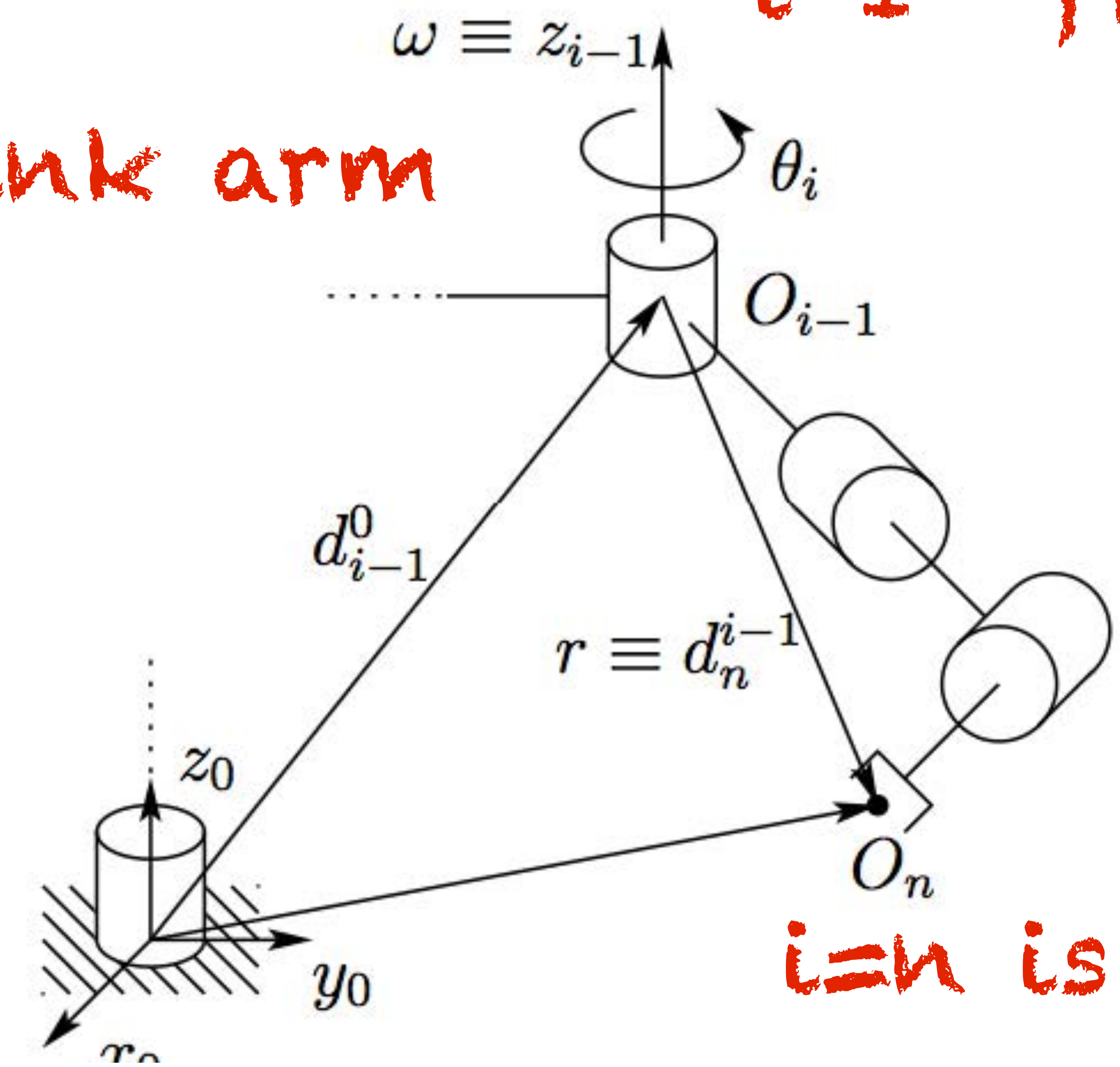
represents partial derivative:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}} = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}} = J(\boldsymbol{q})$$

i-1th frame maps to ith column

The Jacobian

3D N-link arm



i=n is endeffector frame

i=0 is base frame

... effector due to link i.

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

$$\boxed{T^0_n(\boldsymbol{q}) = \begin{bmatrix} R^0_n(\boldsymbol{q}) & o^0_n(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix}}$$

Note: figure taken from Spong et al. textbook, which assumes D-H parameters and offset column index

The Jacobian

3D N-link arm

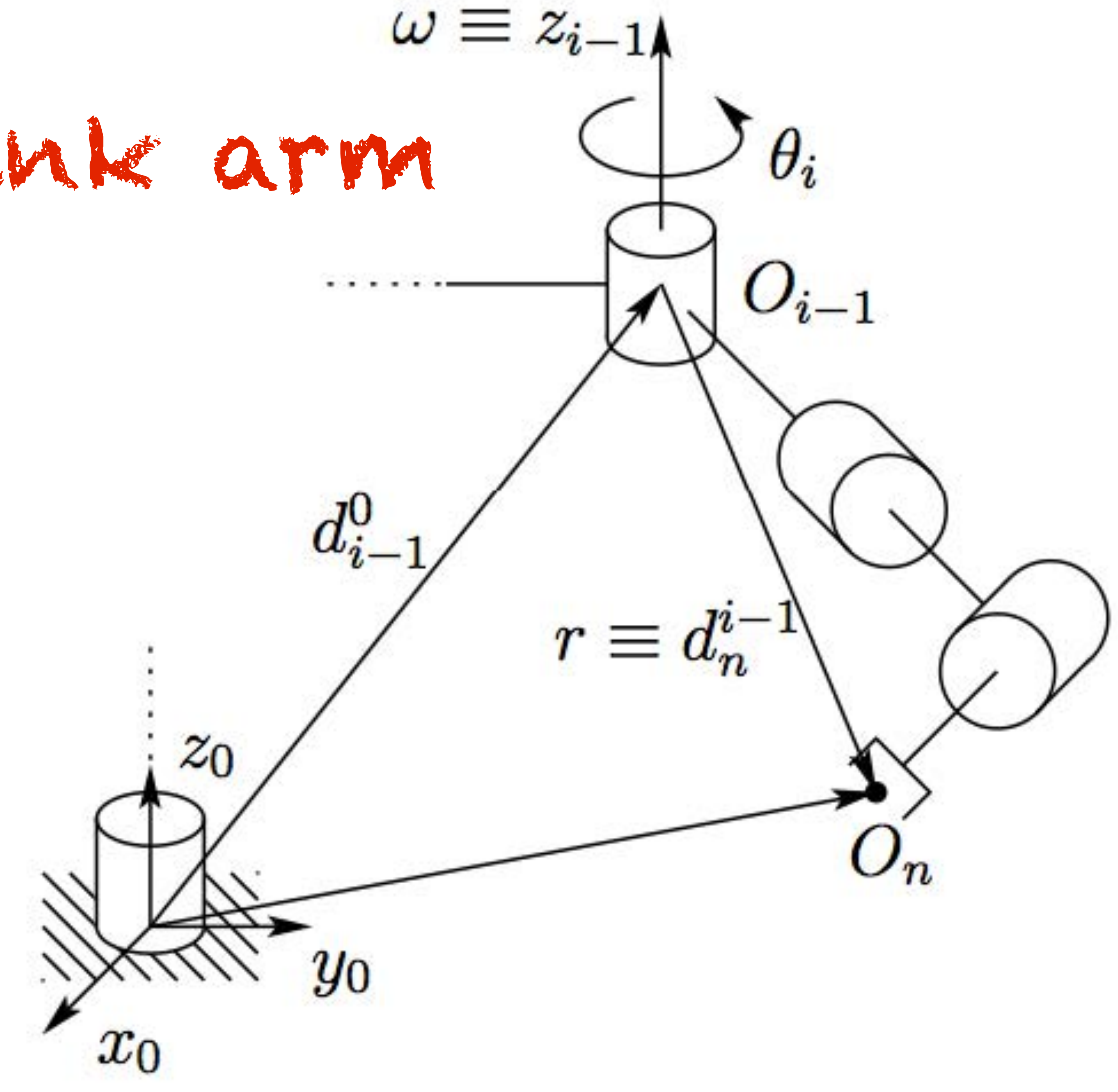


Figure 5.1: Motion of the end-effector due to link i.

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

consisting of two 3xN matrices

$$J = \left[\begin{array}{c} J_v \\ \hline J_\omega \end{array}\right]$$

with overall form

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

$\frac{\partial F_1}{\partial x_1}$ Change in an endeffector variable wrt. change in a joint variable

Slide borrowed from Michigan Robotics autorob.org

3D N-link arm

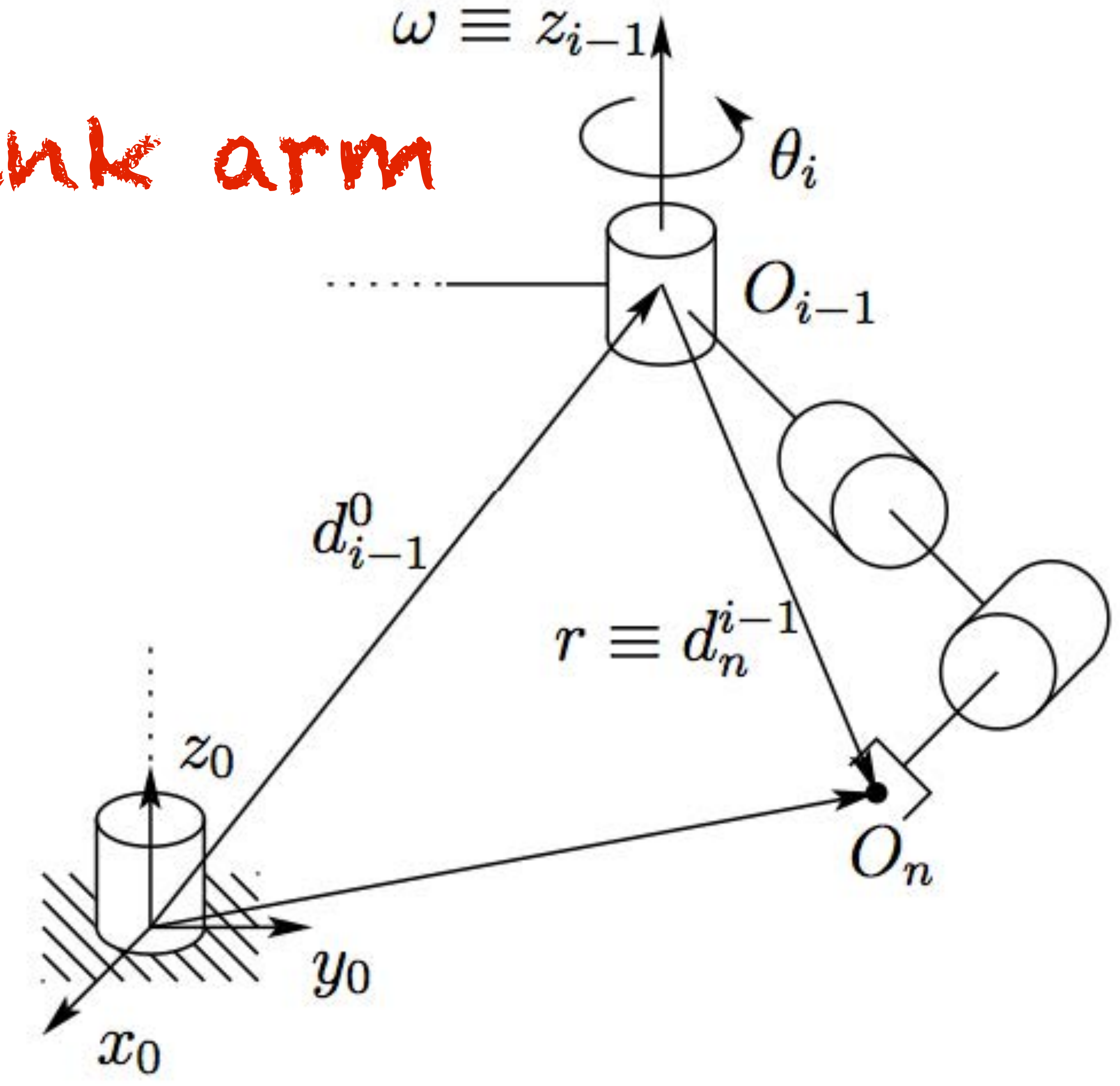


Figure 5.1: Motion of the end-effector due to link i.

The Jacobian

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

consisting of two 3xN matrices

$$J = \begin{bmatrix} J_v \\ \hline J_\omega \end{bmatrix}$$

linear

angular

linear velocity of endeffector → $v_n^0 = J_v \dot{q}$

angular velocity of endeffector → $\omega_n^0 = J_\omega \dot{q}$

vector of of joint angle velocities

The Jacobian

3D N-link arm

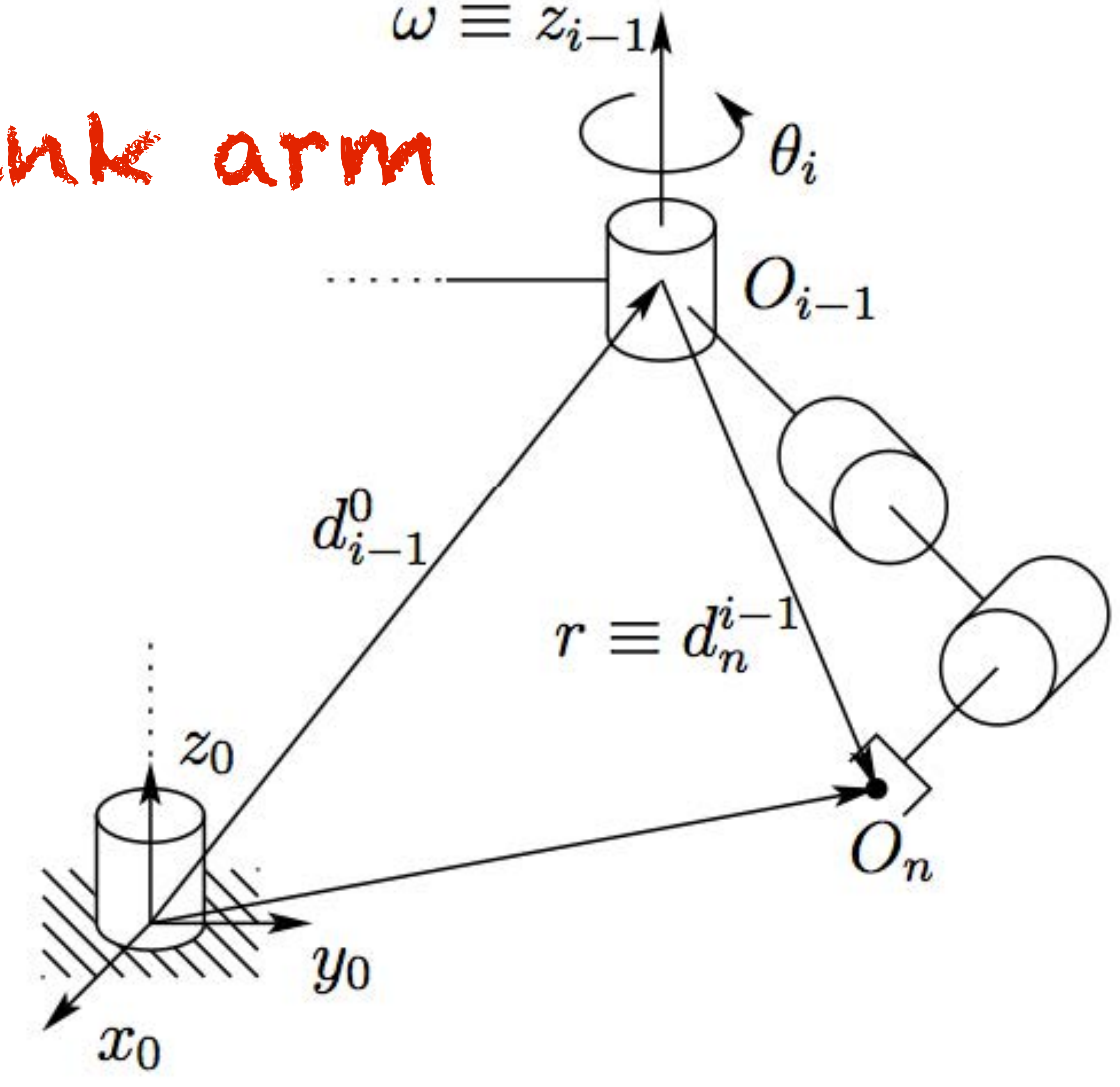


Figure 5.1: Motion of the

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

consisting of two 3xN matrices

$$J = \left[\begin{array}{c} J_v \\ \hline J_\omega \end{array}\right]$$

linear

angular

J_i is a single column of the Jacobian matrix

z is joint axis in world coordinates (overloaded notation)

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

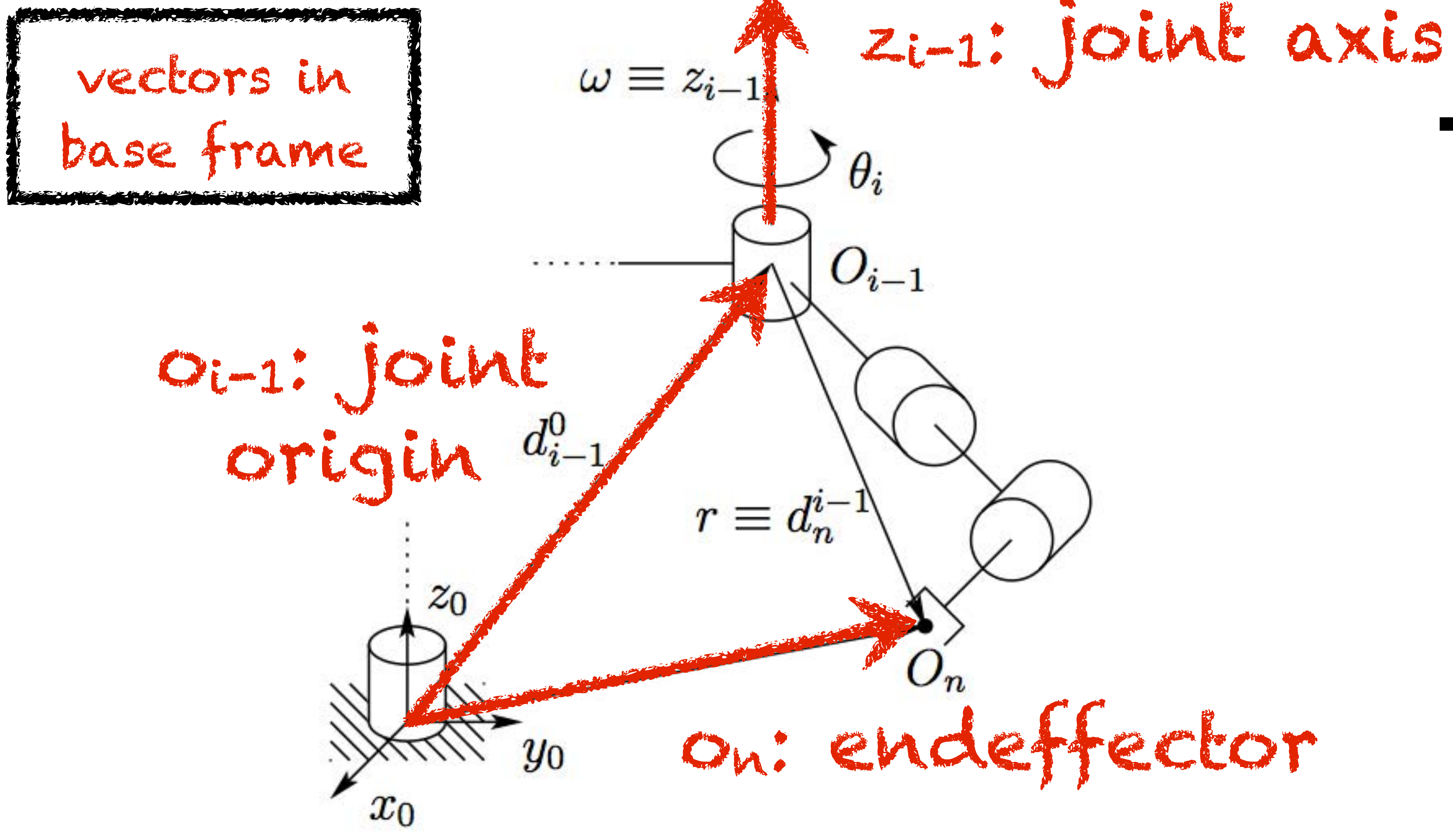


Figure 5.1: Motion of the end-effector due to link i.

The Jacobian

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

$$J = \left[\begin{array}{c} J_v \\ \hline J_\omega \end{array}\right] \begin{array}{l} \leftarrow \text{linear} \\ \leftarrow \text{angular} \end{array}$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

The Jacobian

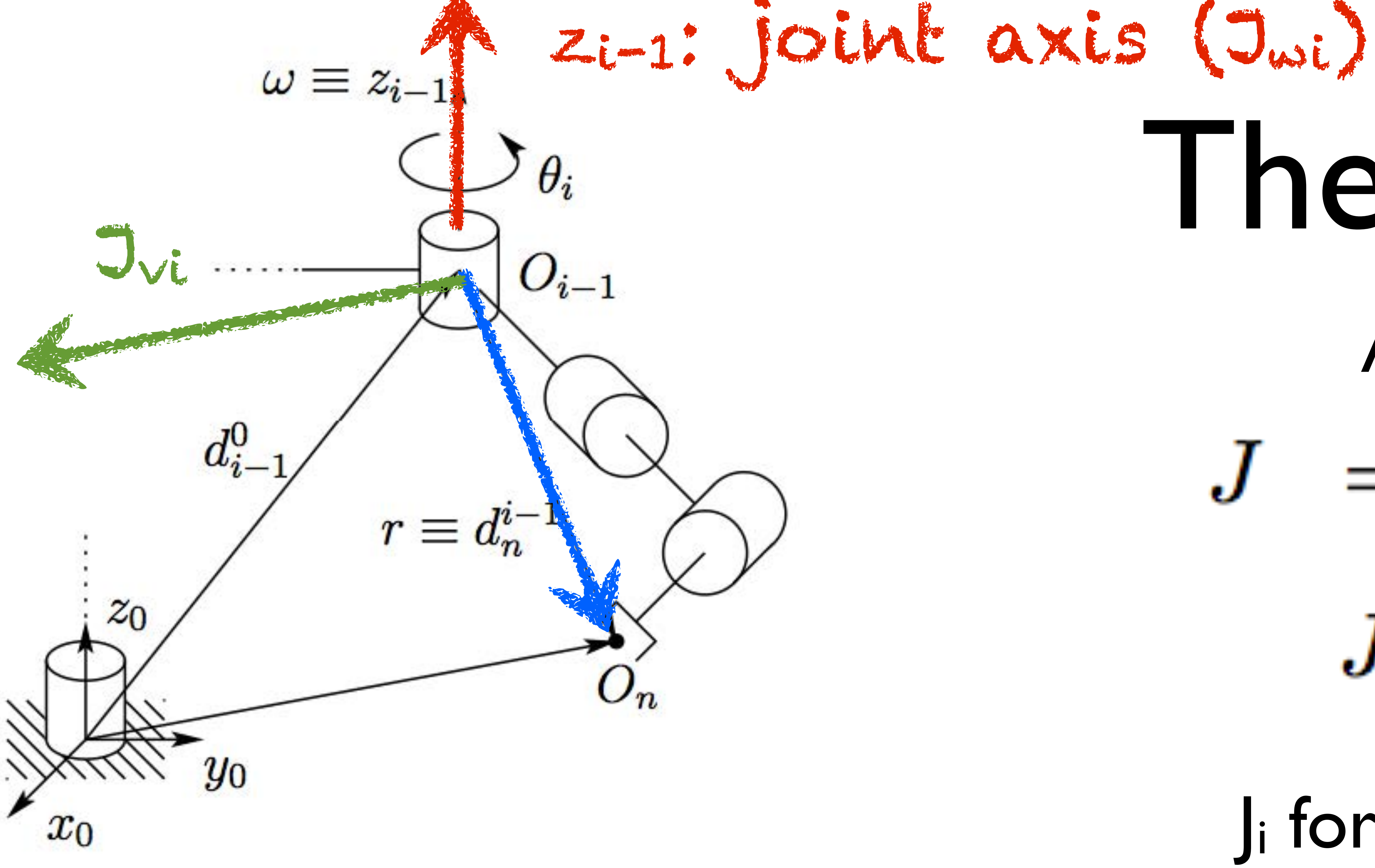


Figure 5.1: Motion of the end-effector due to link i.

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

$$J = \left[\frac{J_v}{J_\omega}\right] \begin{array}{l} \leftarrow \text{linear} \\ \leftarrow \text{angular} \end{array}$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

The Jacobian

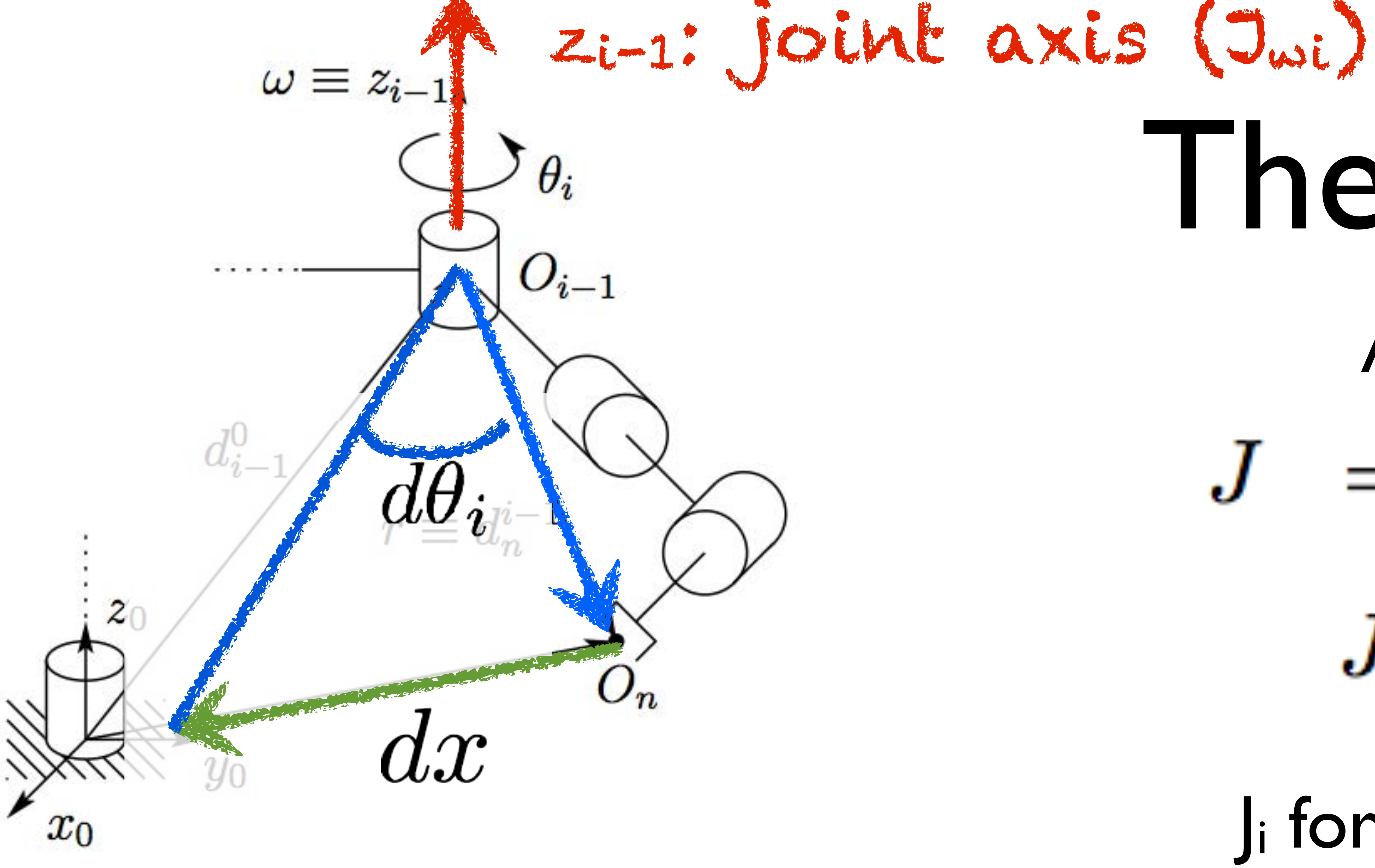


Figure 5.1: Motion of the end-effector due to link i.

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

$$J = \begin{bmatrix} J_v \\ \hline J_\omega \end{bmatrix} \begin{matrix} \leftarrow \text{linear} \\ \leftarrow \text{angular} \end{matrix}$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

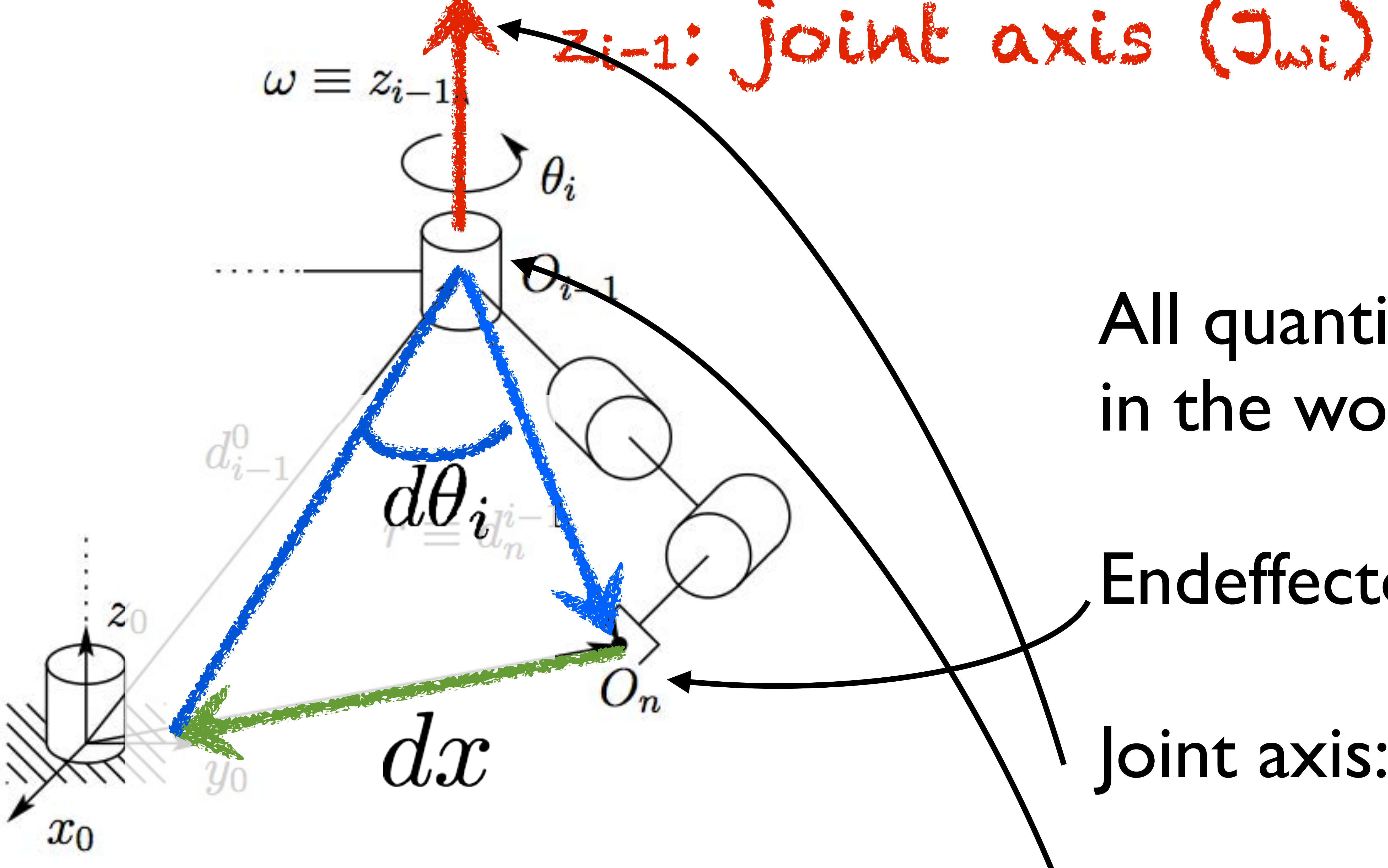


Figure 5.1: Motion of the end-effector due to link i.

Important

All quantities must be expressed in the world frame

Endeffector: $\{\boldsymbol{p}_{\text{tool}}\}^w = T_n^w \{\boldsymbol{p}_{\text{tool}}\}^n$

Joint axis: $\{\boldsymbol{k}_i\}^w = T_i^w \{\boldsymbol{k}_i\}^i$

Joint origin: $\{\boldsymbol{o}_i\}^w = T_i^w \{\boldsymbol{o}_i\}^i$

$$J_{\boldsymbol{v}i} = (\{\boldsymbol{k}_i\}^w - \{\boldsymbol{o}_i\}^w) \times (\{\boldsymbol{p}_{\text{tool}}\}^w - \{\boldsymbol{o}_i\}^w)$$
$$J_{\boldsymbol{\omega}i} = \{\boldsymbol{k}_i\}^w - \{\boldsymbol{o}_i\}^w$$

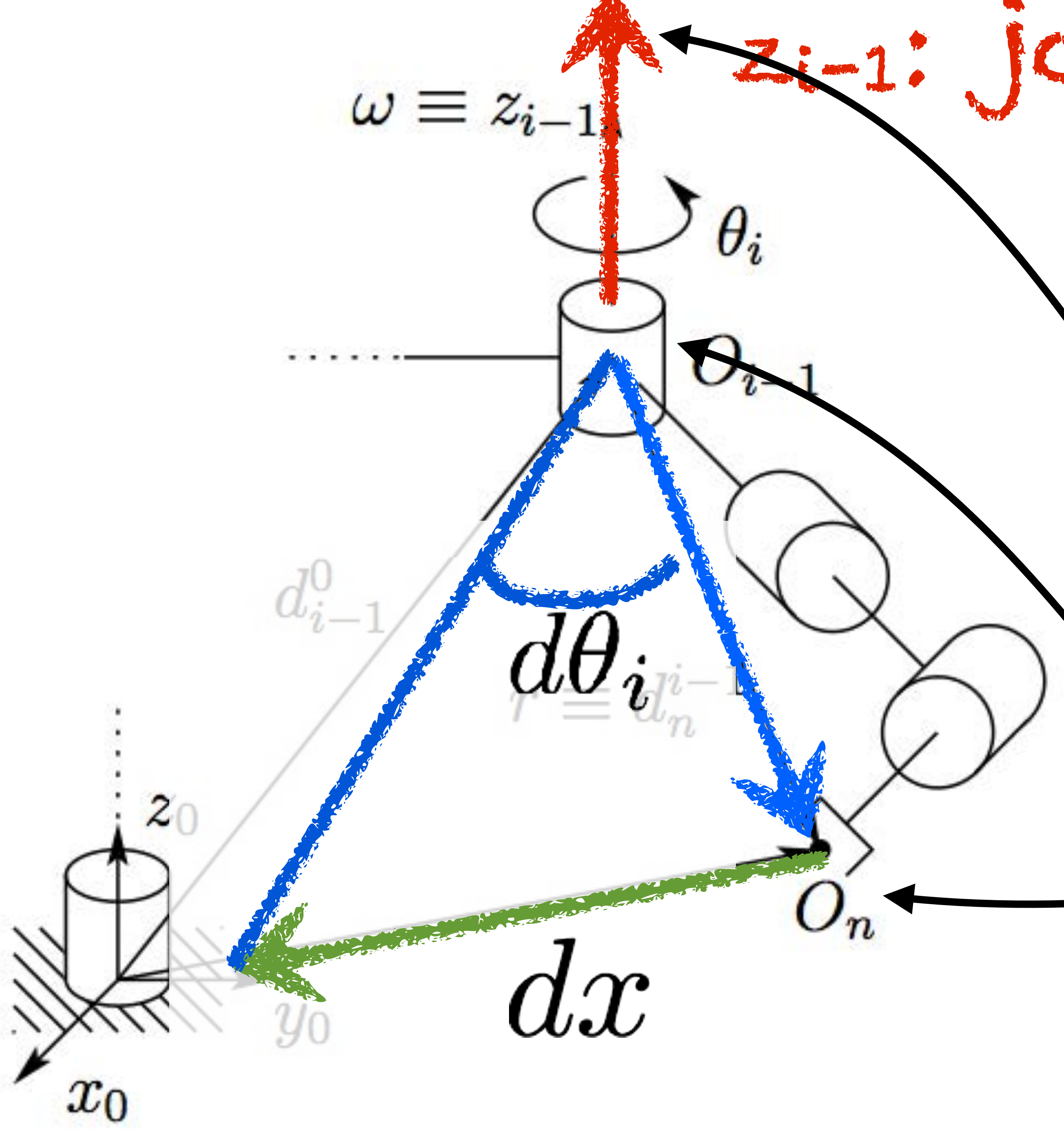


Figure 5.1: Motion of the end-effector due to link i.

Impo

All quantities must be in the world frame

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

Endeffector: $\{\boldsymbol{p}_{\text{tool}}\}^w = T_n^w \{\boldsymbol{p}_{\text{tool}}\}^n$

Joint axis: $\{\boldsymbol{k}_i\}^w = T_i^w \{\boldsymbol{k}_i\}^i$

Joint origin: $\{\boldsymbol{o}_i\}^w = T_i^w \{\boldsymbol{o}_i\}^i$

$$\boxed{\begin{gathered} J_{\boldsymbol{v}i} = (\{\boldsymbol{k}_i\}^w - \{\boldsymbol{o}_i\}^w) \times (\{\boldsymbol{p}_{\text{tool}}\}^w - \{\boldsymbol{o}_i\}^w) \\ J_{\boldsymbol{\omega}i} = \{\boldsymbol{k}_i\}^w - \{\boldsymbol{o}_i\}^w \end{gathered}}$$

How did we get the Geometric Jacobian?

Velocity of Point Rotating on N-link Arm

$$T_n^0(\boldsymbol{q}) = \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix}$$

Angular Velocity

$$R_n^0 = R_1^0 R_2^1 \cdots R_n^{n-1}$$

assuming velocities expressed in the same frame

$$\omega_n^0 = \omega_1^0 + R_1^0\omega_2^1 + R_2^0\omega_3^2 + R_3^0\omega_4^3 + \cdots + R_{n-1}^0\omega_n^{n-1}$$

$$z_{i-1}^0 = R_{i-1}^0 \boldsymbol{k}$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

$$J_\omega = [z_0^0 \quad z_1^0 \quad \ldots \quad z_{n-1}^0]$$

Velocity of Point Rotating on N-link Arm

$$
\begin{aligned}
T_n^0(\boldsymbol{q}) &= \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \\
&= T_n^0 \\
&= T_{i-1}^0 T_i^{i-1} T_n^i \\
&= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_n^0 & R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix}
\end{aligned}
$$

consider effect of all frames (0..n) on endeffector

Velocity of Point Rotating on N-link Arm

$$
\begin{aligned}
T_n^0(\boldsymbol{q}) &= \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \\
&= T_n^0 \\
&= T_{i-1}^0 T_i^{i-1} T_n^i \\
&= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_n^0 & \boxed{R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0} \\ \mathbf{0} & 1 \end{bmatrix}
\end{aligned}
$$

<u>Linear Velocity for Rotational Joint</u>

$$
o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0
$$

position of endeffector frame

Velocity of Point Rotating on N-link Arm

$$
\begin{aligned}
T_n^0(\boldsymbol{q}) &= \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \\
&= T_n^0 \\
&= T_{i-1}^0 T_i^{i-1} T_n^i \\
&= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_n^0 & \boxed{R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0} \\ \mathbf{0} & 1 \end{bmatrix}
\end{aligned}
$$

<u>Linear Velocity for Rotational Joint</u>

$$o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0$$

$$\frac{\partial}{\partial \theta_i} o_n^0 = \frac{\partial}{\partial \theta_i} \left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} \right]$$

take derivative wrt.
i^{th} joint angle

Velocity of Point Rotating on N-link Arm

$$\begin{aligned}
T_n^0(\boldsymbol{q}) &= \begin{bmatrix} R_n^0(\boldsymbol{q}) & o_n^0(\boldsymbol{q}) \\ \mathbf{0} & 1 \end{bmatrix} \\
&= T_n^0 \\
&= T_{i-1}^0 T_i^{i-1} T_n^i \\
&= \begin{bmatrix} R_{i-1}^0 & o_{i-1}^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_n^i & o_n^i \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_n^0 & \boxed{R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0} \\ \mathbf{0} & 1 \end{bmatrix}
\end{aligned}$$

<u>Linear Velocity for Rotational Joint</u>

$$o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} o_n^0 &= \frac{\partial}{\partial \theta_i}\left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1}\right] \\
&= \dot{\theta}_i S(z_{i-1}^0) R_i^0 o_n^i + \dot{\theta}_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1} \\
&= \dot{\theta}_i z_{i-1}^0 \times (o_n^0 - o_{i-1}^0)
\end{aligned}$$

$$\boxed{J_{\boldsymbol{v}_i} = z_{i-1} \times (o_n - o_{i-1})}$$

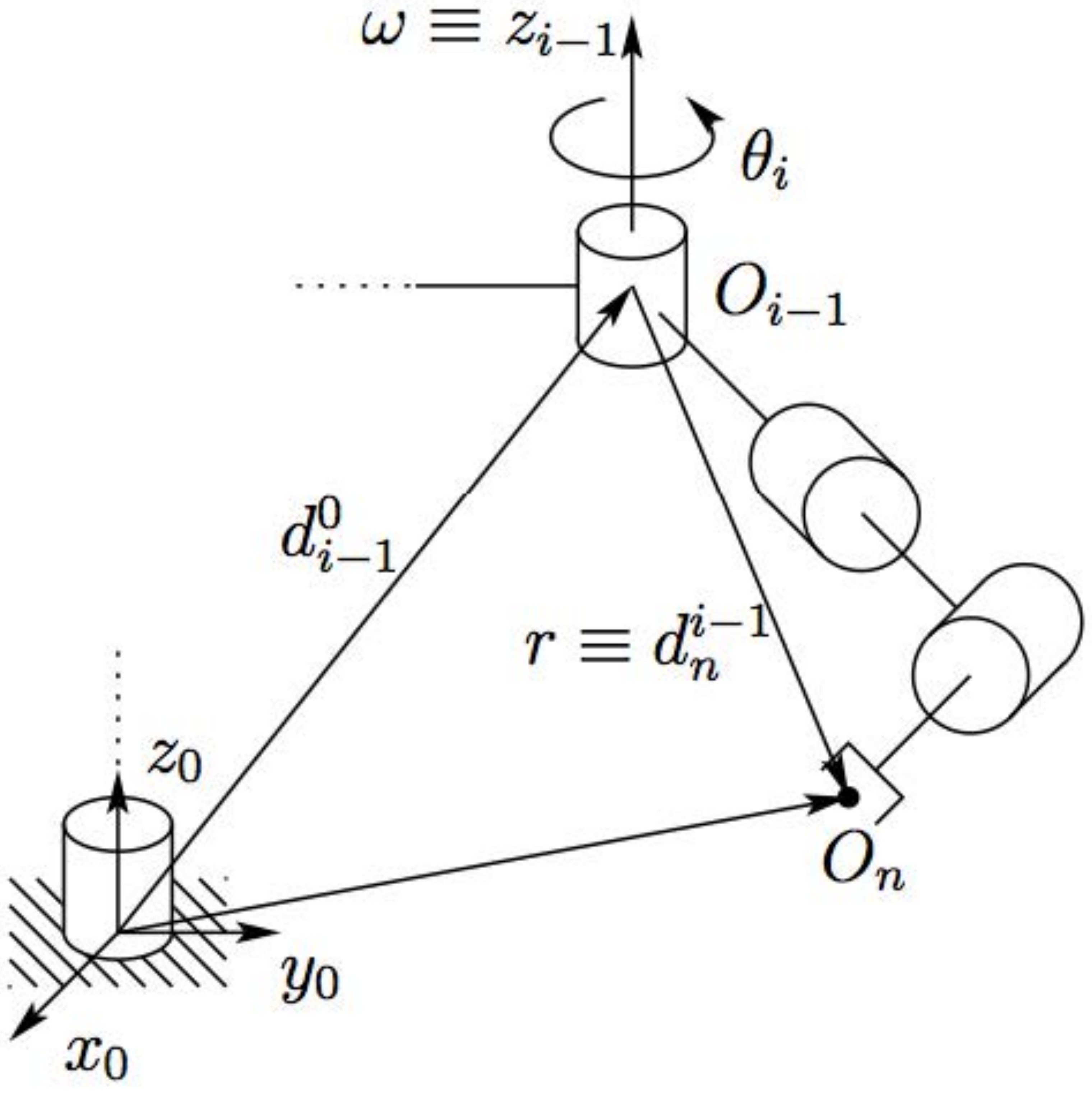
J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

IK Prodedure Restated

Geometric

The Jacobian



A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

IK Procedure restated:

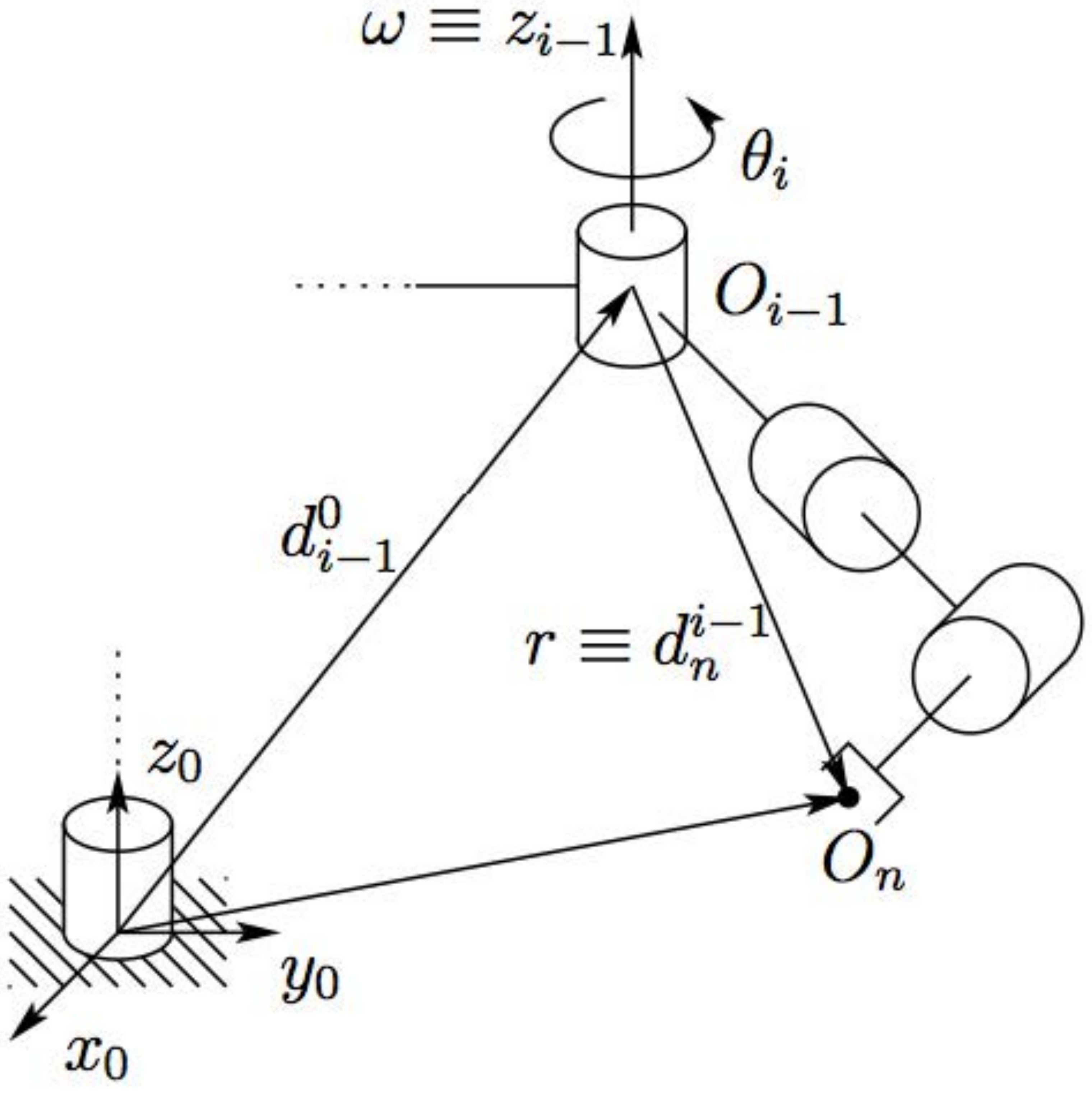
$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$

Geometric

The Jacobian



compute endpoint error

IK Procedure restated:

$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

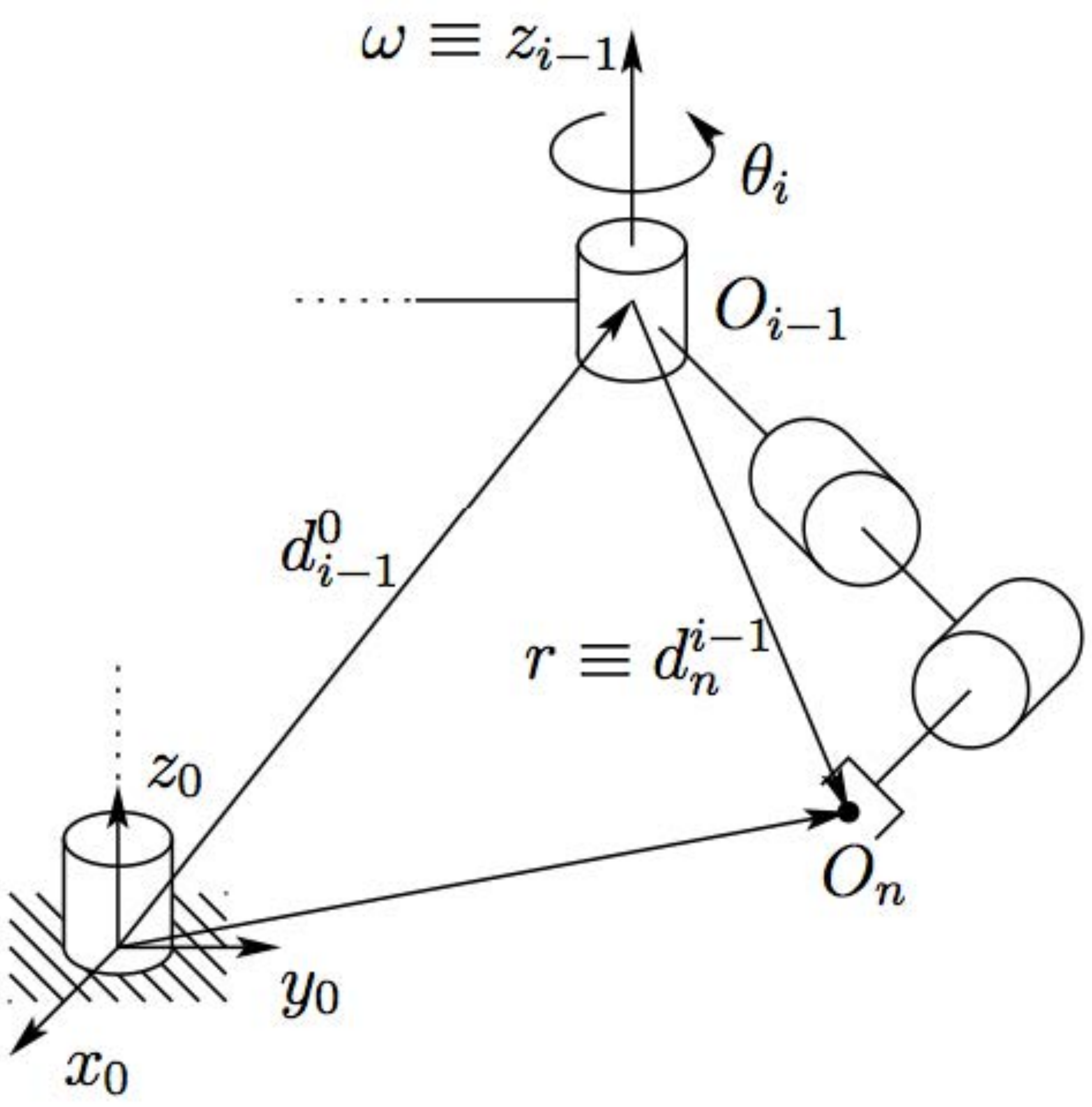
J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

The Geometric Jacobian



A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

IK Procedure restated:

compute endpoint error

$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

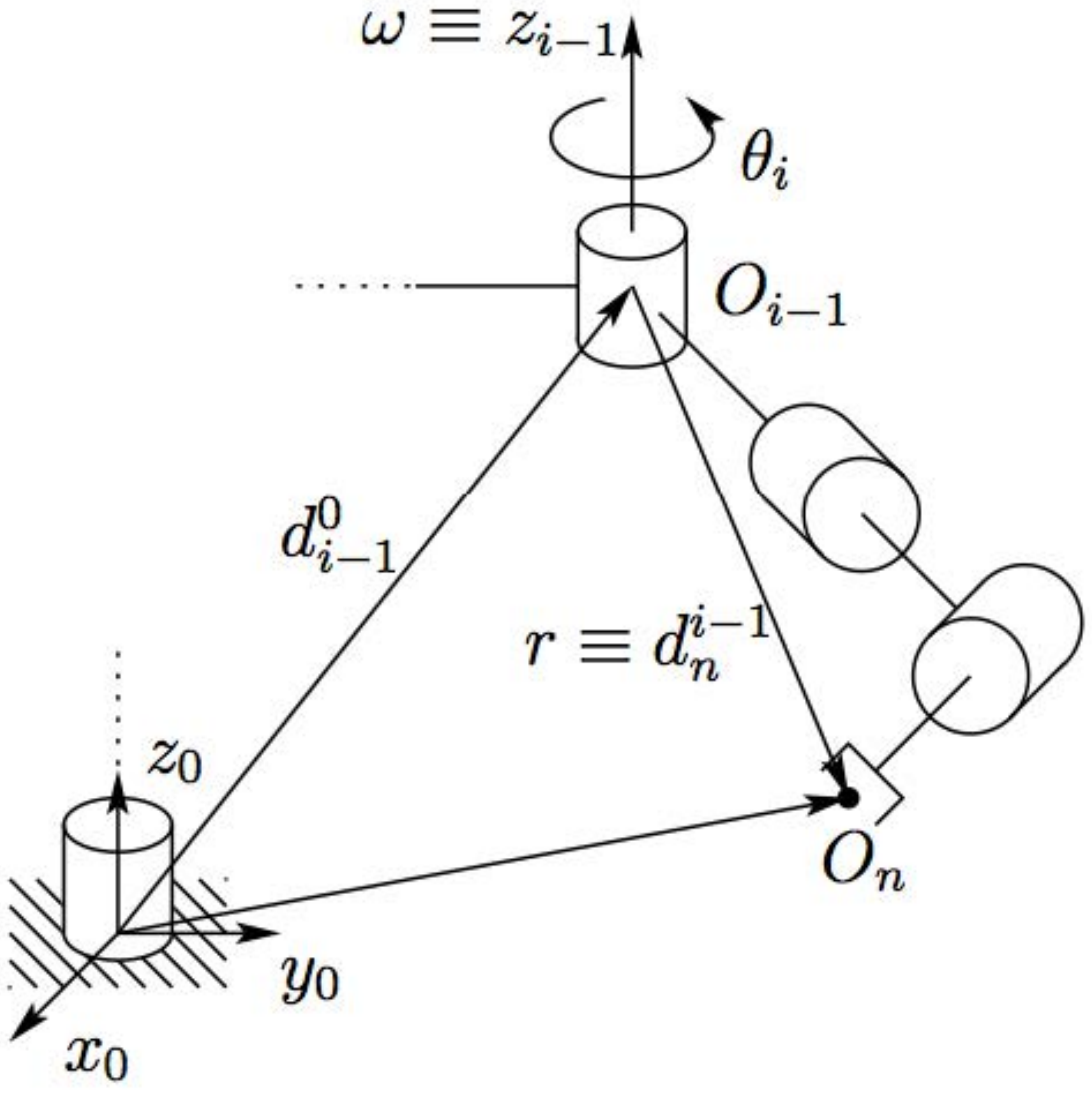
compute step direction

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$

Geometric

The Jacobian



A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

IK Procedure restated:

compute endpoint error

$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

compute step direction

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

perform step direction

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$

The Geometric Jacobian

A 6xN matrix

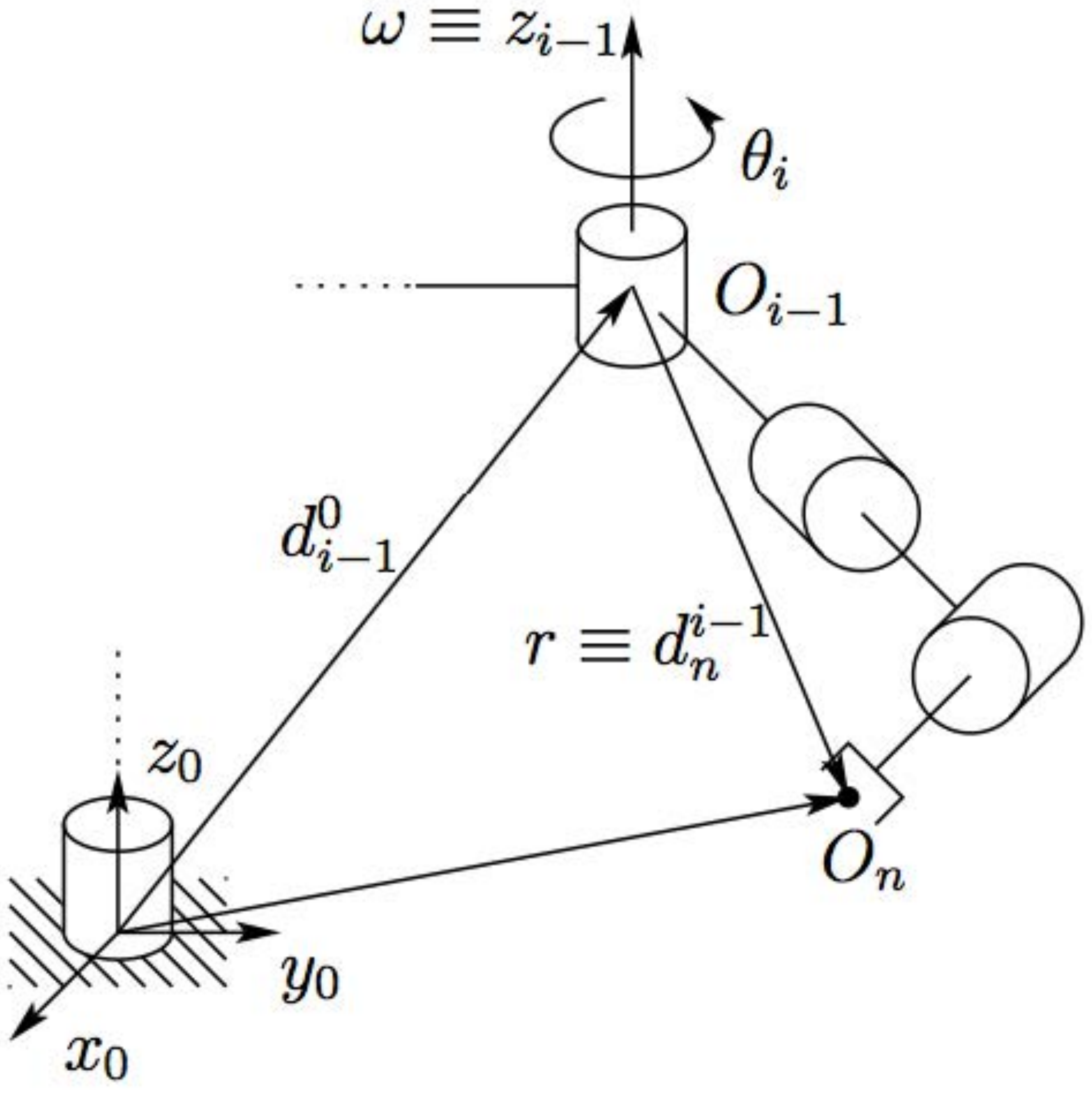
$$J = [J_1 J_2 \cdots J_n]$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$



IK Procedure restated:

compute endpoint error

$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

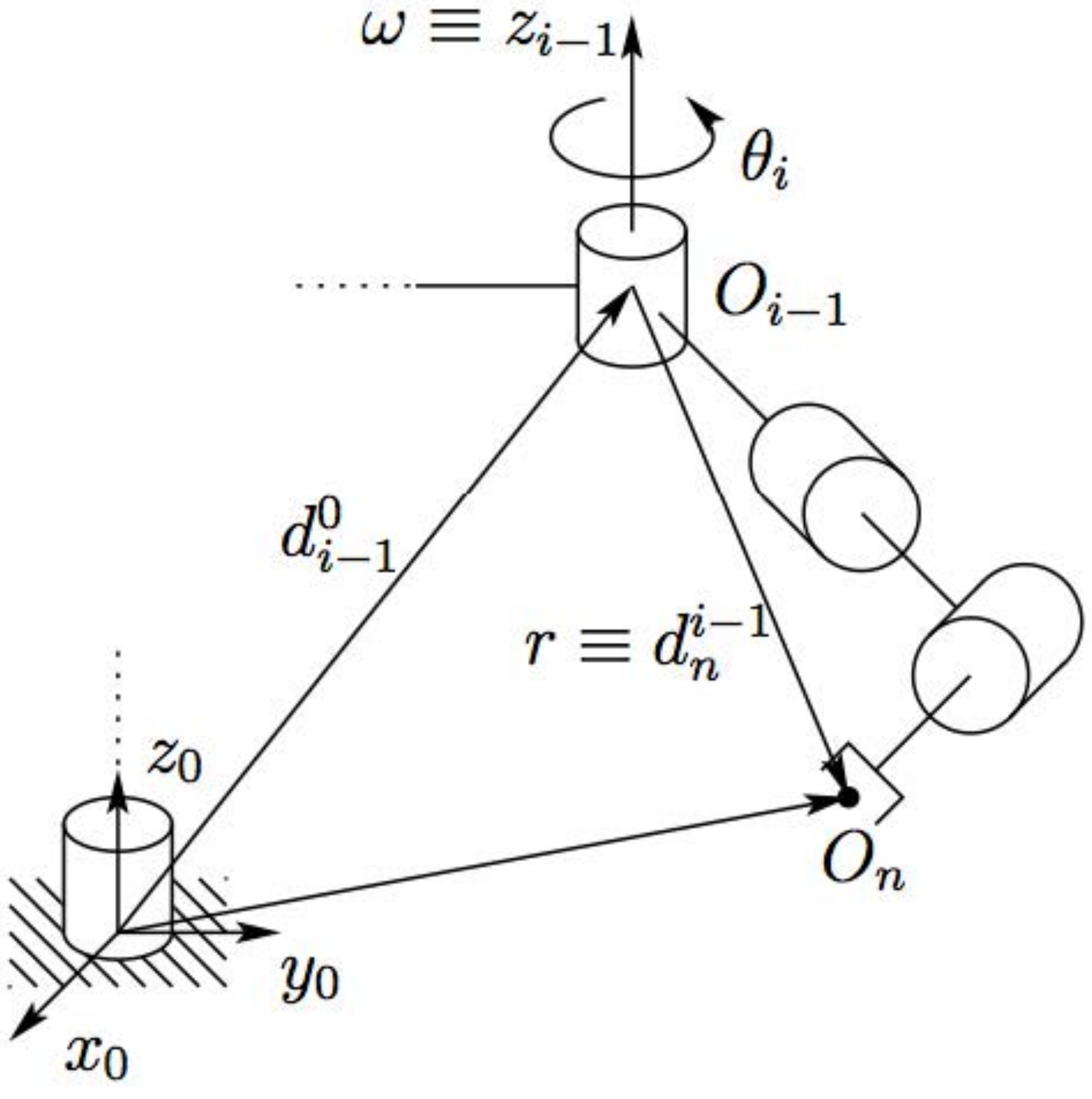
compute step direction

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

repeat

perform step direction

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$



The Jacobian

A 6xN matrix

$$J = [J_1 J_2 \cdots J_n]$$

J_i for a rotational joint

$$J_i = \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

J_i for a prismatic joint

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$$

when can we invert J(q)?

$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \gamma \Delta \mathbf{q}_n$$

Which Pseudoinverse

- For matrix A with dimensions N x M with full rank

- Left pseudoinverse, for when N > M, (i.e., "tall", less than than 6 DoFs)

$$A_{\text{left}}^{-1} = \left(A^T A\right)^{-1} A^T \qquad \text{s.t.} \qquad A_{\text{left}}^{-1} A = I_n$$

- Right pseudoinverse, for when N < M, (i.e., "wide", more than 6 DoFs)

$$A_{\text{right}}^{-1} = A^T \left(A A^T\right)^{-1} \qquad \text{s.t.} \qquad A A_{\text{right}}^{-1} = I_m$$

Maybe there is a simpler approach to IK?

Next lecture:
IK continued …
Manipulation New Frontiers