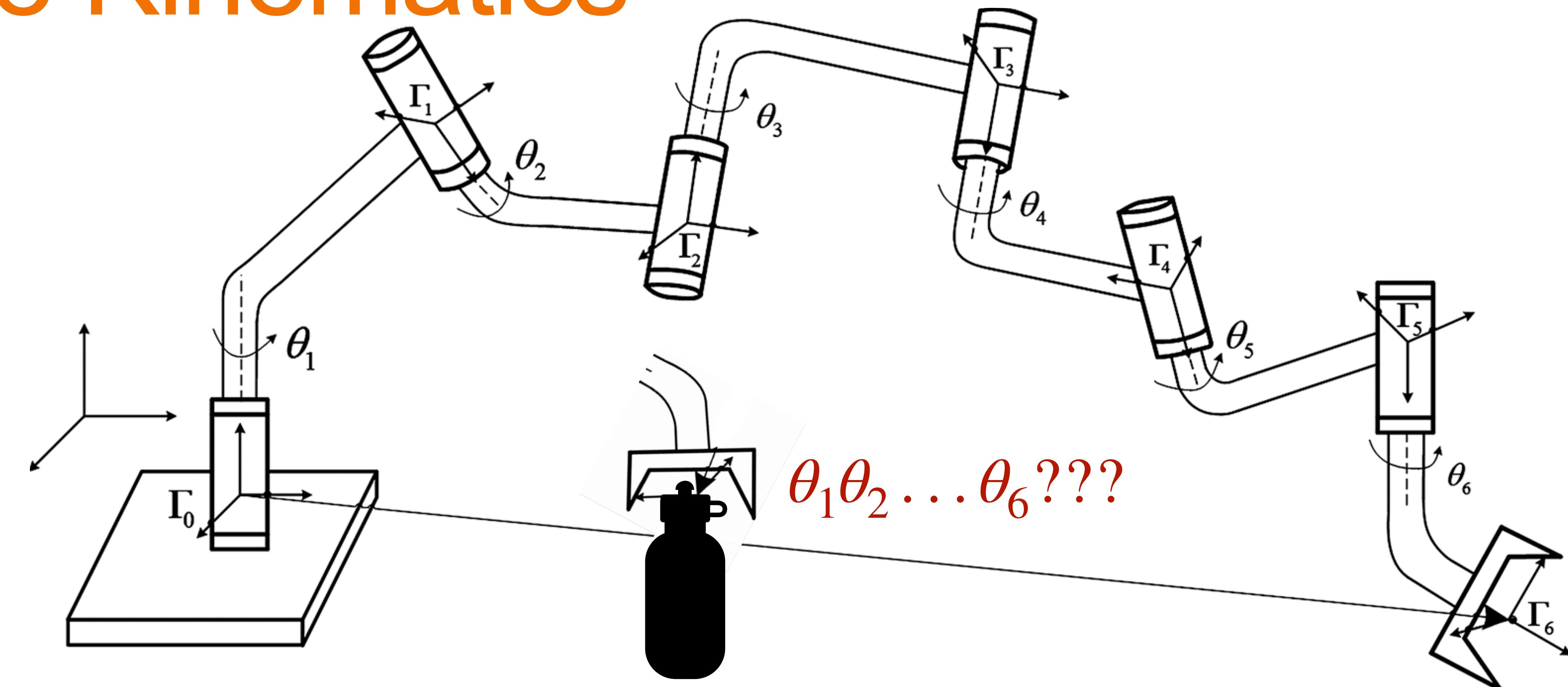


Lecture 08

Manipulation - III

Inverse Kinematics



Course Logistics

- Project 3 was posted on 02/12 and will be due 02/19.
- Project 4 will be posted on 02/19 and will be due 03/05 (yes 2 weeks).
- Quiz 4 will be posted tomorrow at **6pm** and will be due on Wed at noon.

Previously

PID Control

Error signal:

$$e(t) = x_{desired}(t) - x(t)$$

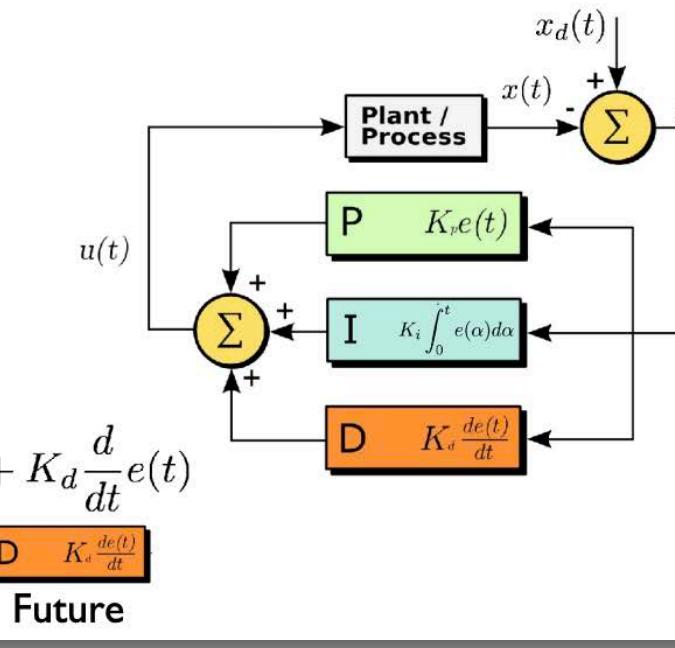
Control signal:

$$u(t) = K_p e(t) + K_i \int_0^t e(\alpha) d\alpha + K_d \frac{d}{dt} e(t)$$

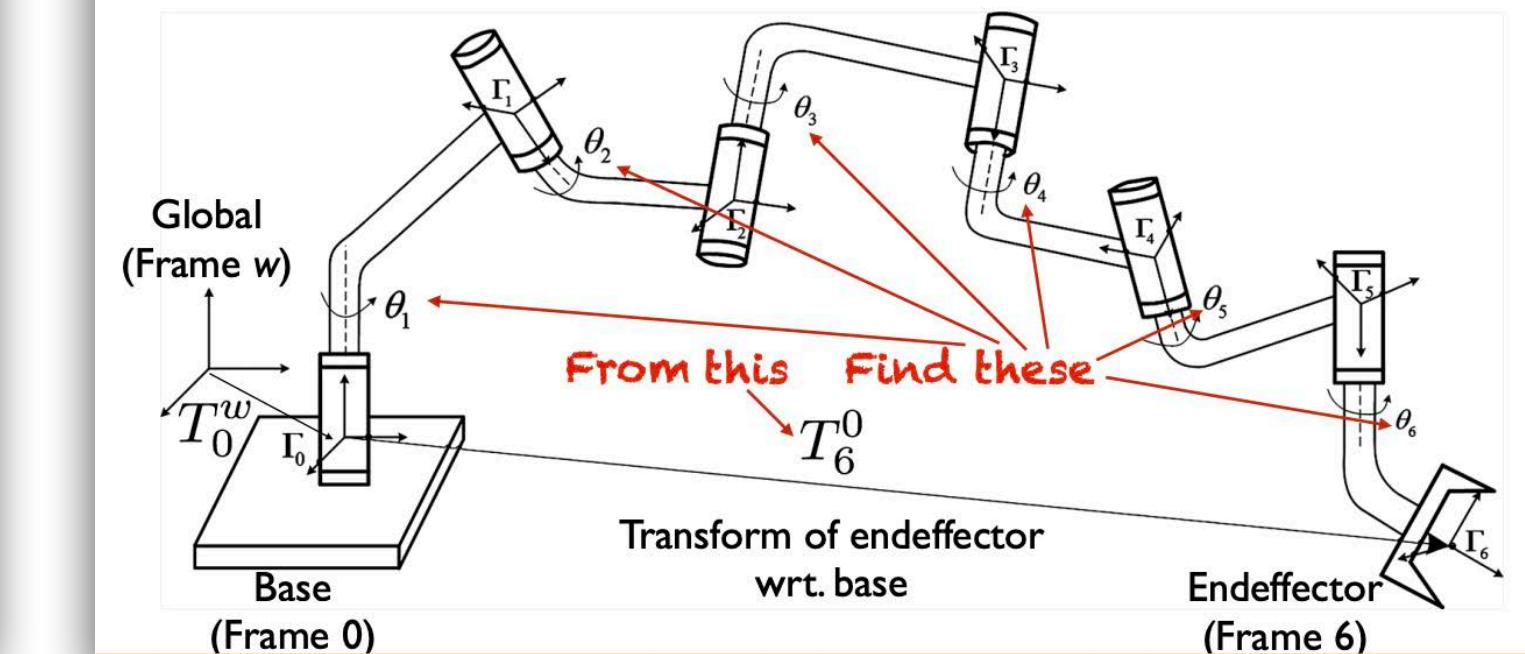
P $K_p e(t)$
Current

I $K_i \int_0^t e(\alpha) d\alpha$
Past

D $K_d \frac{de(t)}{dt}$
Future



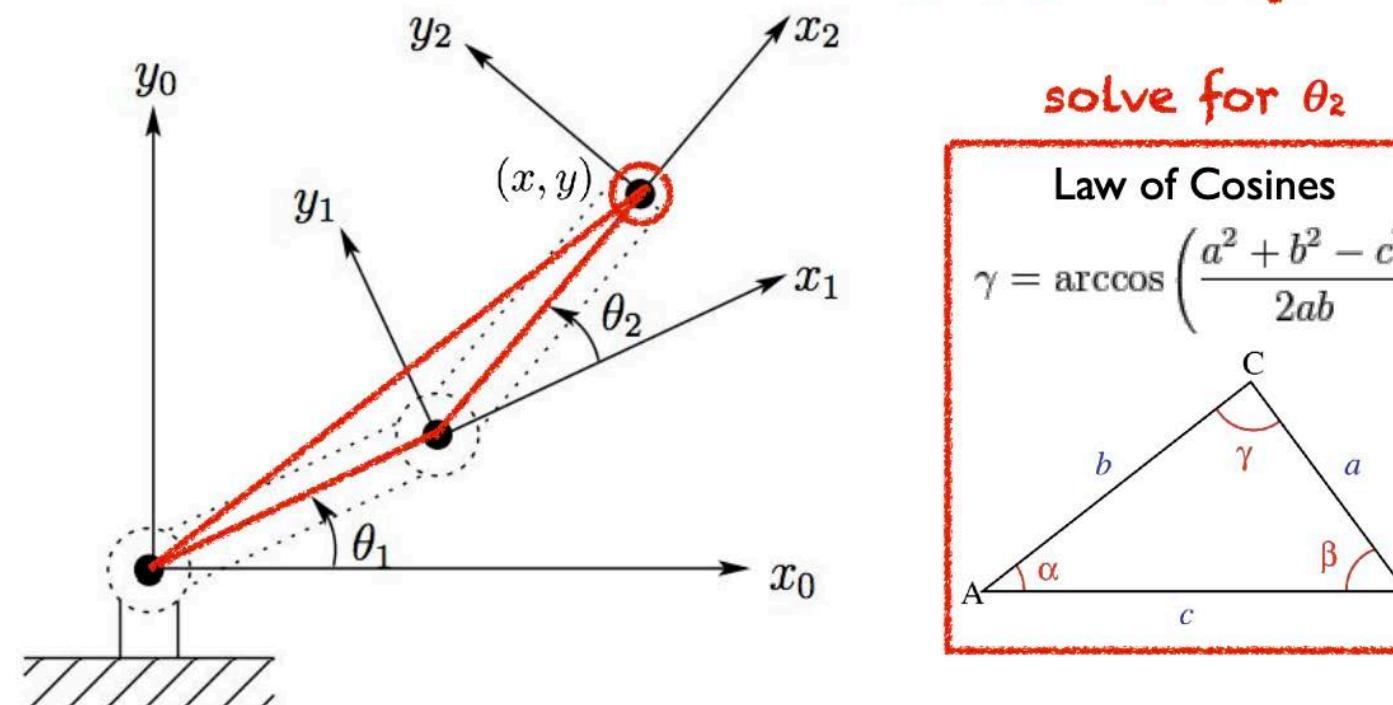
Inverse kinematics: one-to-many mapping of workspace endeffector pose to robot configuration



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

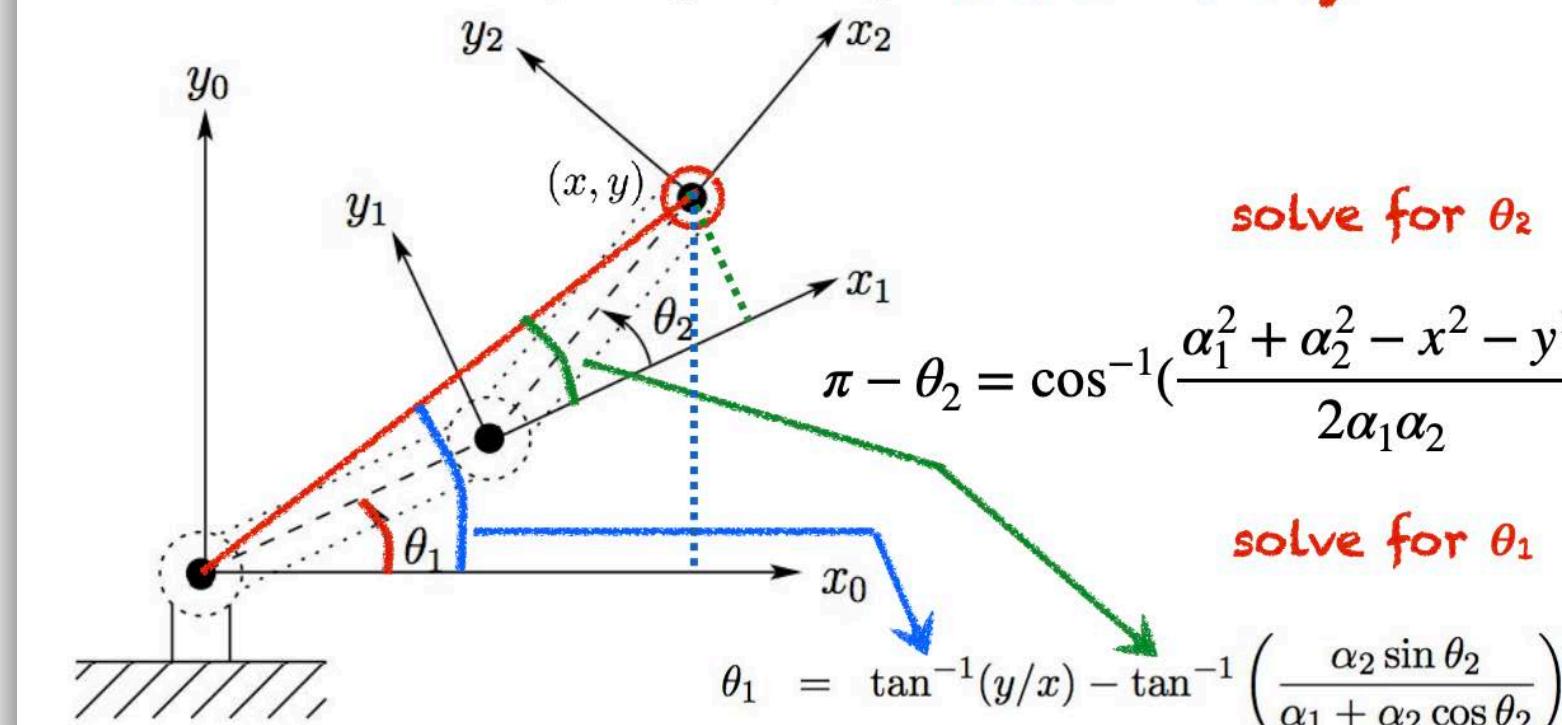
Inverse kinematics: $q = f^{-1}([\mathbf{o}^0_N, \mathbf{R}^0_N])$

$[\theta_1, \theta_2] = f^{-1}(x, y)$



Inverse kinematics: $q = f^{-1}([\mathbf{o}^0_N, \mathbf{R}^0_N])$

$[\theta_1, \theta_2] = f^{-1}(x, y)$



Inverse Kinematics: 3D

Configuration

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_6 \end{bmatrix}$$

$T_n^0(q_1, \dots, q_n) = H$ ← Transform from endeffector

$$H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$$

Closed form solution?

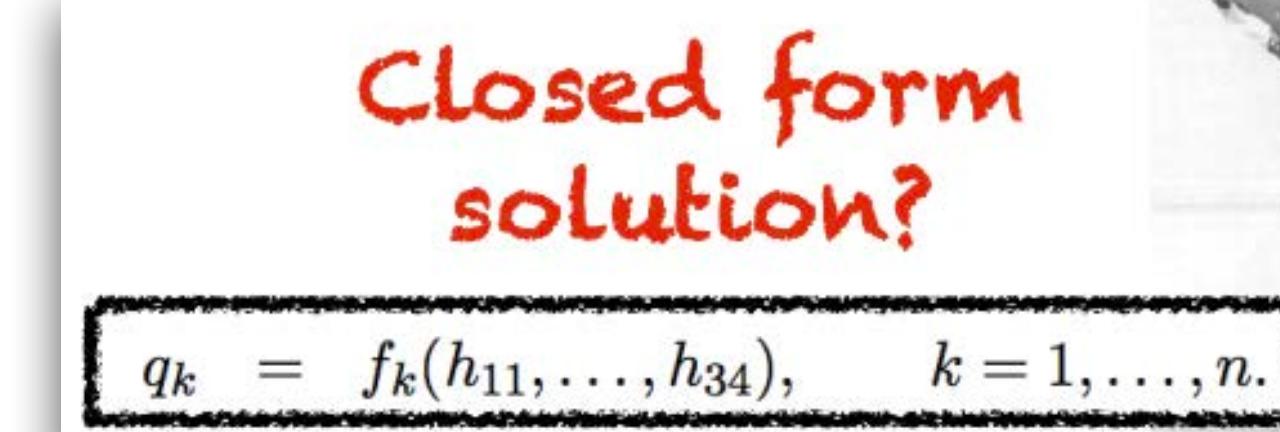
$$q_k = f_k(h_{11}, \dots, h_{34}), \quad k = 1, \dots, n.$$

$$H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & o_x \\ r_{21} & r_{22} & r_{23} & o_y \\ r_{31} & r_{32} & r_{33} & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

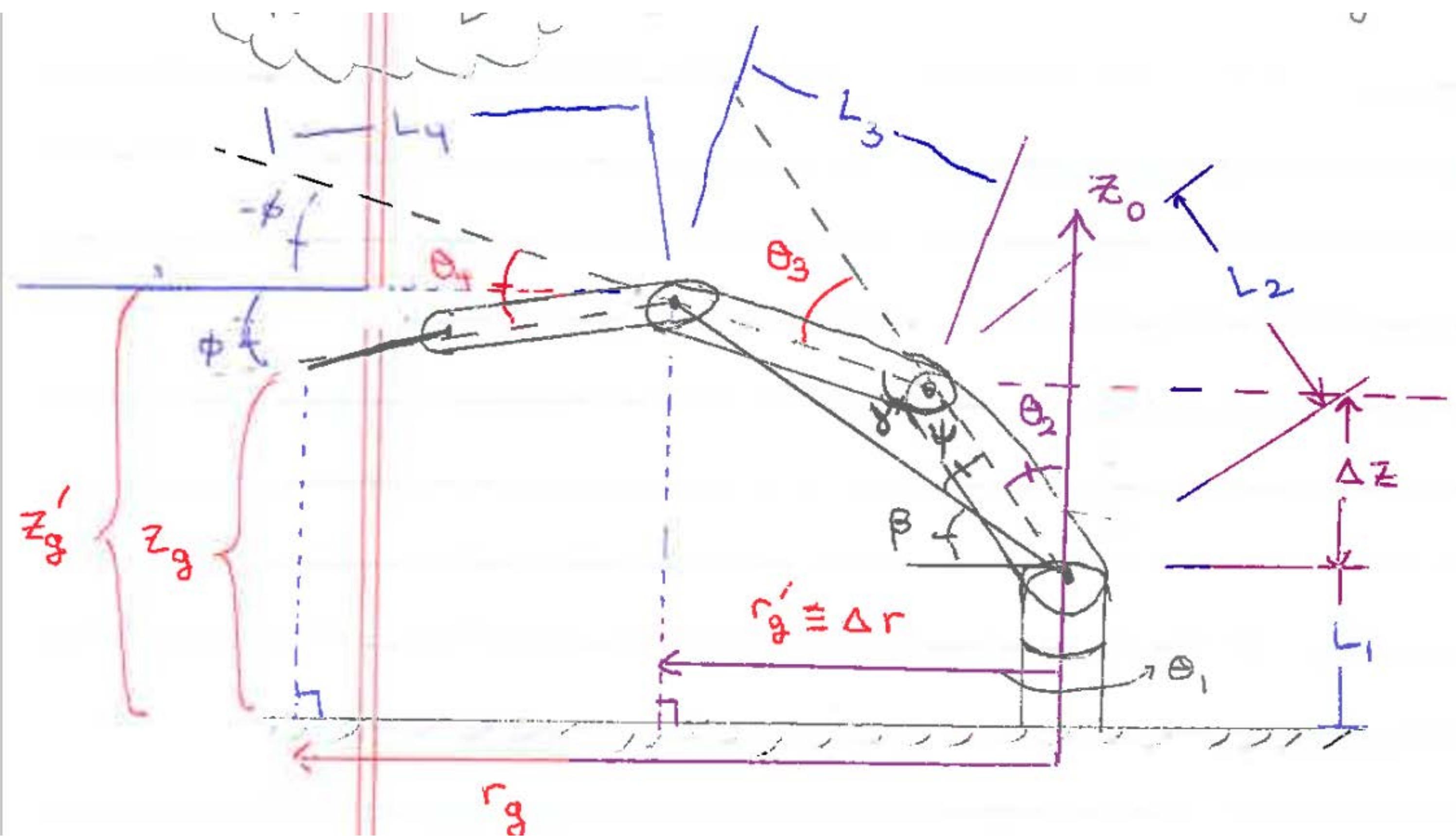
6 DOF position and orientation of endeffector

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



RexArm from the above videos



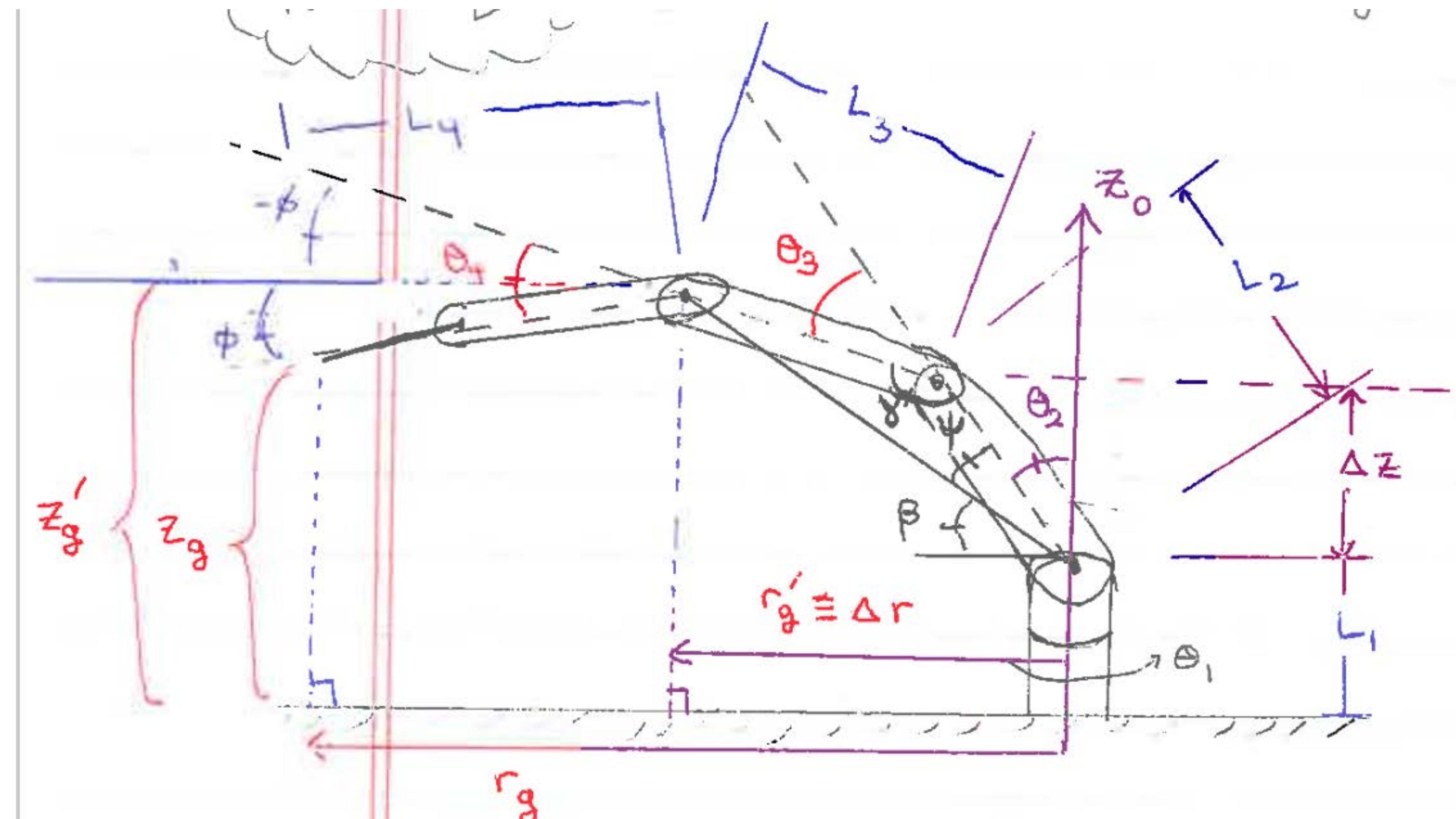
Find: configuration
 $q = [\theta_1 \theta_2 \theta_3 \theta_4]$
as robot joint angles

Given:

Find: configuration

$$\mathbf{q} = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4]$$

as robot joint angles



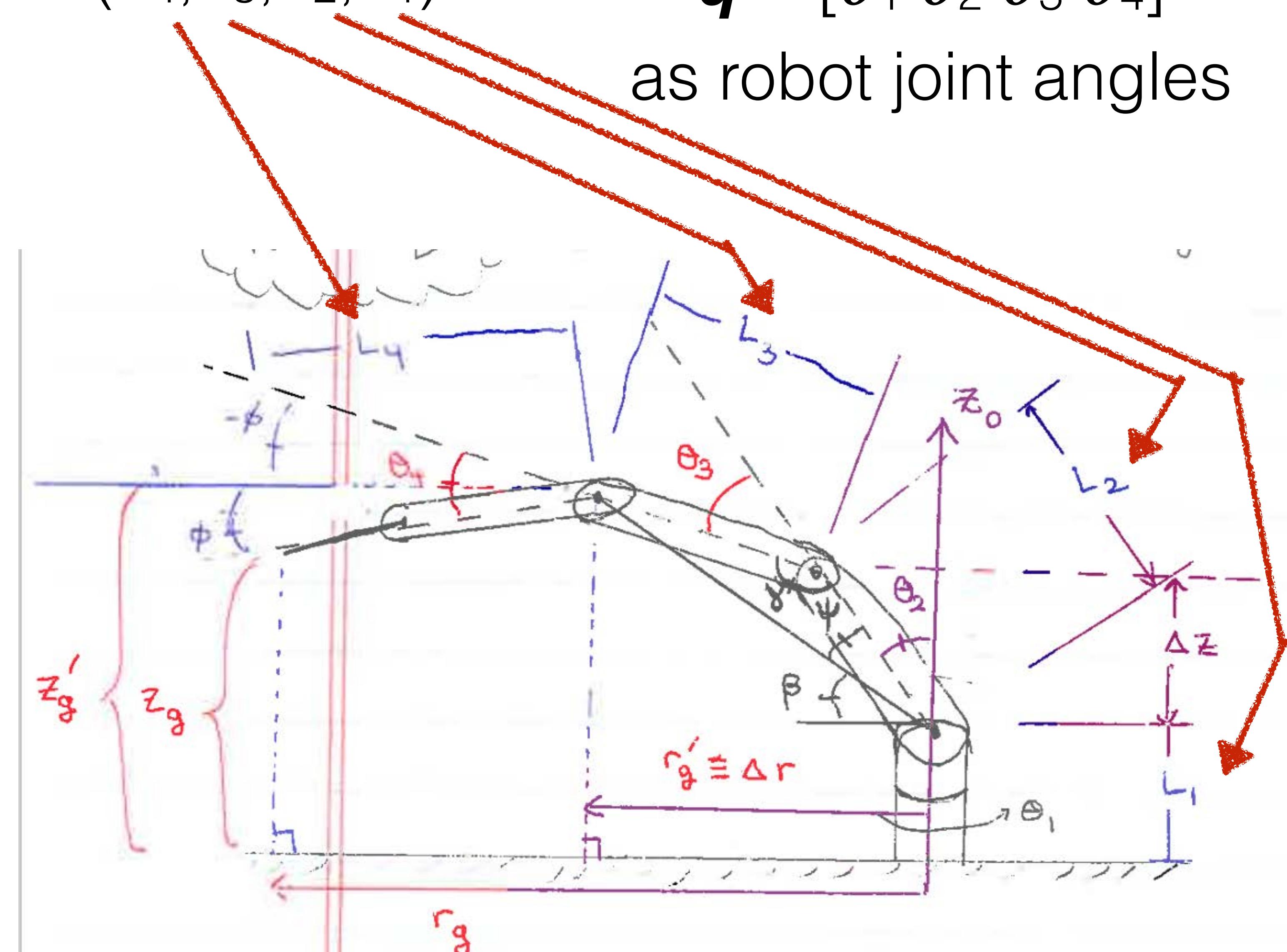
Given:

link lengths (L_4, L_3, L_2, L_1)

Find: configuration

$$\mathbf{q} = [\theta_1 \theta_2 \theta_3 \theta_4]$$

as robot joint angles

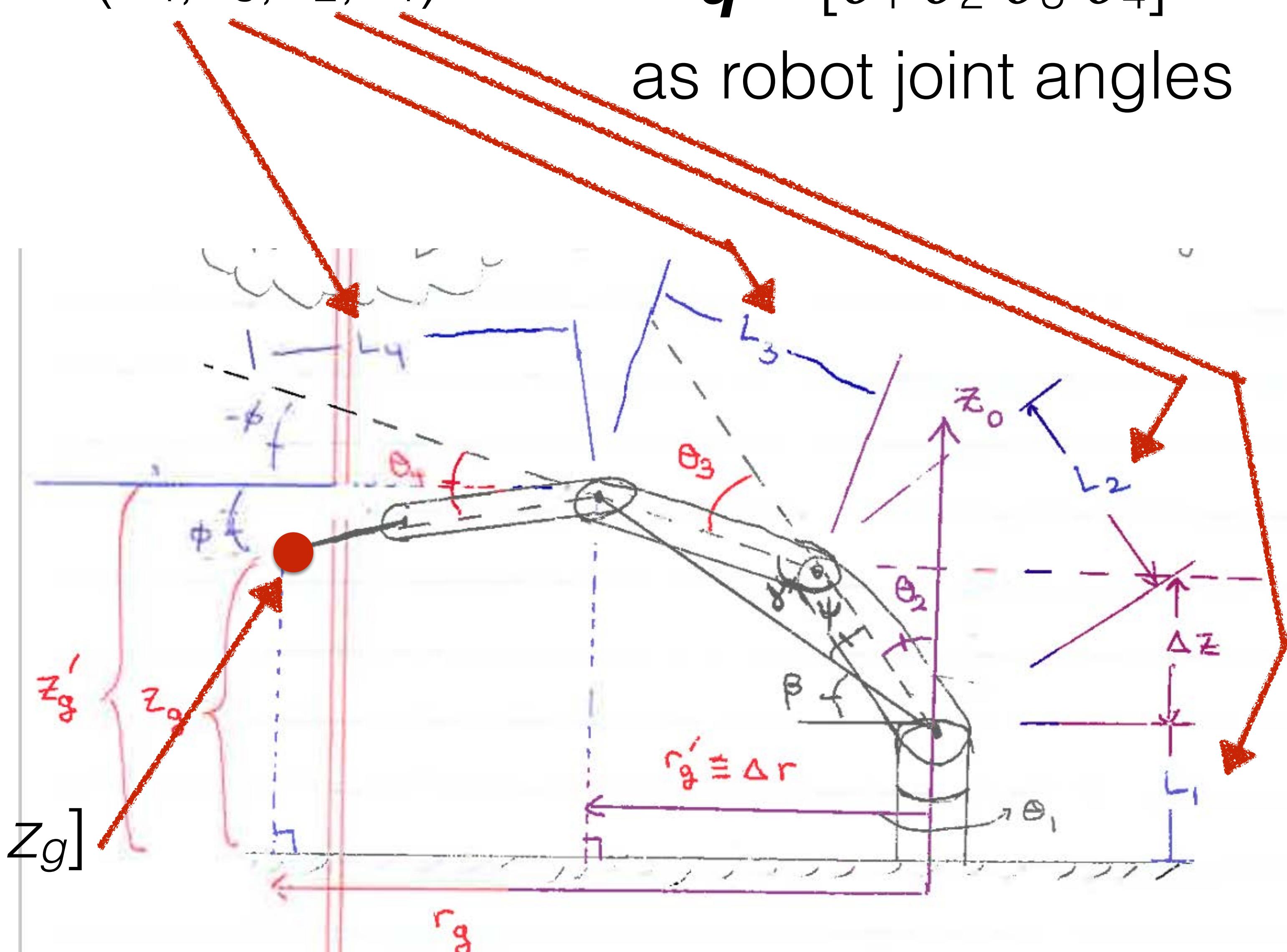


Given:

link lengths (L_4, L_3, L_2, L_1)

endeffector position $[x_g \ y_g \ z_g]$
wrt. base frame

Find: configuration
 $q = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4]$
as robot joint angles



Given:

endeffector orientation ϕ

as angle wrt. plane
centered at \mathbf{o}_3 and
parallel to ground plane

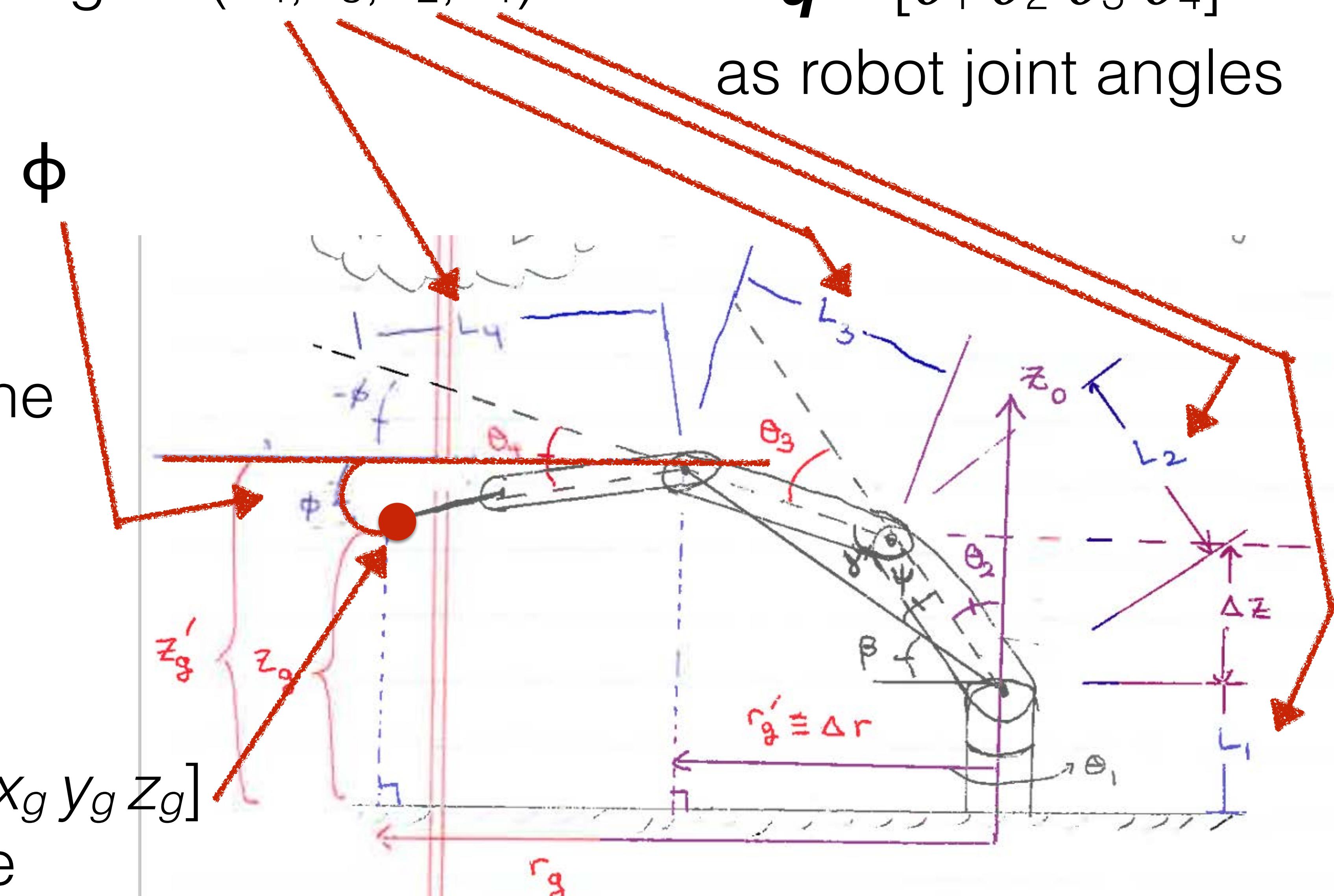
endeffector position $[x_g \ y_g \ z_g]$
wrt. base frame

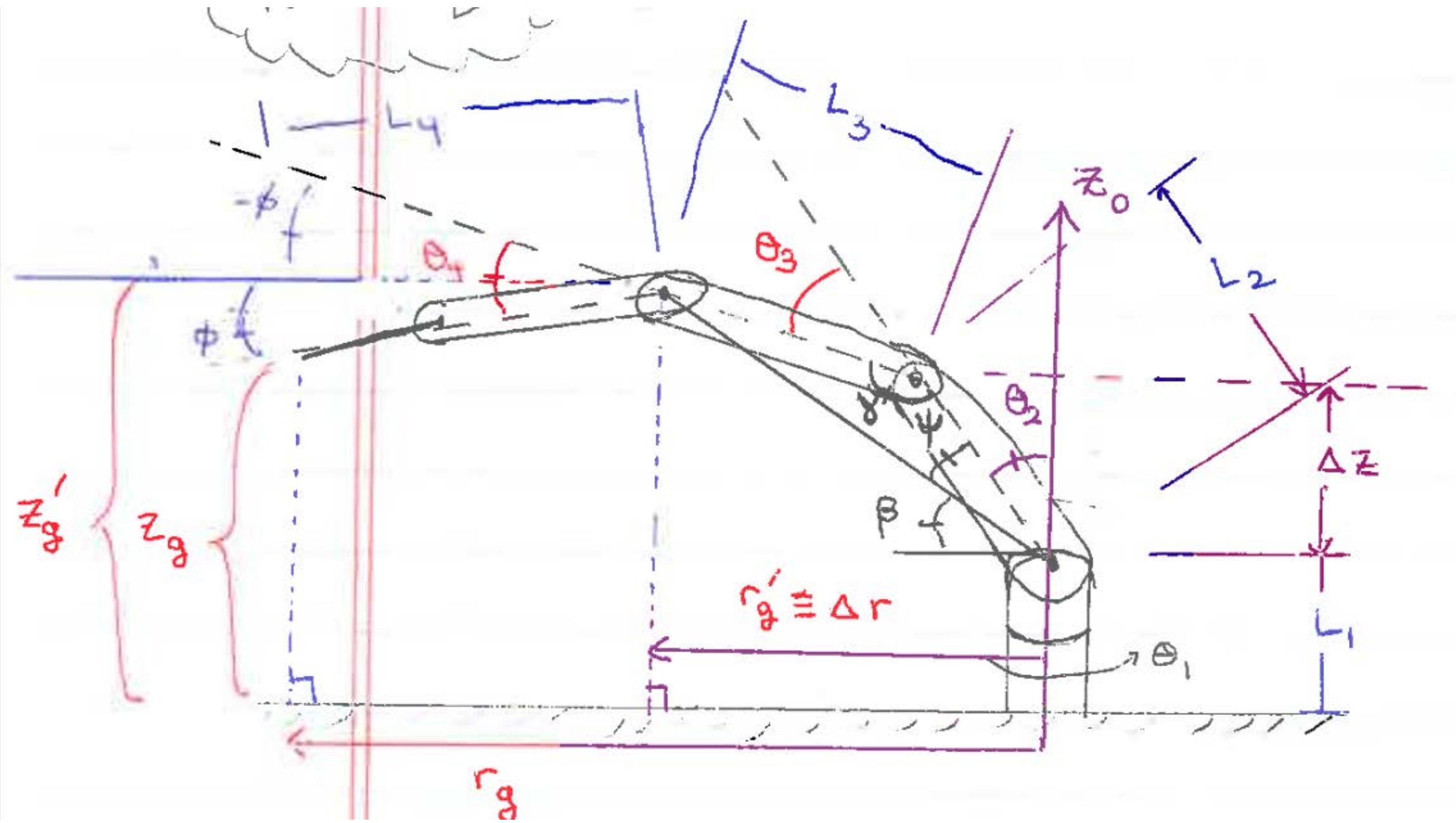
link lengths (L_4, L_3, L_2, L_1)

Find: configuration

$$\mathbf{q} = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4]$$

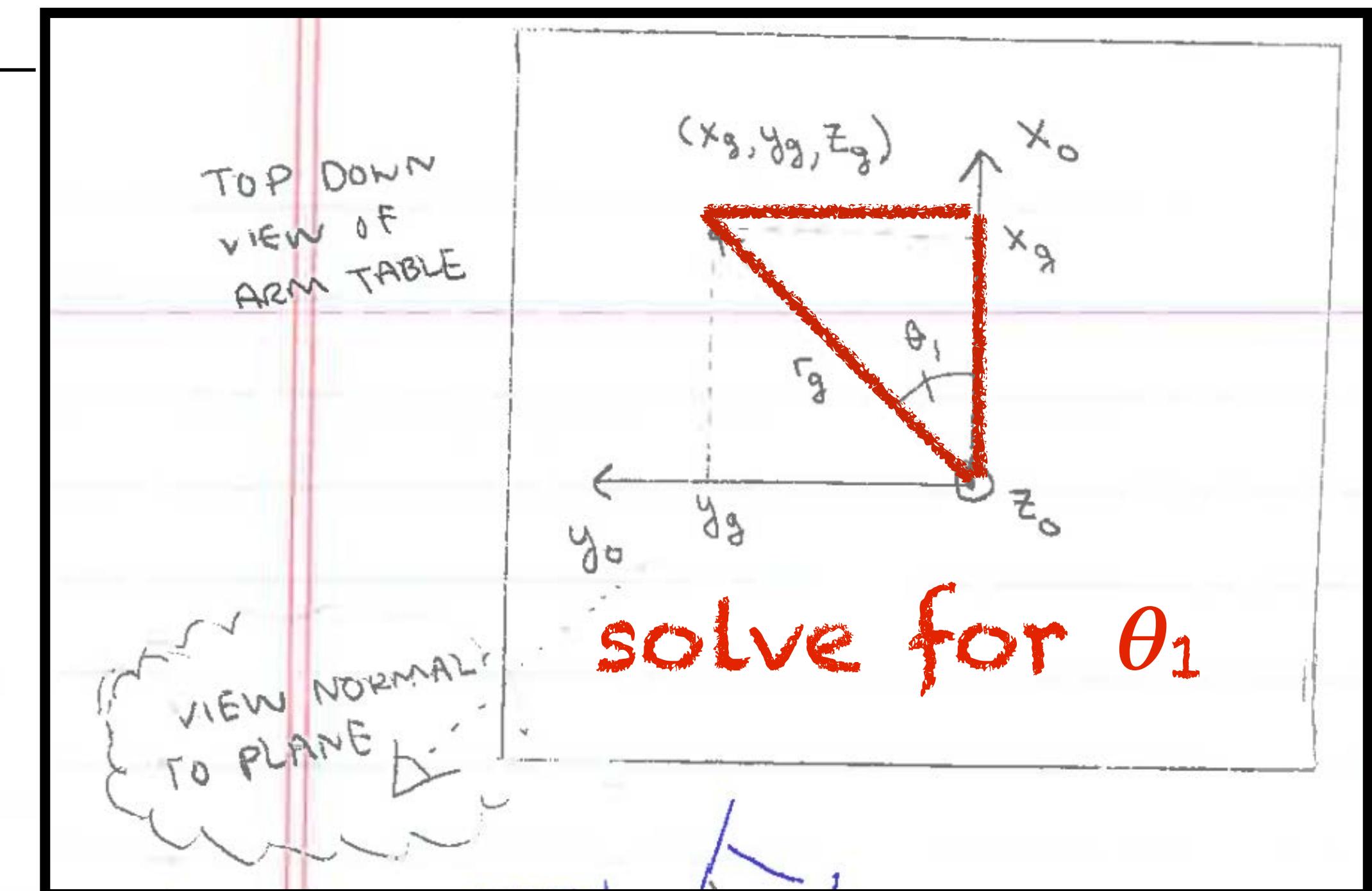
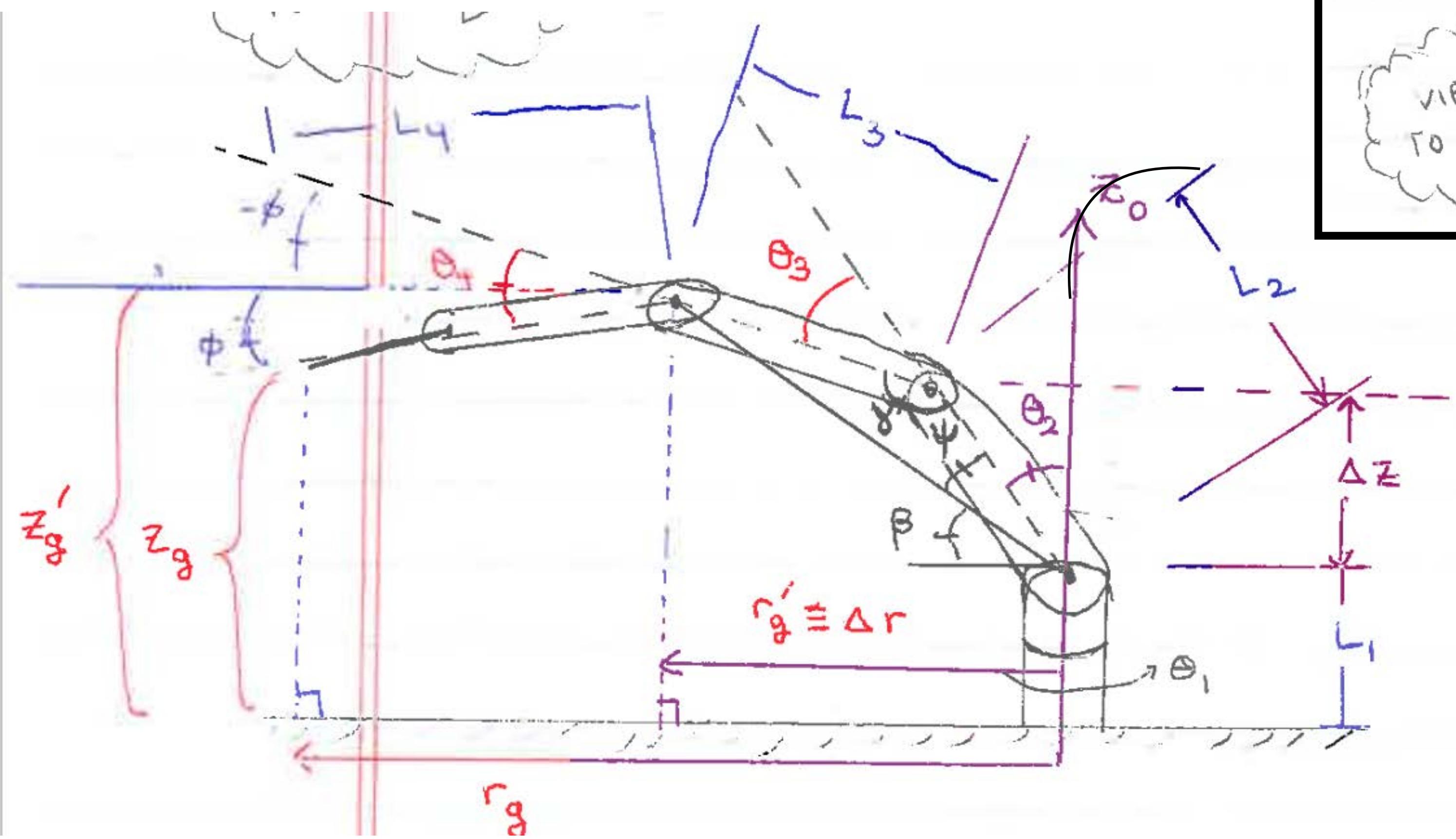
as robot joint angles



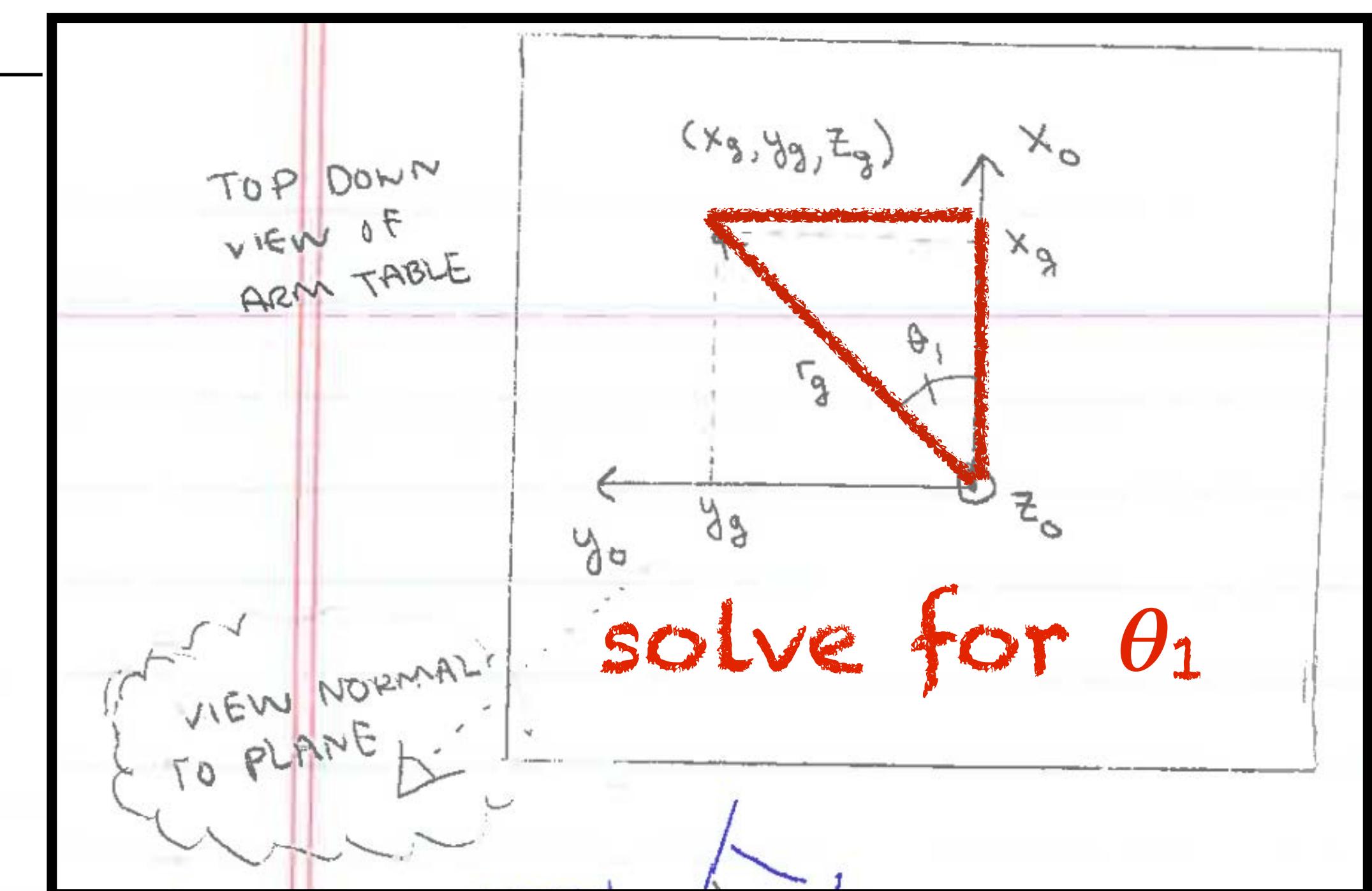
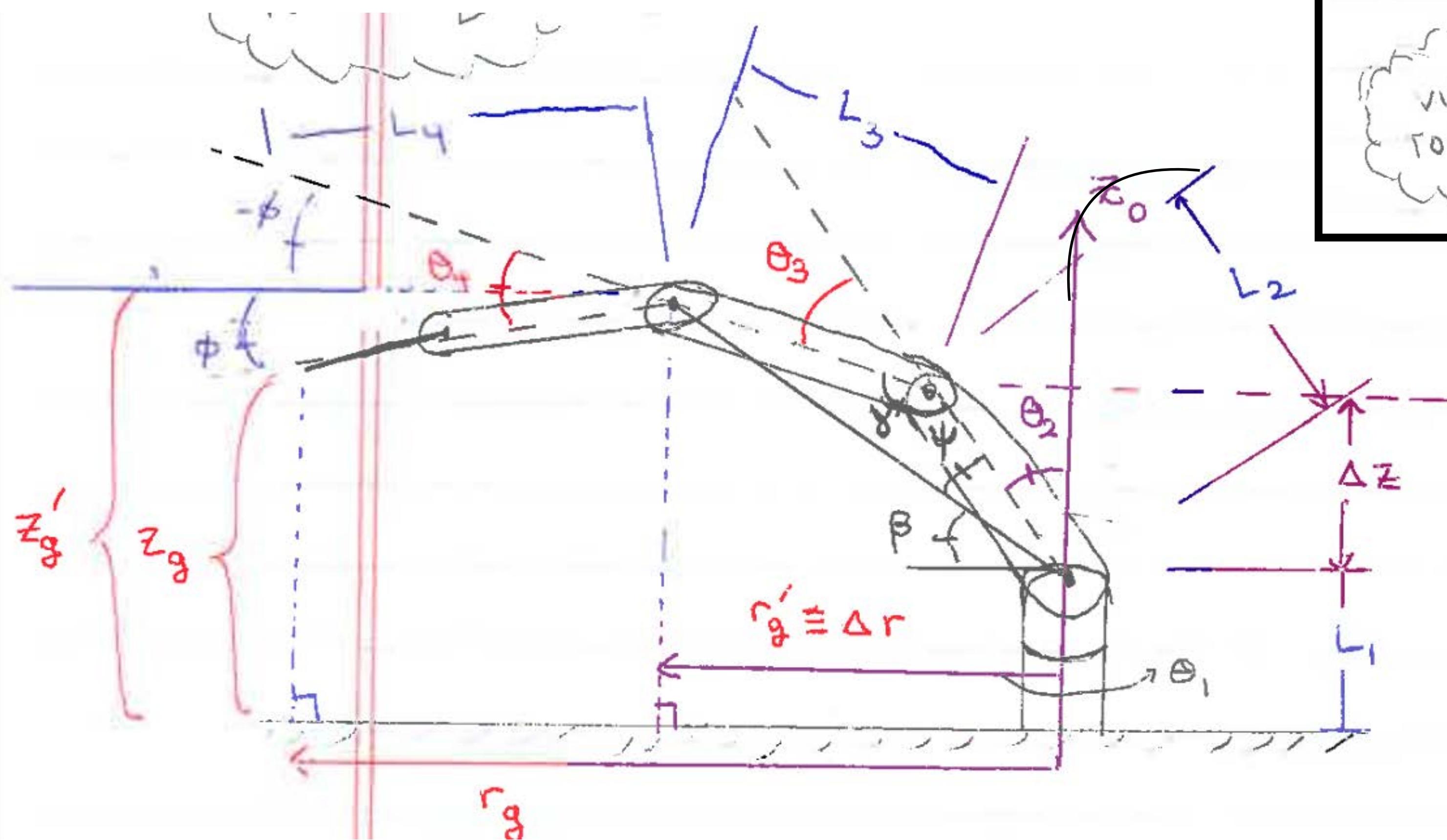
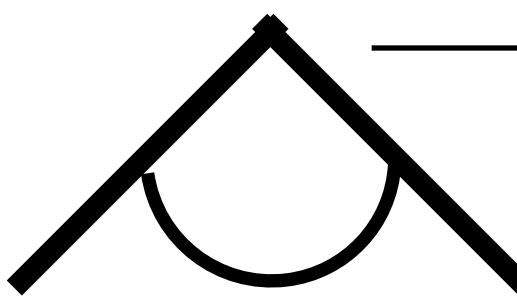


solve for θ_1

overhead view



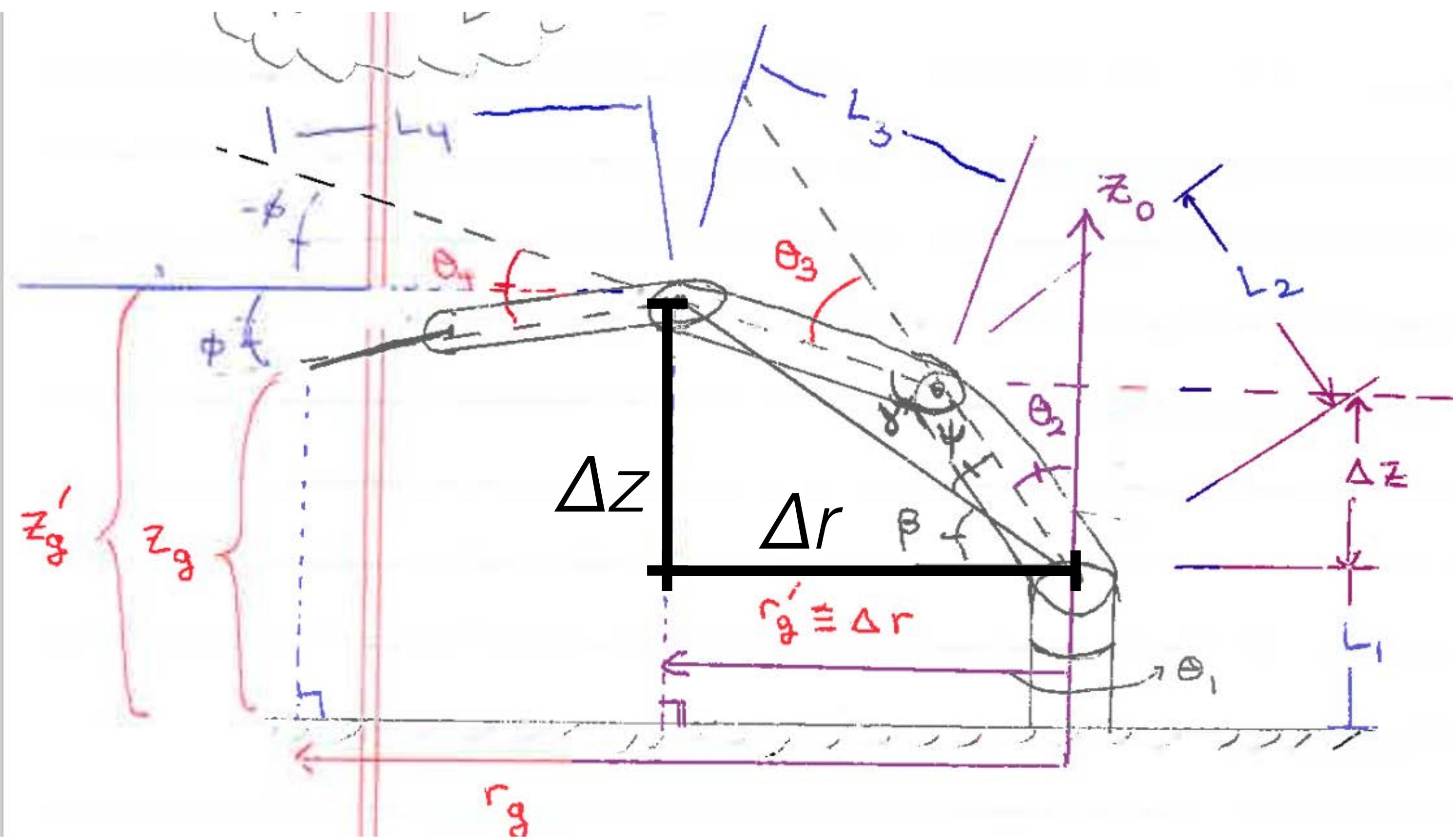
overhead view



$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_1

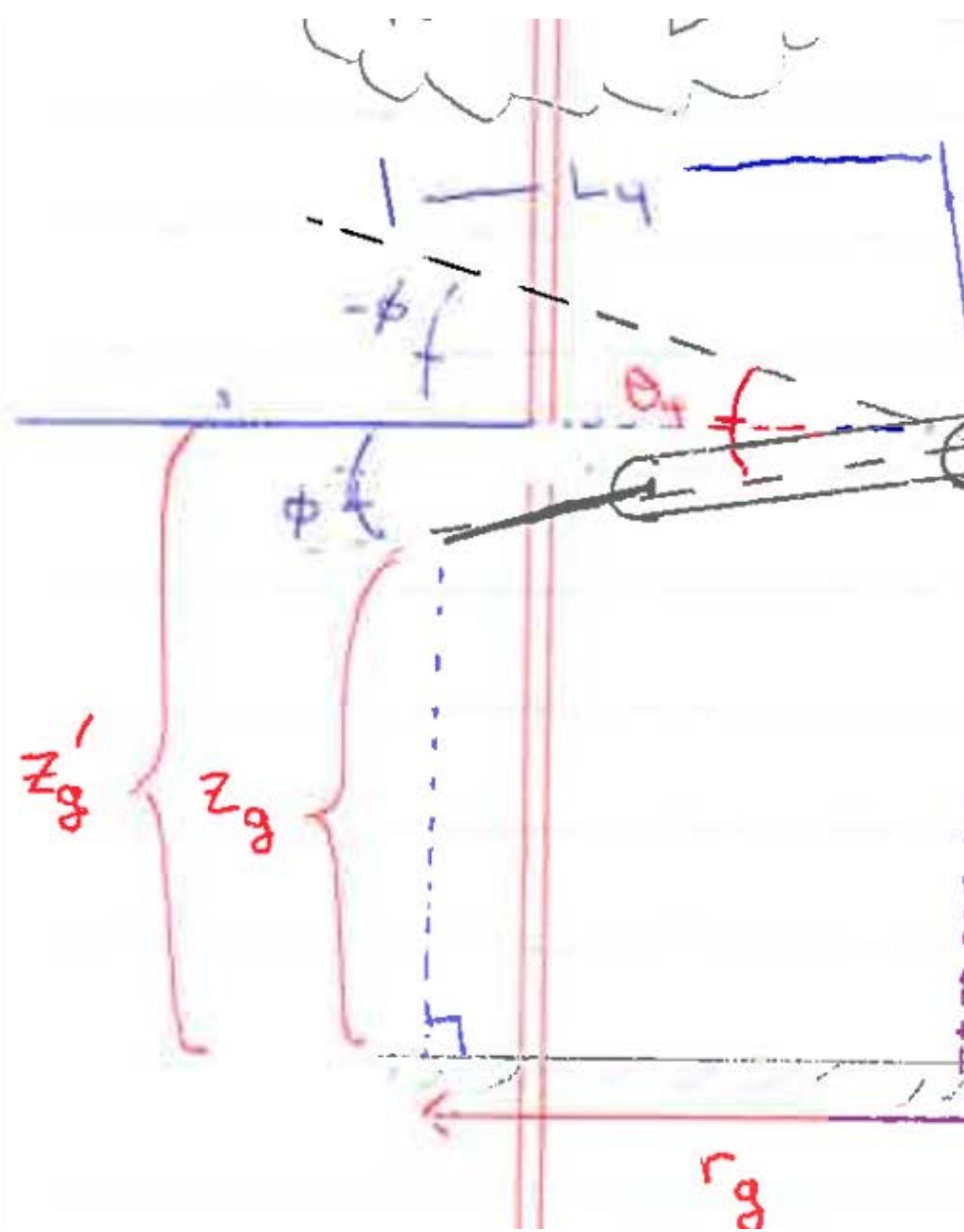
$$\theta_1 = \text{atan2}(y_g, x_g)$$



solve for θ_3

solve for θ_1

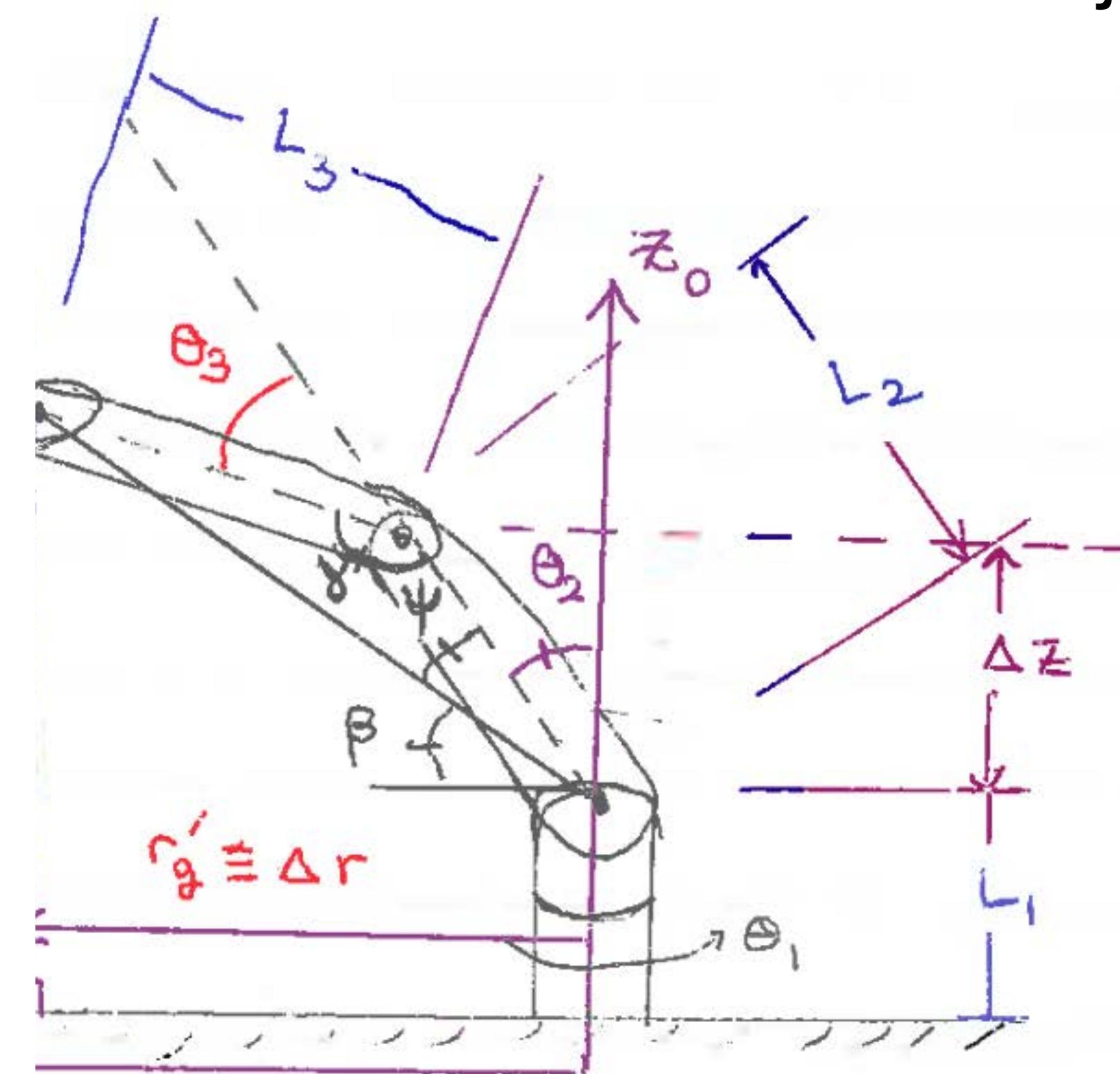
$$\theta_1 = \text{atan2}(y_g, x_g)$$



Decoupling:

separate endeffector from
rest of the robot at last joint

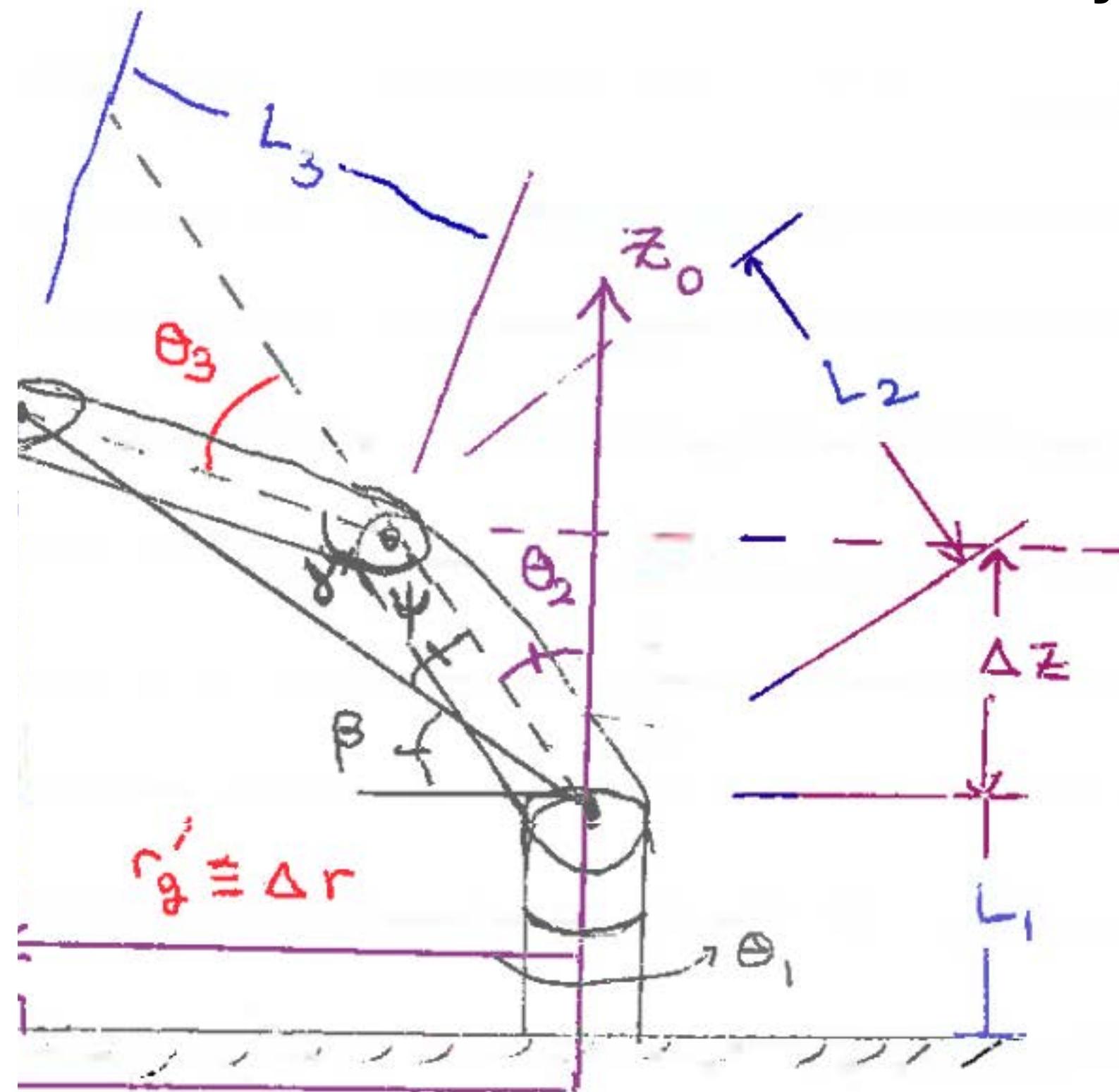
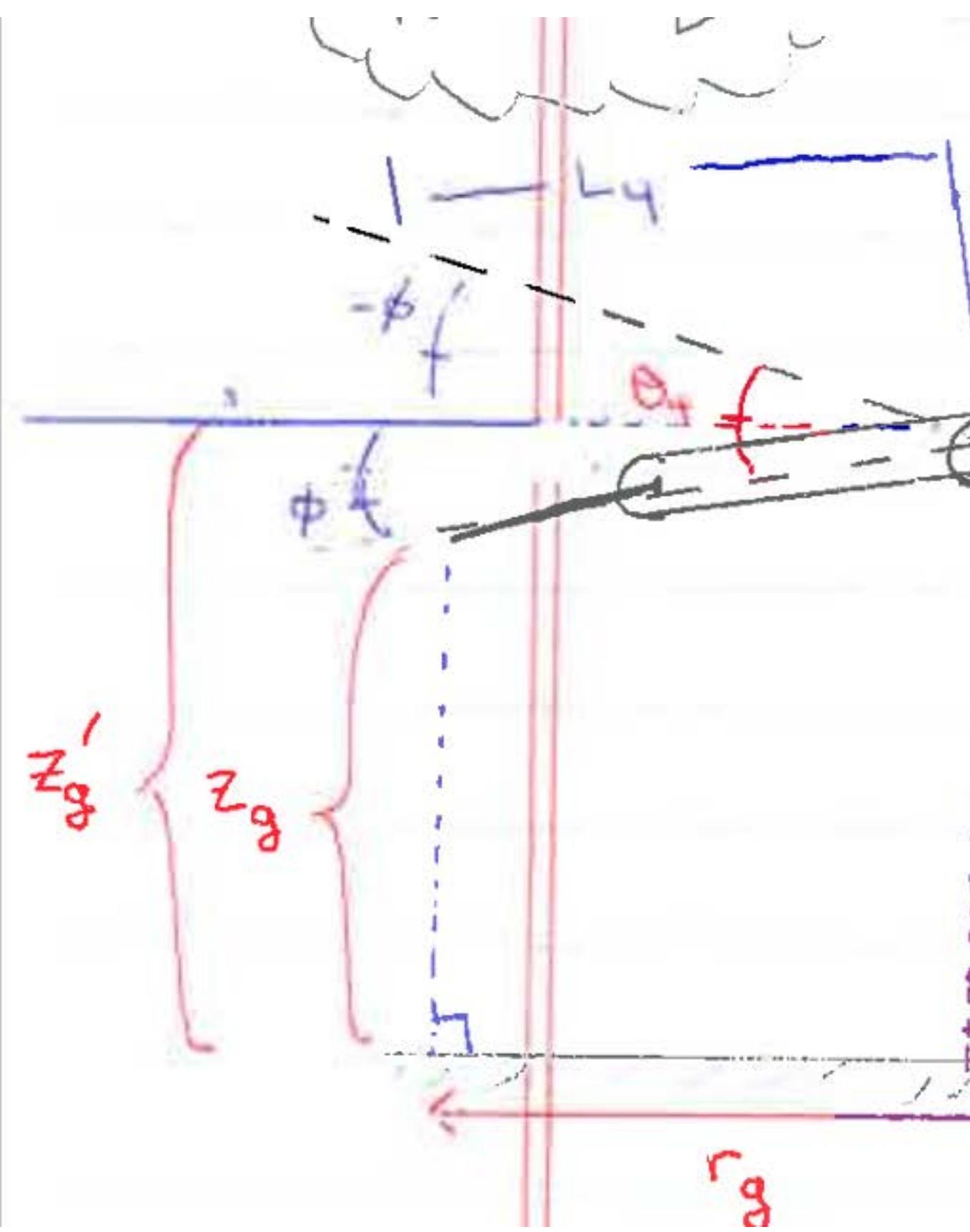
solve for θ_3



solve for θ_1

$$\theta_1 = \text{atan2} (y_g, x_g)$$

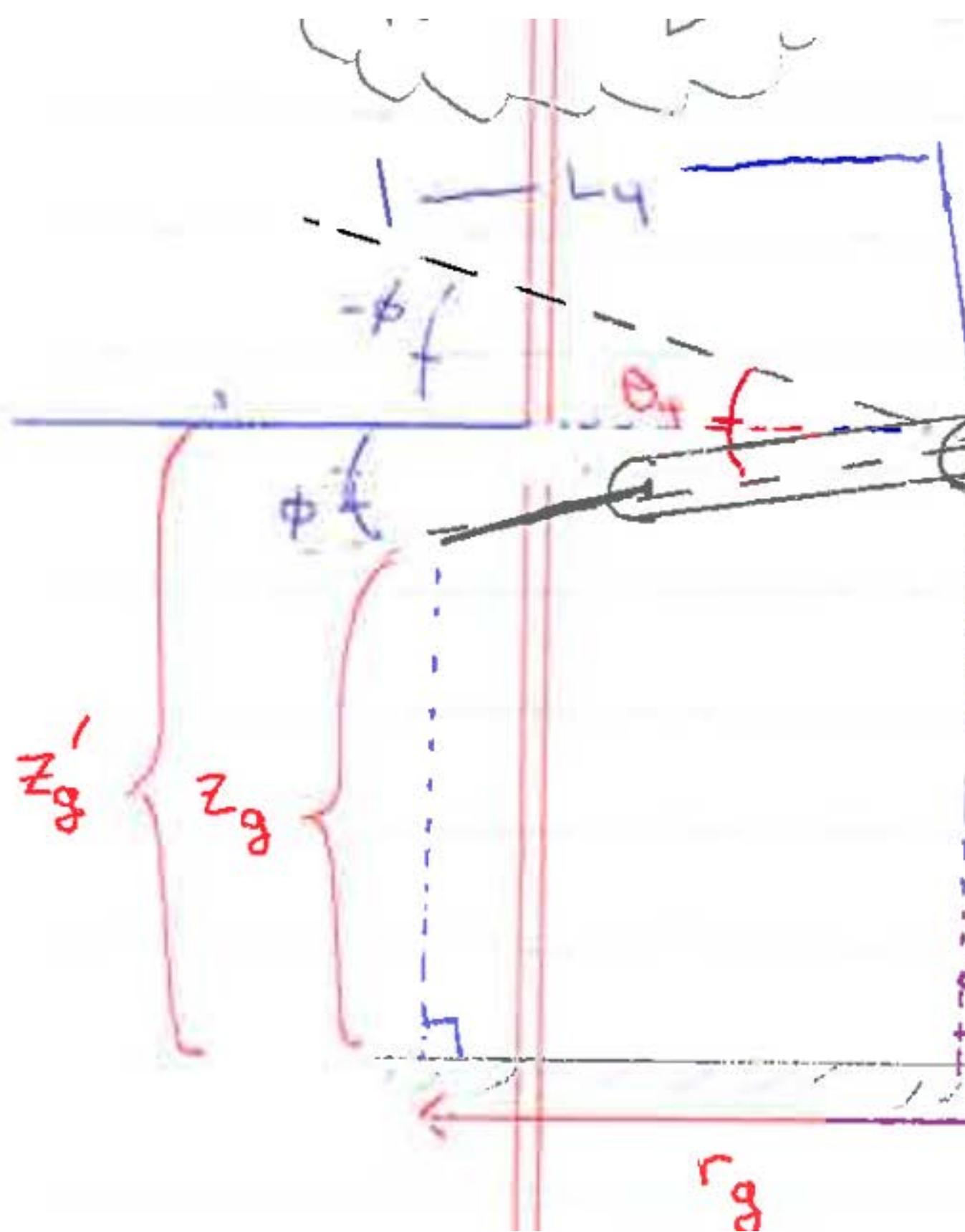
Decoupling: solve for θ_3
separate endeffector from
rest of the robot at last joint



and...

solve for θ_1

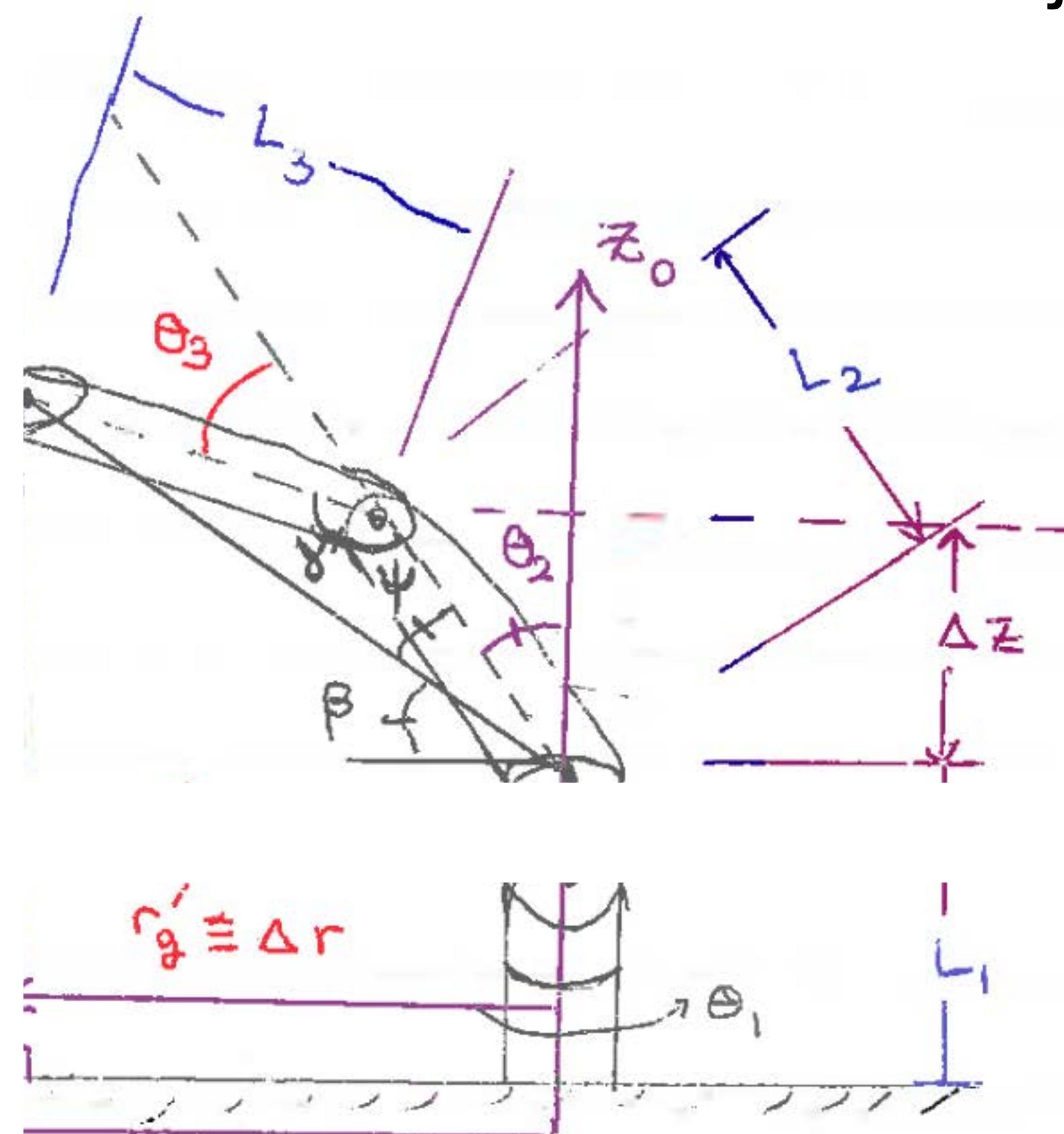
$$\theta_1 = \text{atan2}(y_g, x_g)$$



Decoupling:

separate endeffector from
rest of the robot at last joint

solve for θ_3

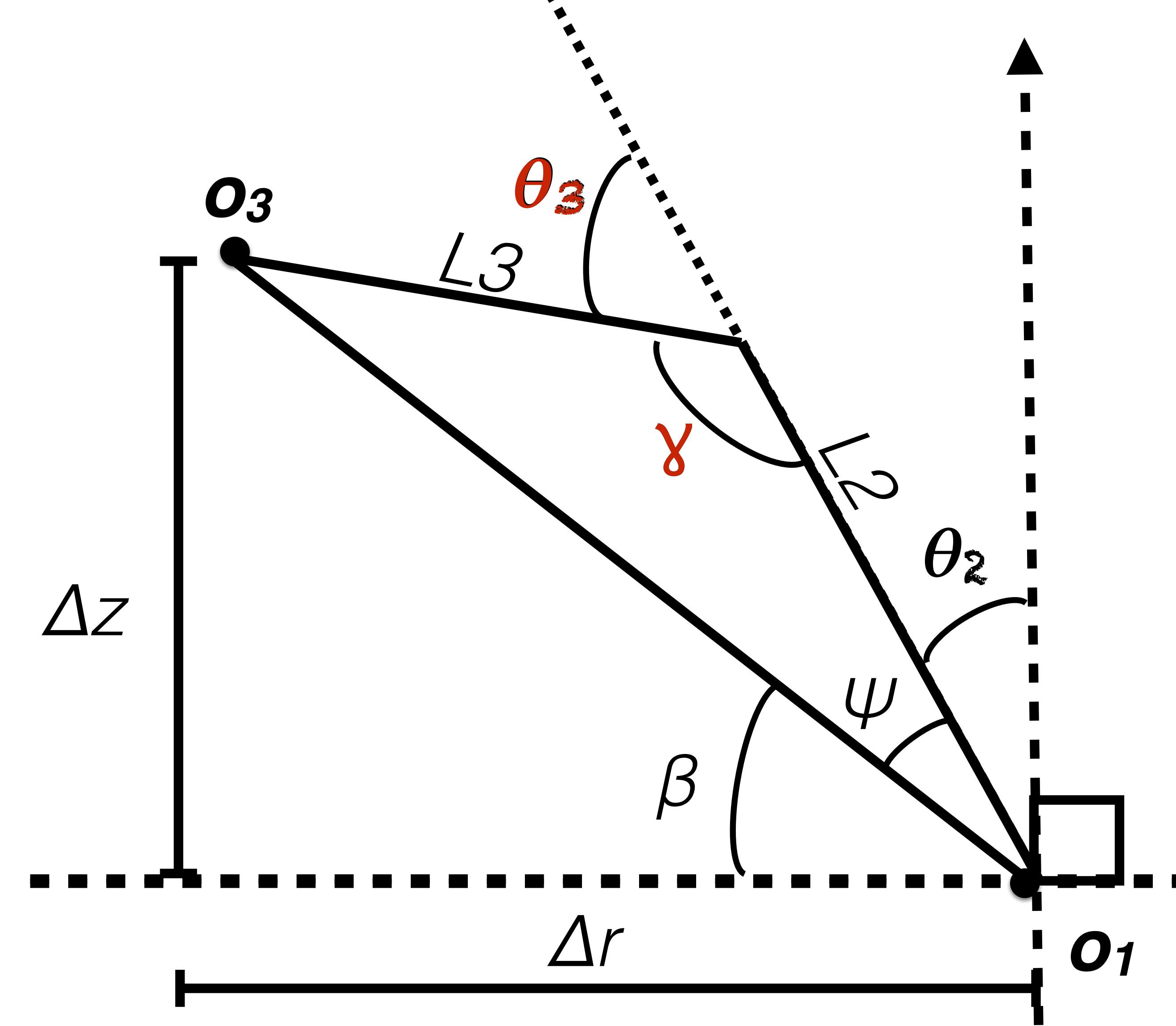
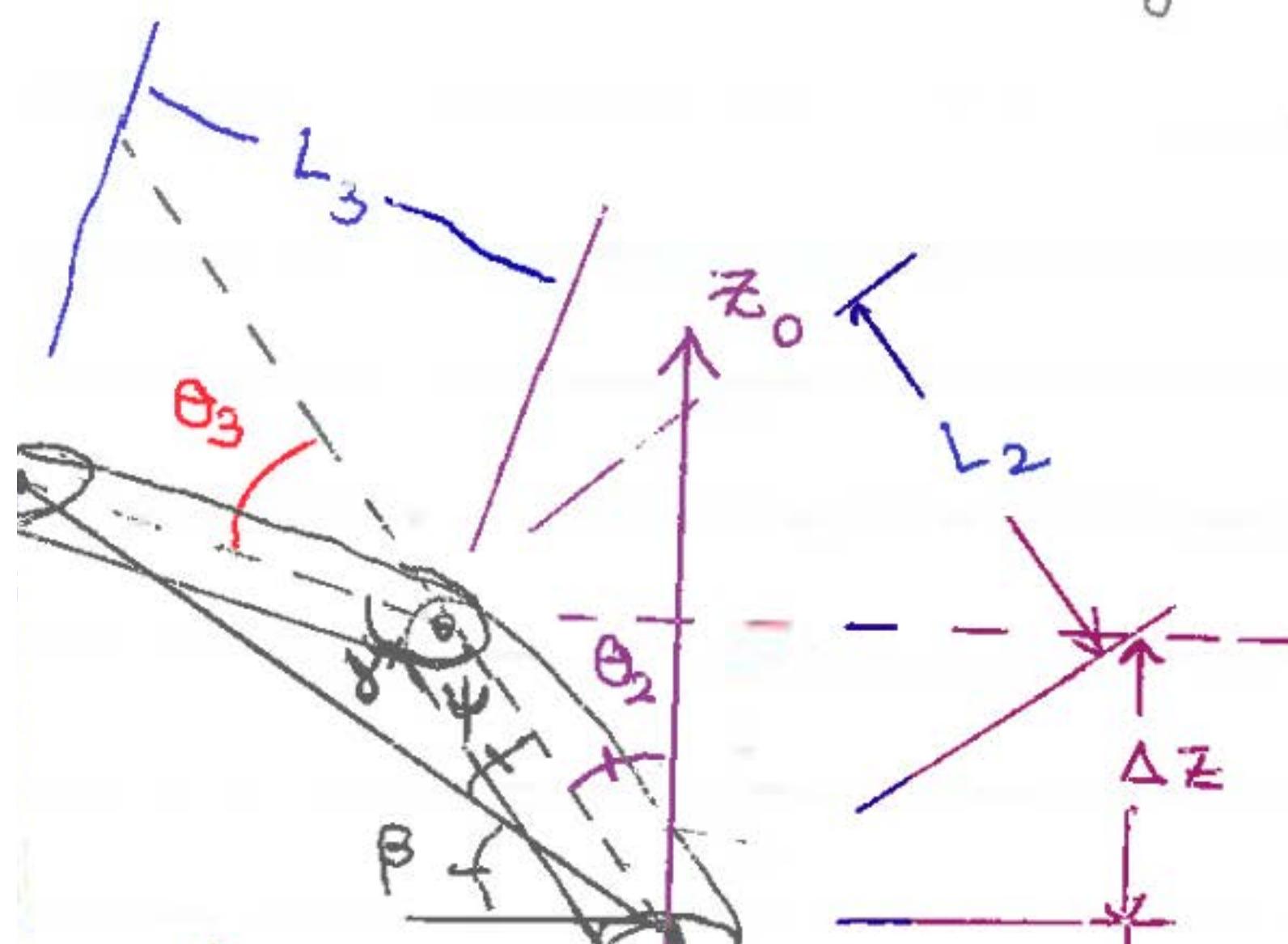


and joint 1 from rest
of robot

solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_3



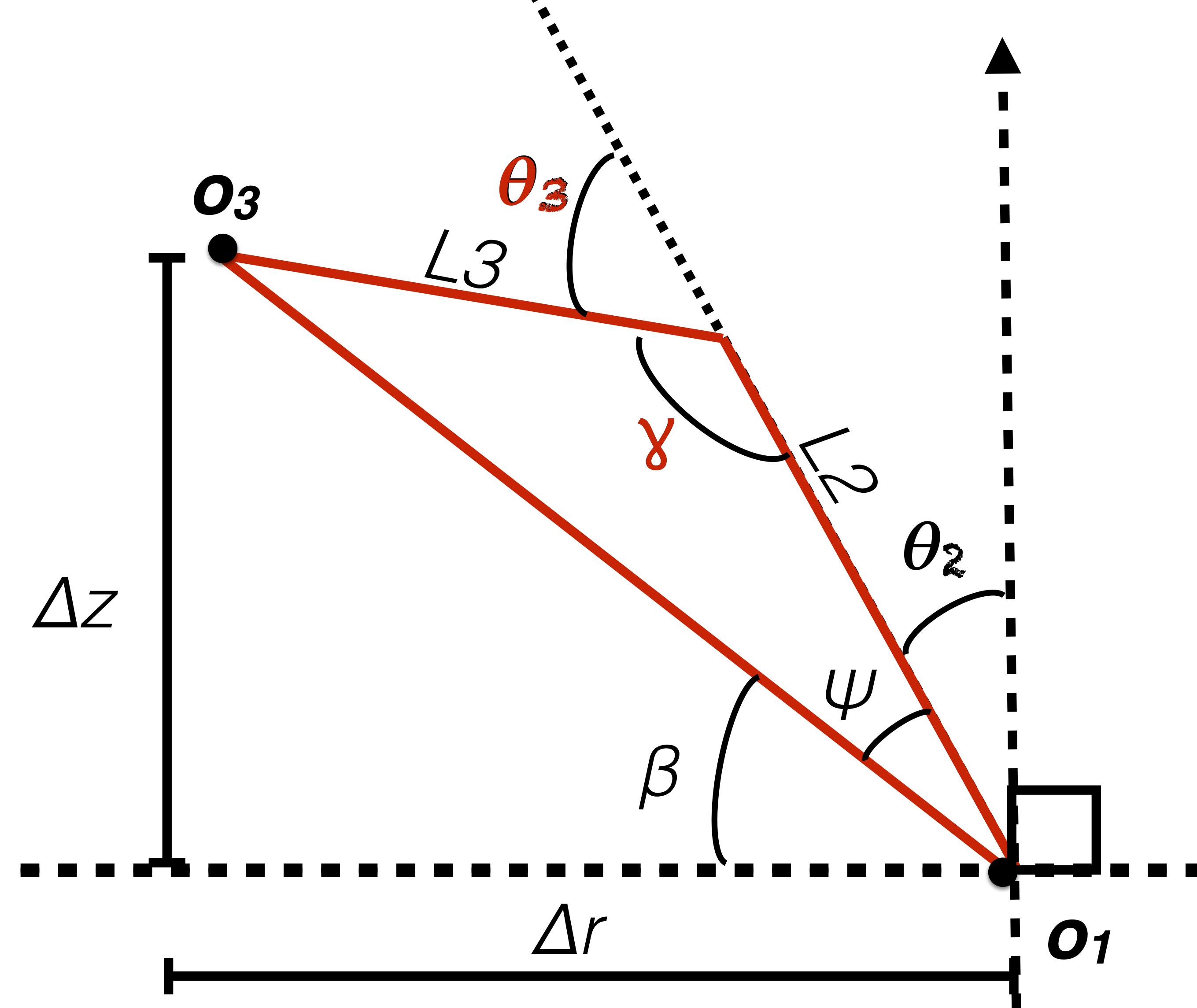
solve for θ_1

$$\theta_1 = \text{atan2}(\gamma_g, x_g)$$

solve for θ_3

(Law of cosines with supplementary angle γ)

$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$



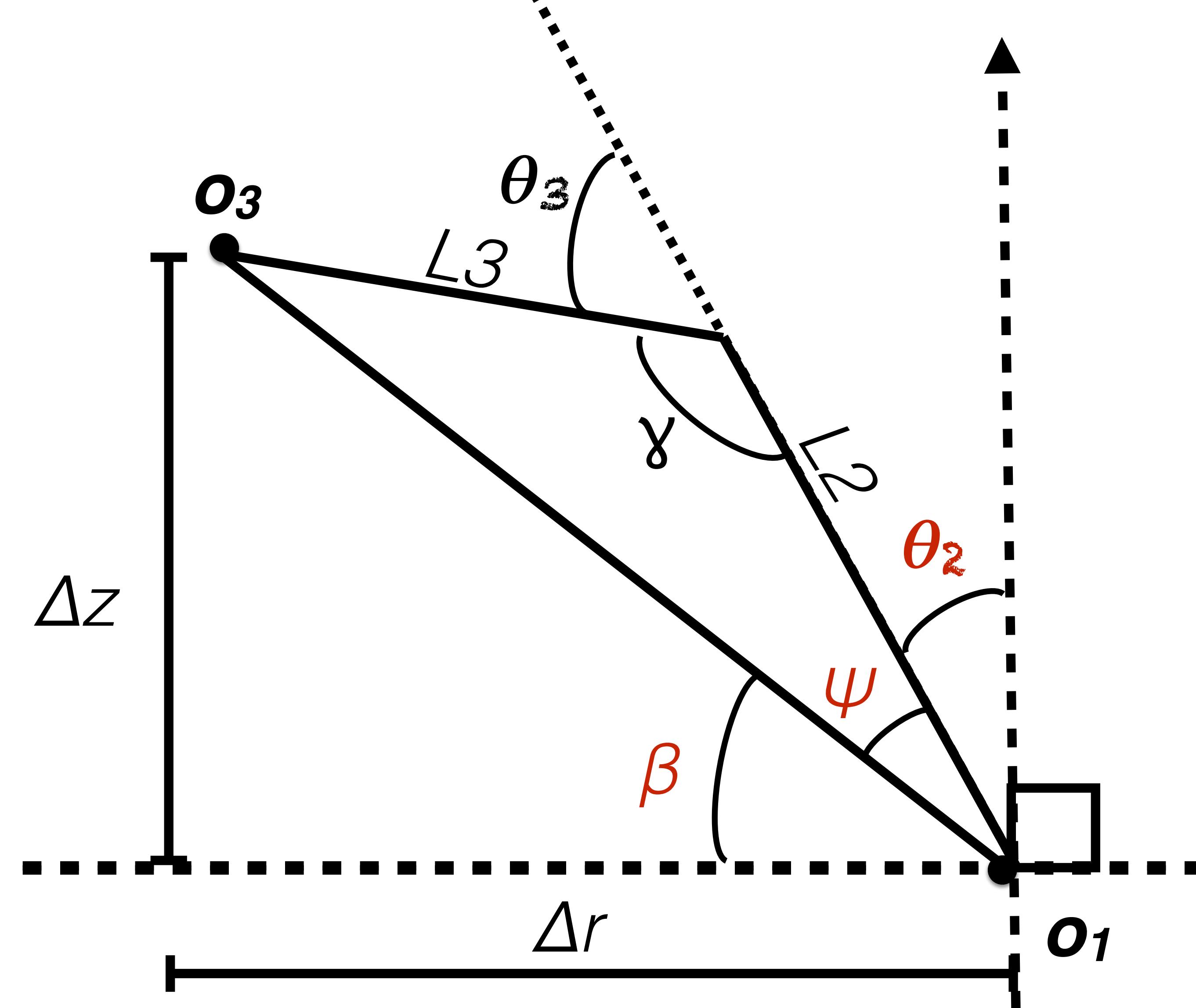
solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_3

$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2



solve for θ_1

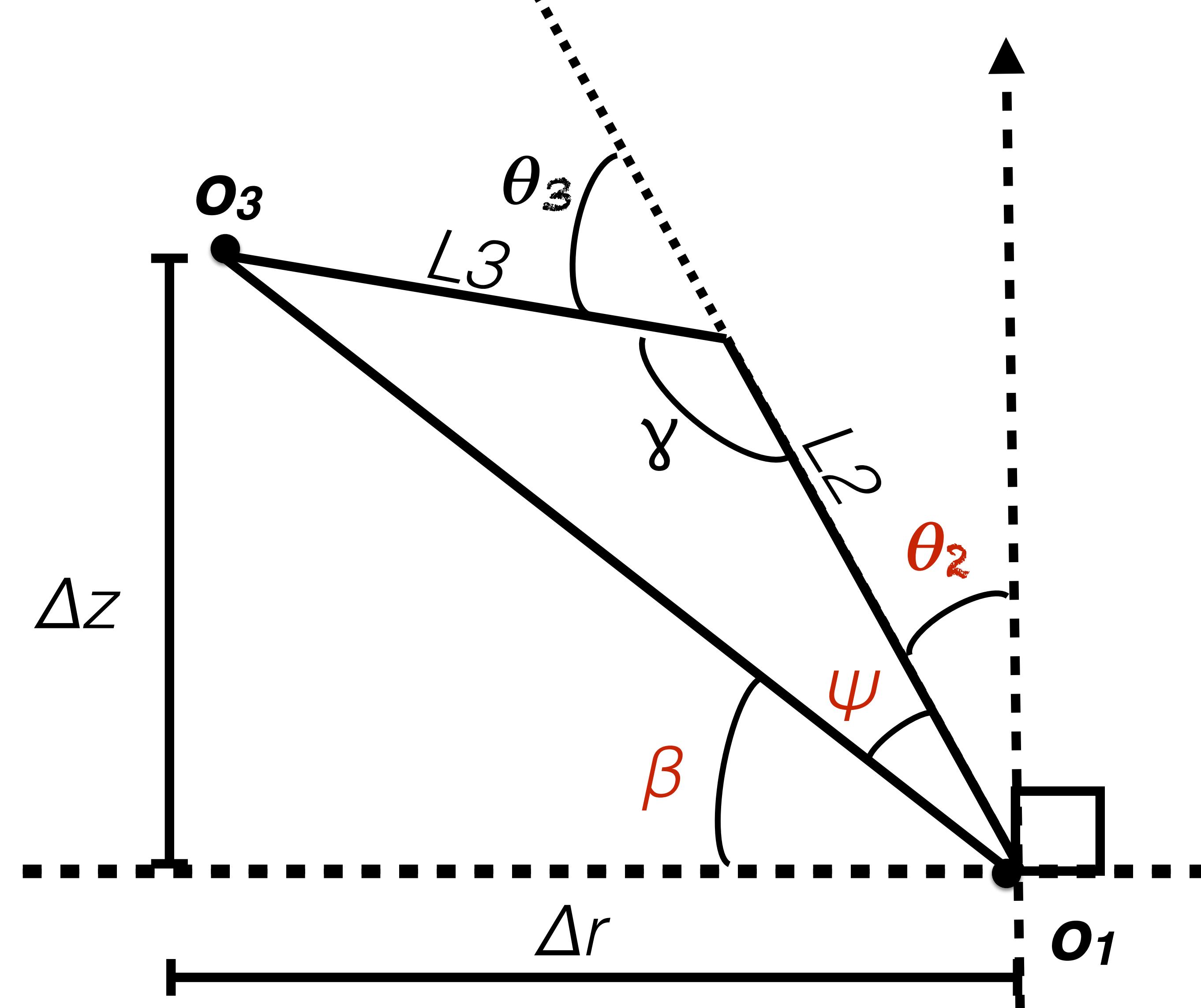
$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_3

$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2

(Law of cosines with angle ψ ,
arctan with angle β)



solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

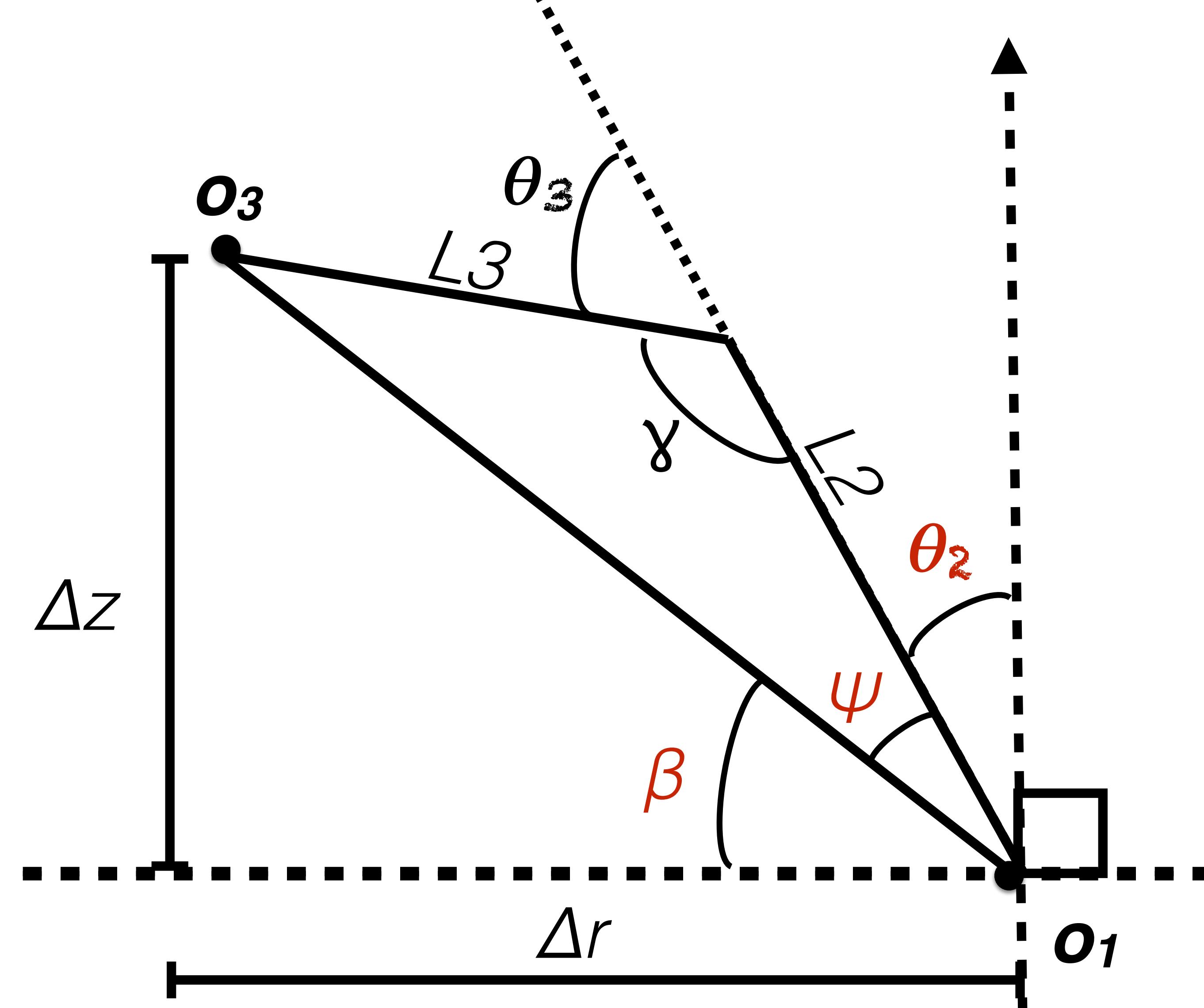
solve for θ_3

$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2

$$\theta_2 = \begin{cases} \frac{\pi}{2} - \beta - \psi & \text{if } \theta_3 \geq 0 \quad \text{"Elbow up"} \\ \frac{\pi}{2} - \beta + \psi & \text{if } \theta_3 < 0 \quad \text{"Elbow-down"} \end{cases}$$

two potential solutions
depending on elbow angle



solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

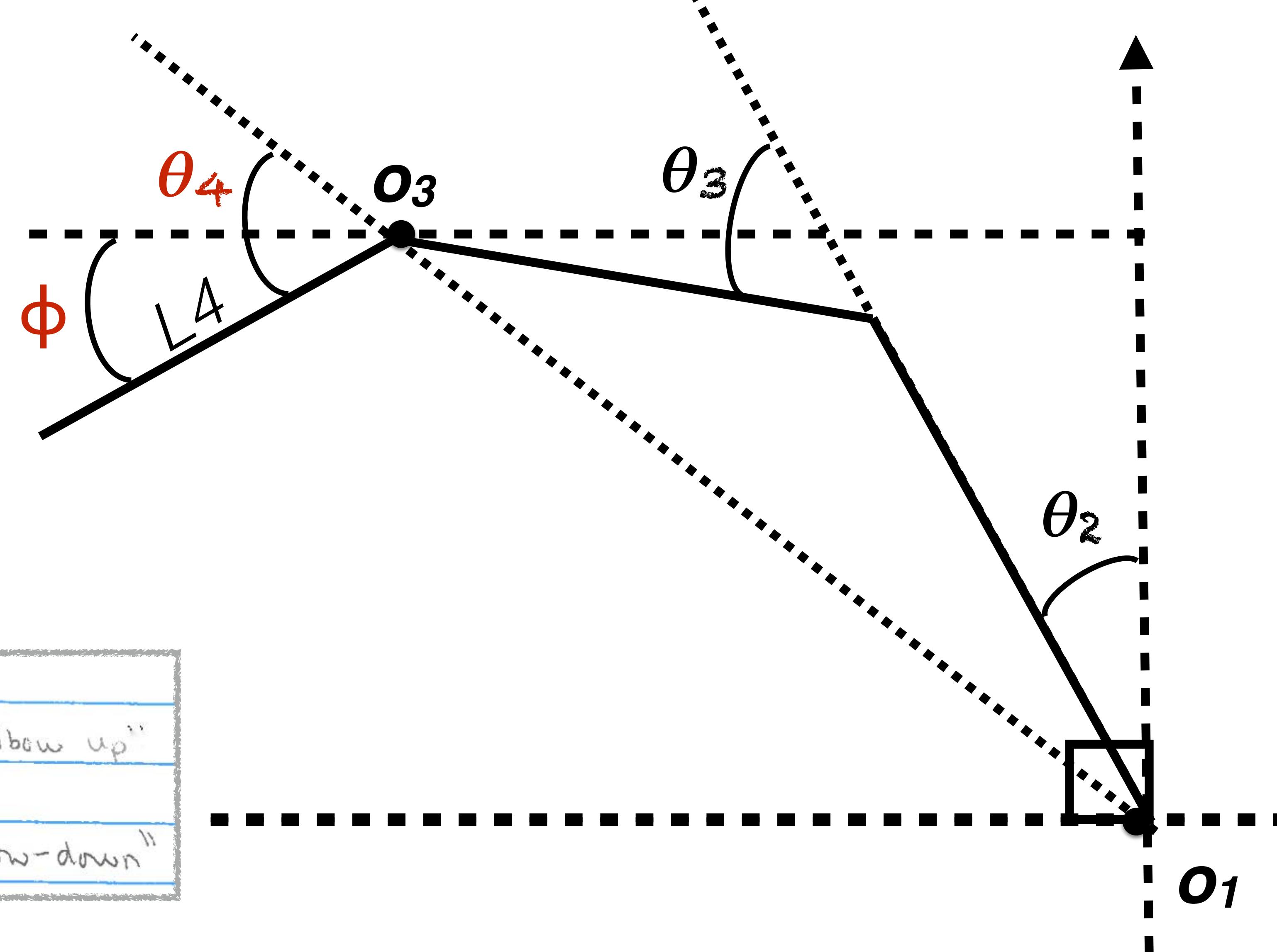
solve for θ_3

$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2

$$\theta_2 = \begin{cases} \frac{\pi}{2} - \beta - \psi & \text{if } \theta_3 \geq 0 \quad \text{"Elbow up"} \\ \frac{\pi}{2} - \beta + \psi & \text{if } \theta_3 < 0 \quad \text{"Elbow-down"} \end{cases}$$

solve for θ_4



solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_3

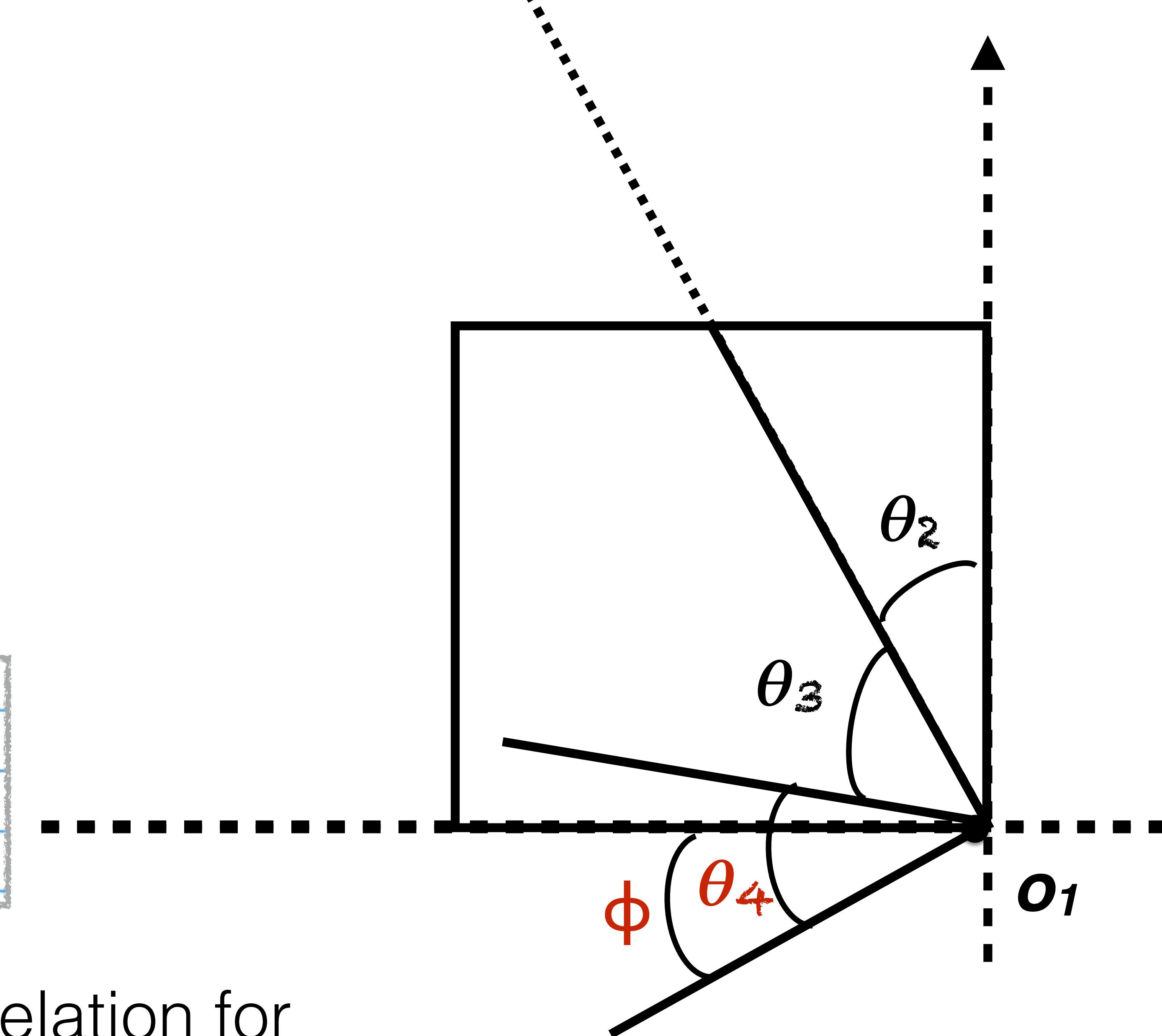
$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2

$$\theta_2 = \begin{cases} \frac{\pi}{2} - \beta - \psi & \text{if } \theta_3 \geq 0 \quad \text{"Elbow up"} \\ \frac{\pi}{2} - \beta + \psi & \text{if } \theta_3 < 0 \quad \text{"Elbow-down"} \end{cases}$$

solve for θ_4

(Equivalence relation for adding angles from \mathbf{z}_0)



solve for θ_1

$$\theta_1 = \text{atan2}(y_g, x_g)$$

solve for θ_3

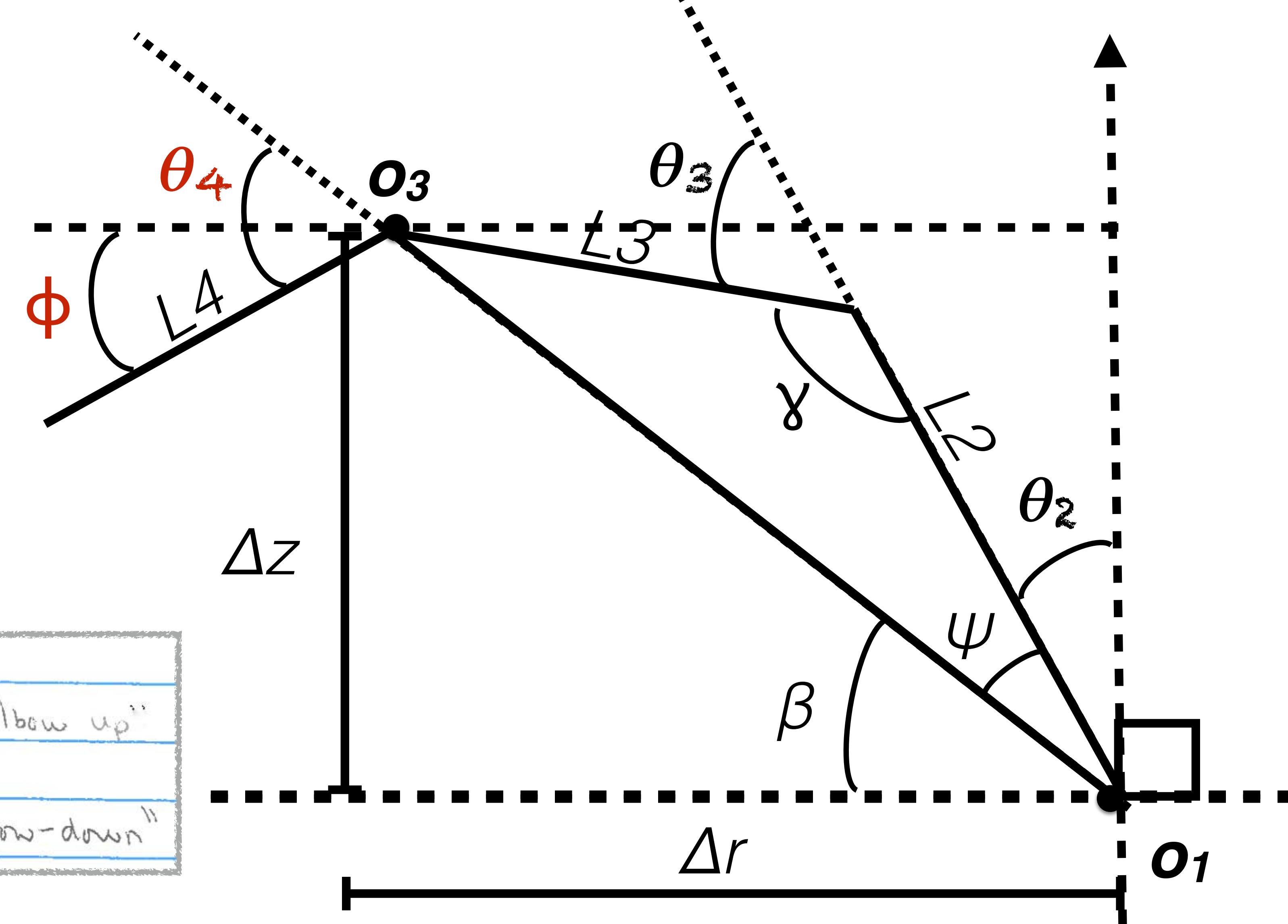
$$\cos \theta_3 = \frac{\Delta z^2 + \Delta r^2 - L_2^2 - L_3^2}{2 L_2 L_3}$$

solve for θ_2

$$\theta_2 = \begin{cases} \frac{\pi}{2} - \beta - \psi & \text{if } \theta_3 \geq 0 \quad \text{"Elbow up"} \\ \frac{\pi}{2} - \beta + \psi & \text{if } \theta_3 < 0 \quad \text{"Elbow-down"} \end{cases}$$

solve for θ_4

$$\theta_4 = \phi - \theta_2 - \theta_3 + \frac{\pi}{2}$$



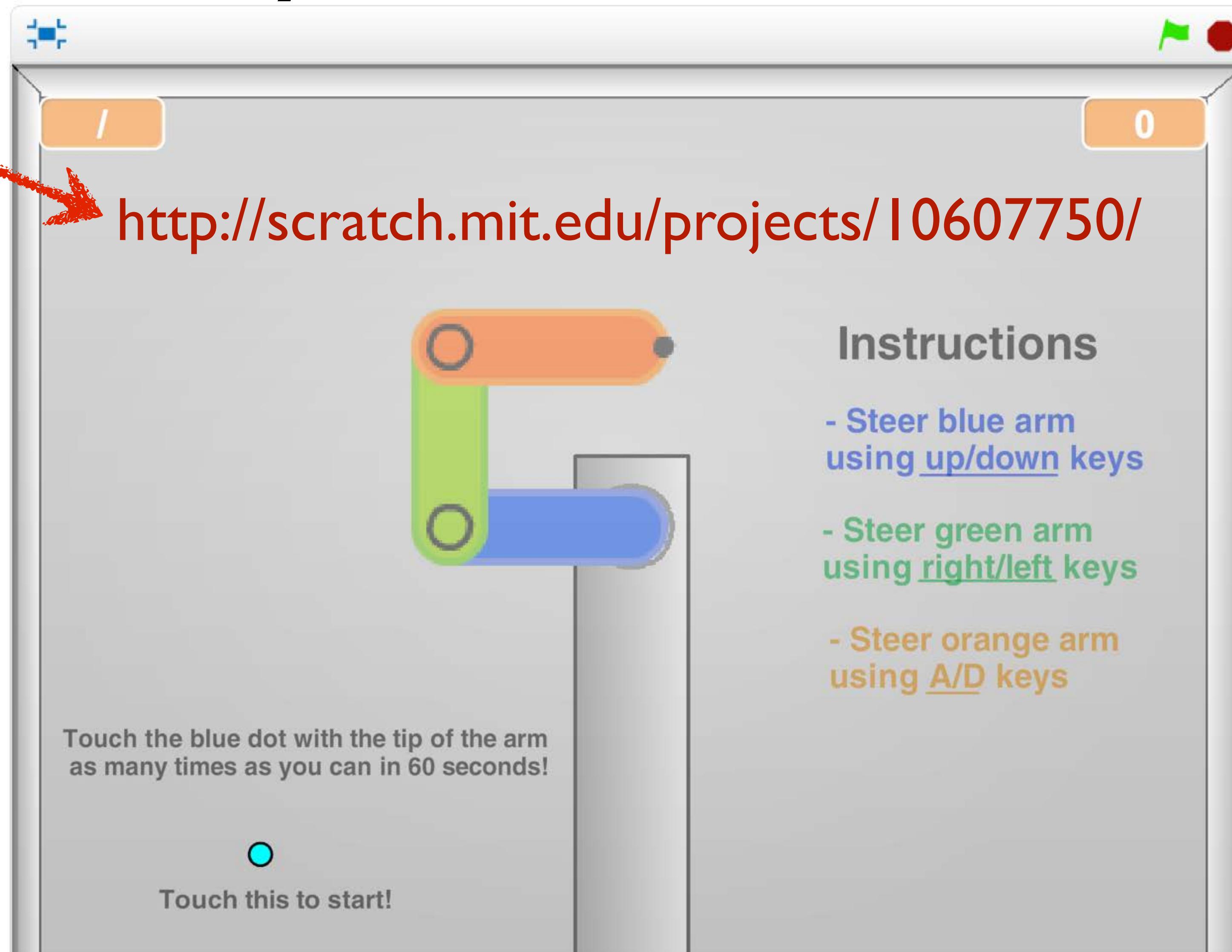
(Addition of angles in arm plane starting from z_0)

Inverse Kinematics: 2 possibilities

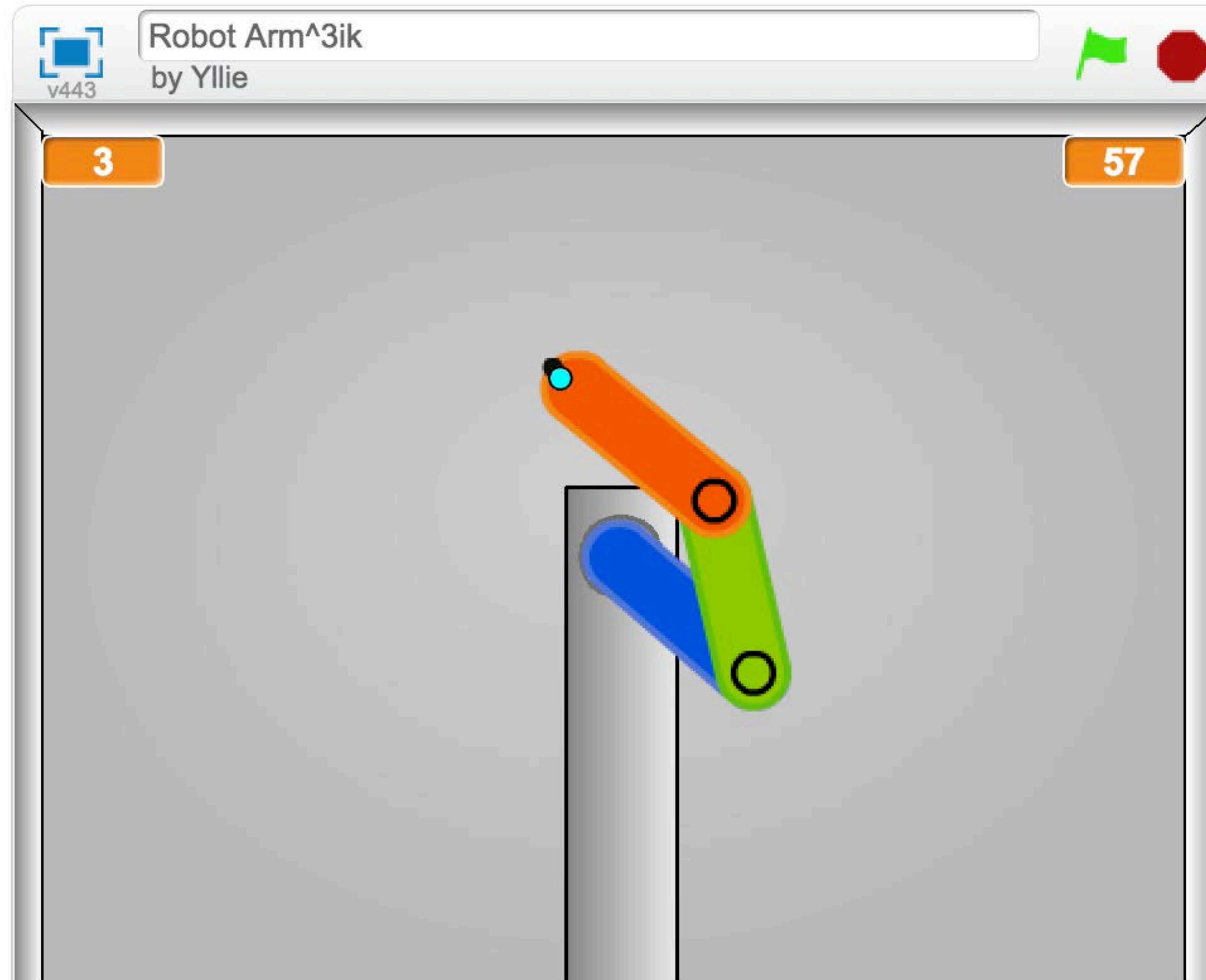
- **Closed-form solution:** geometrically infer satisfying configuration
 - *Speed:* solution often computed in constant time
 - *Predictability:* solution is selected in a consistent manner
- **Solve by optimization:** minimize error of endeffector to desired pose
 - often some form of Gradient Descent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

Anyone tried this?

Try this



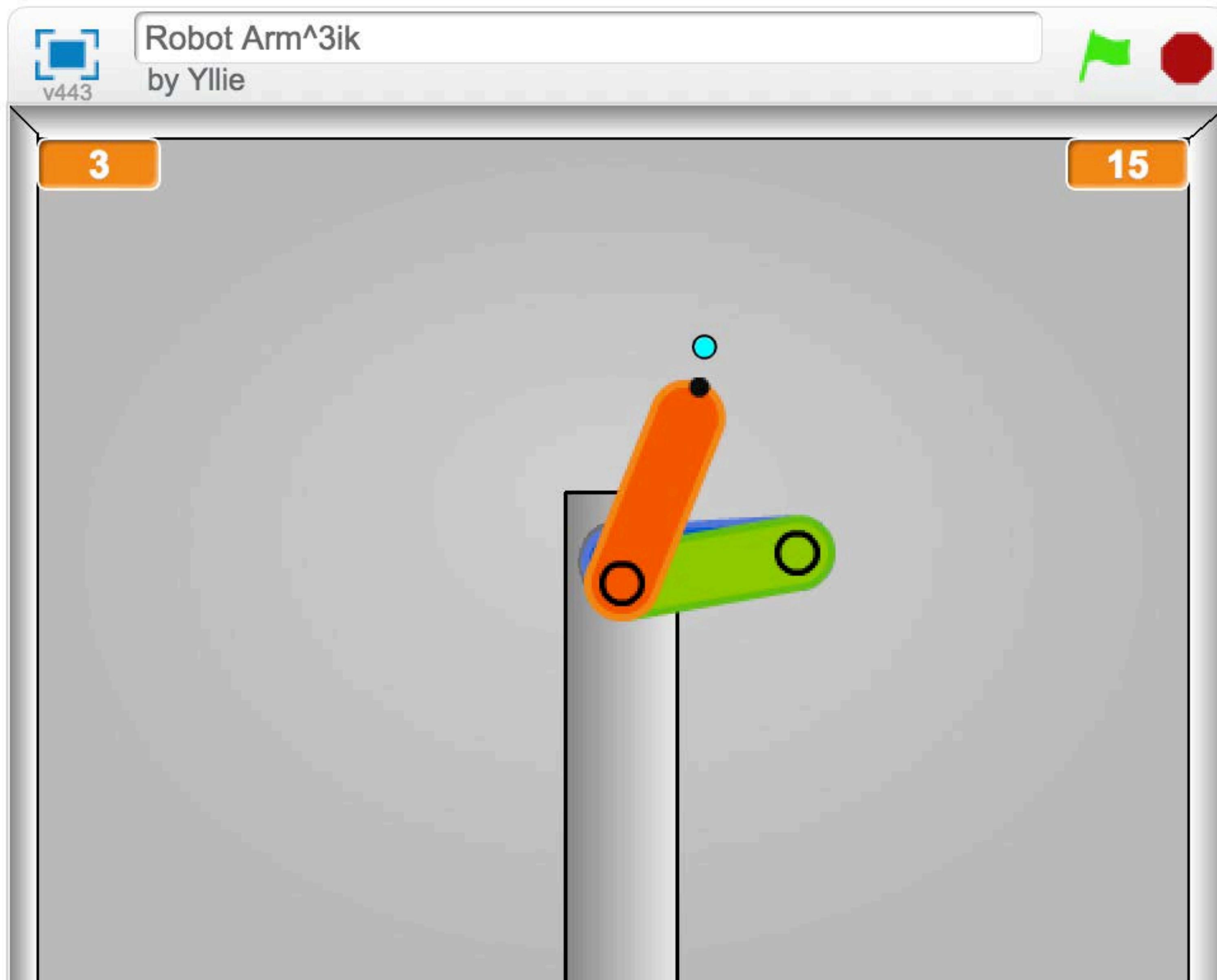
Aggressively tuned IK



By Dr. Jenkins



Conservatively tuned IK



By Dr. Jenkins



Isn't IK supposed to give one answer?

How is the arm moving here?

Here IK is implemented as an iterative method.
Here is it visualized with all the intermediate solution.

How to programmatically do this?

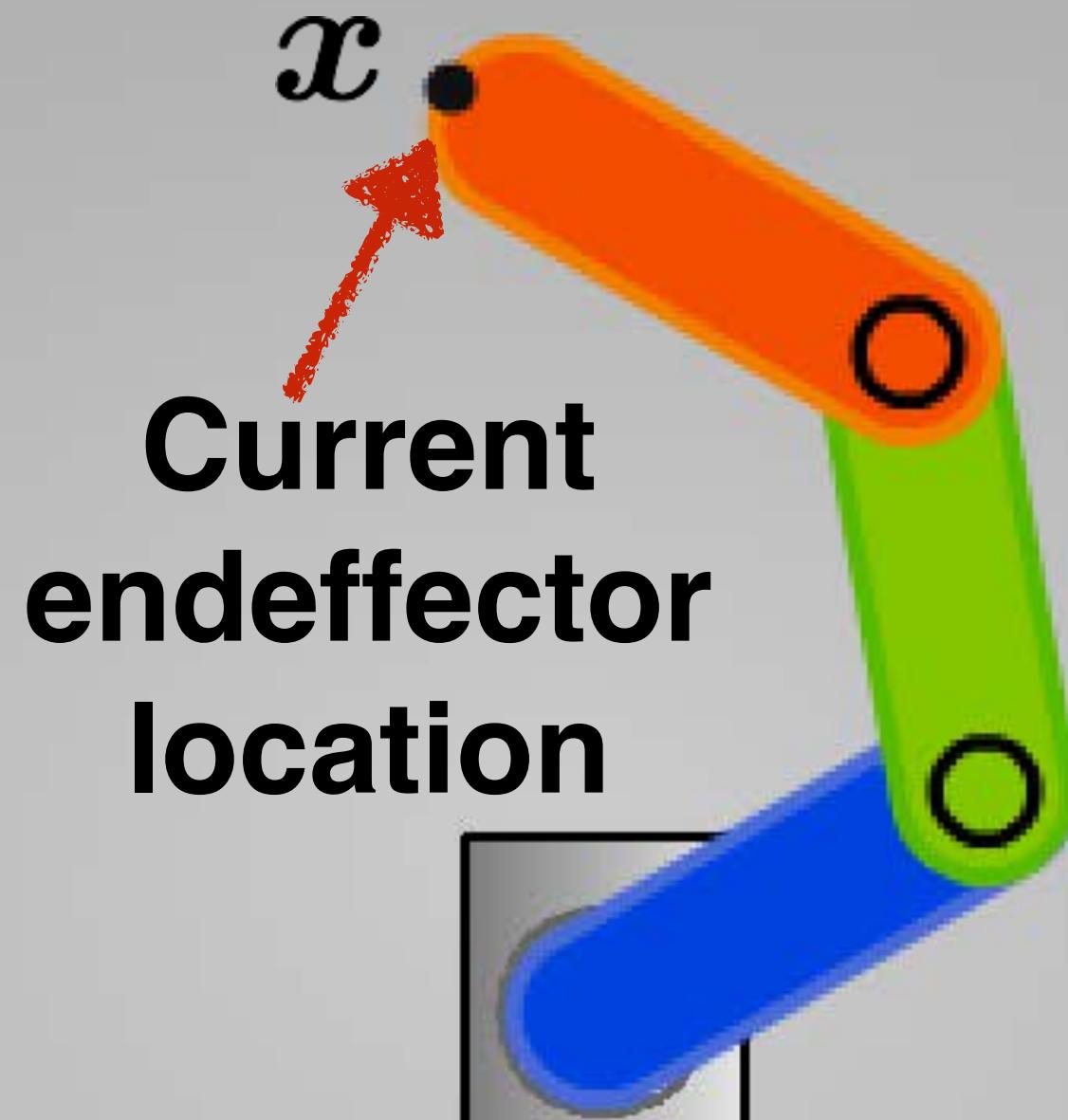
How to programmatically do
this?

Jacobian Transpose



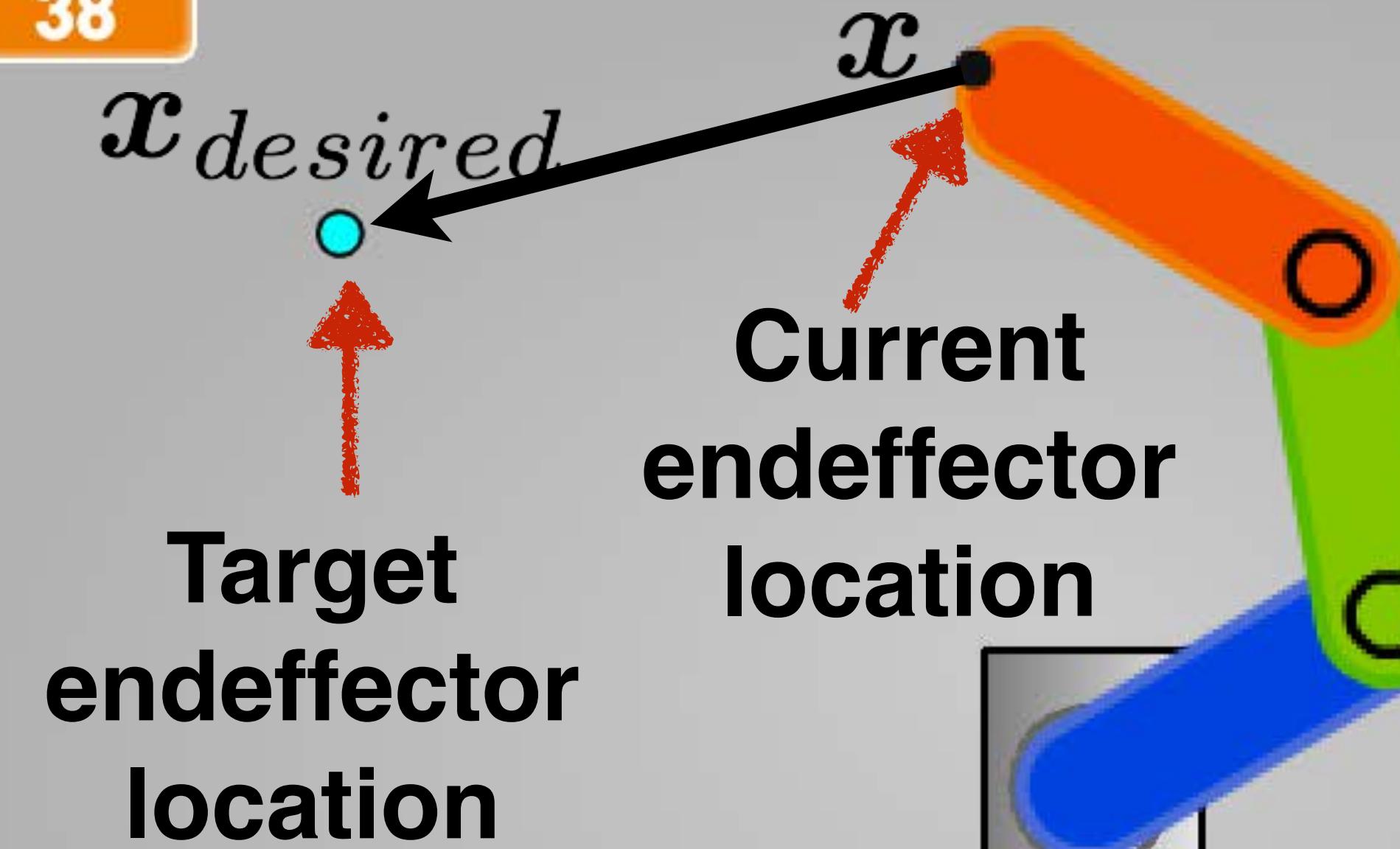
$x_{desired}$ 

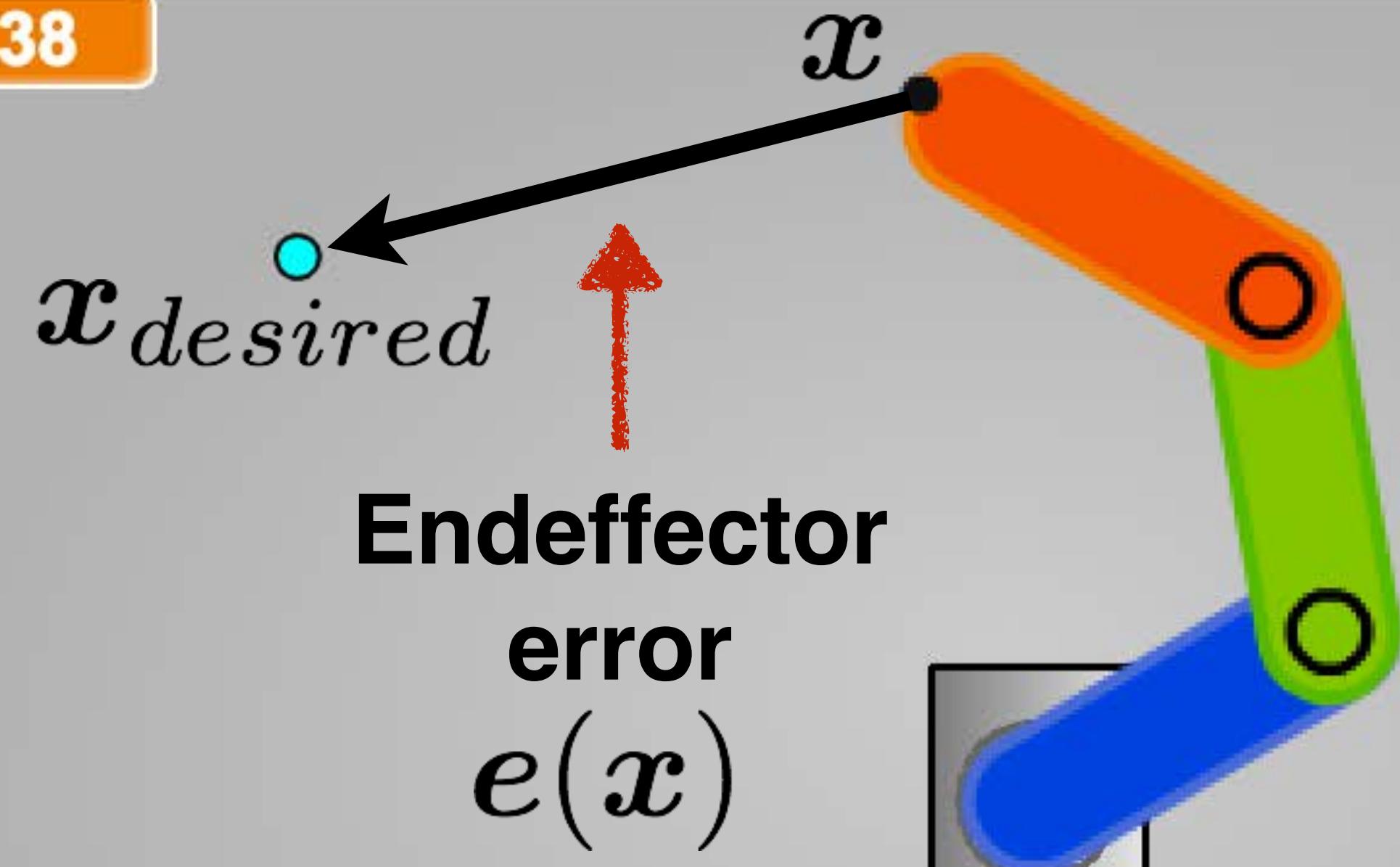
Target
endeffector
location



38

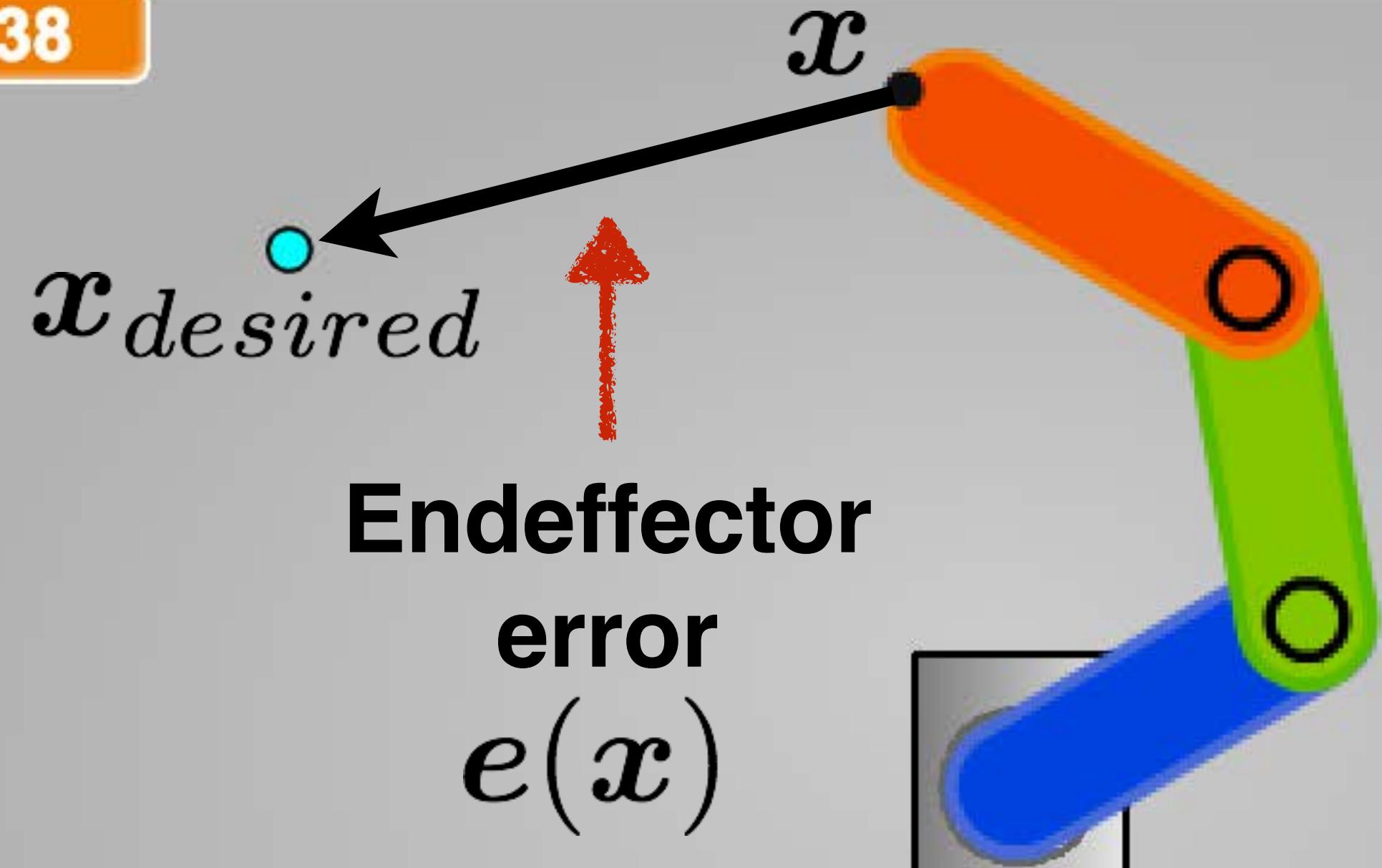
2





**Endeffector
error**
 $e(x)$

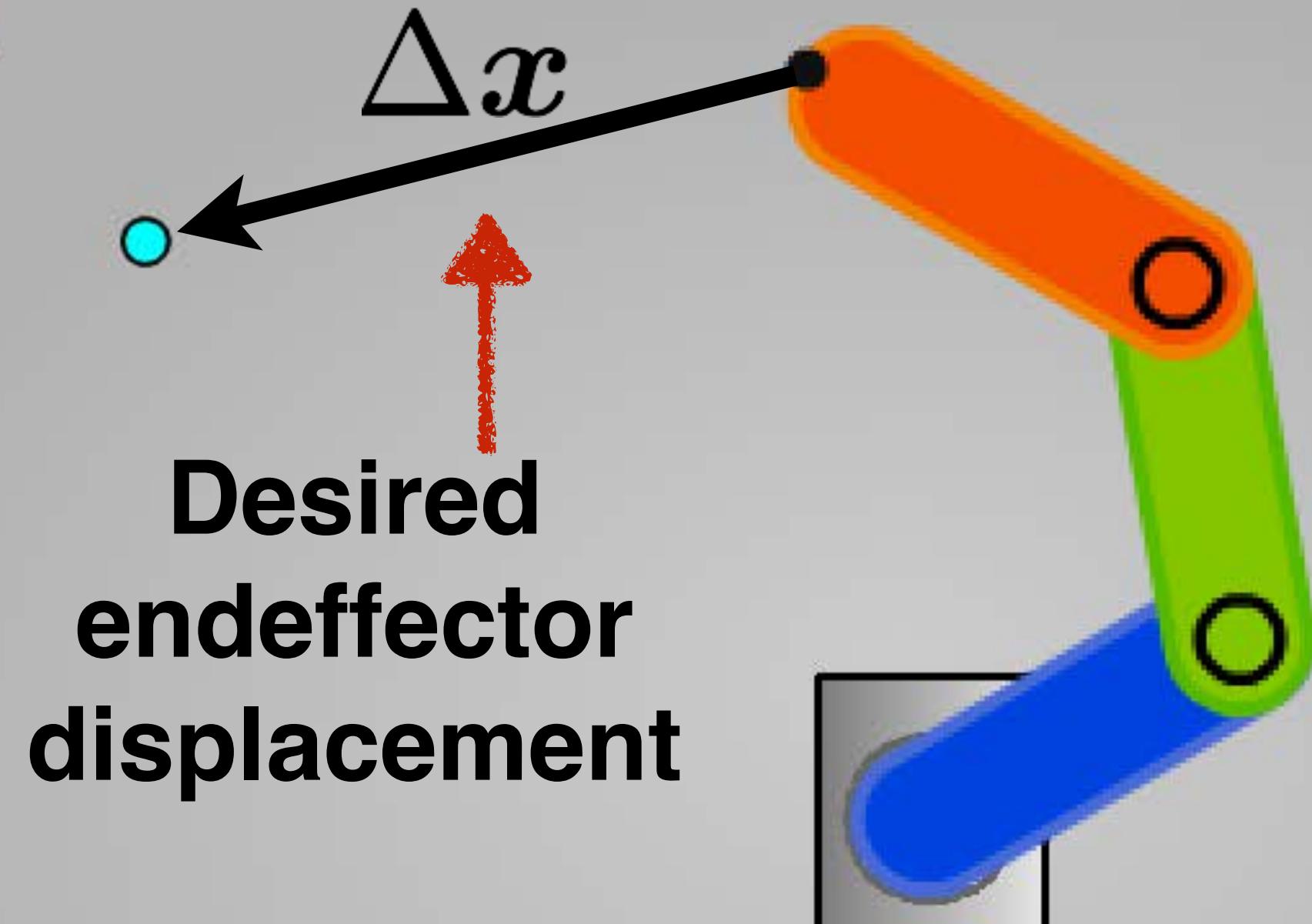
Can we move the
endeffector to
minimize error?



Can we move the
endeffector to
minimize error?

Yes!

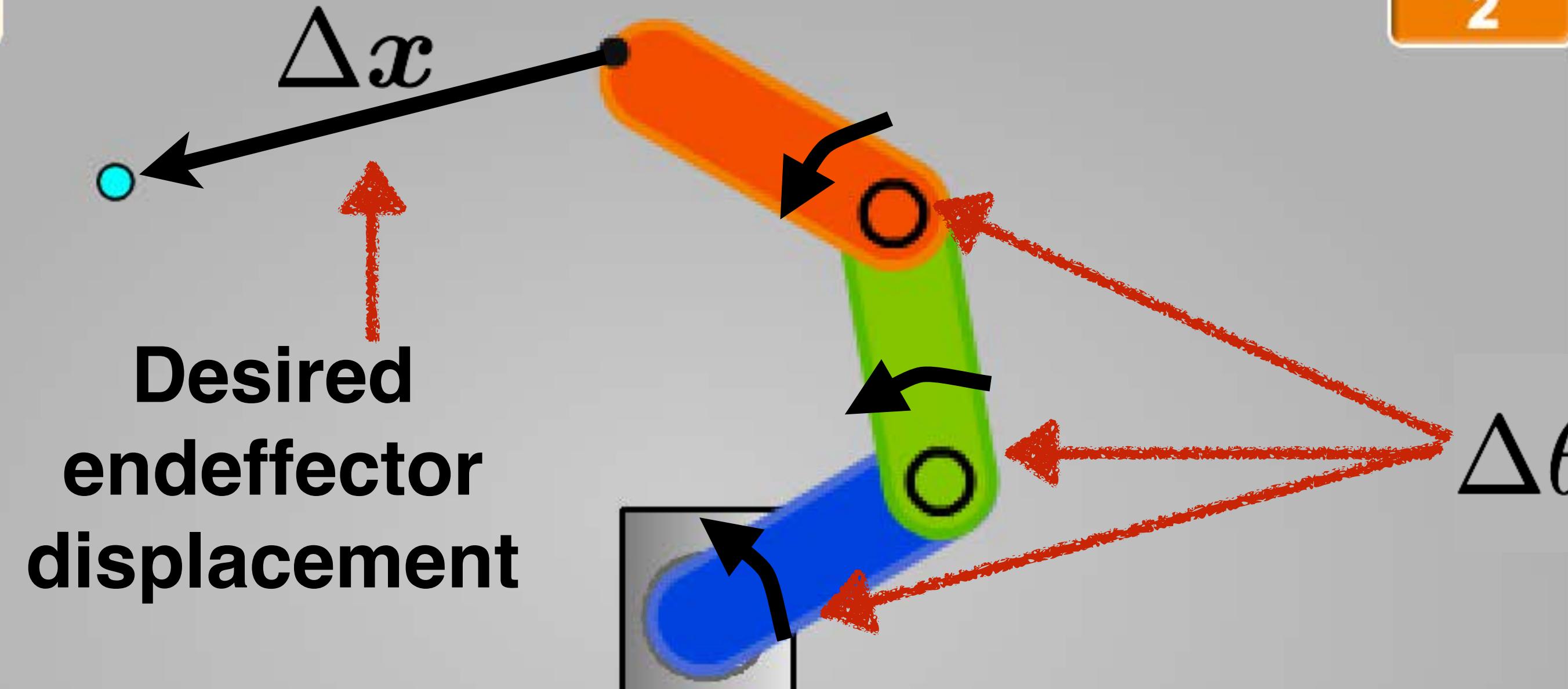
convert linear velocity at endeffector
to angular velocities at joints.



Can we move the
endeffector to
minimize error?

Yes!

convert linear velocity at endeffector
to angular velocities at joints.



Desired
endeffector
displacement

Joint angle
displacements

Can we move the
endeffector to
minimize error?

Yes!

convert linear velocity at endeffector
to angular velocities at joints.

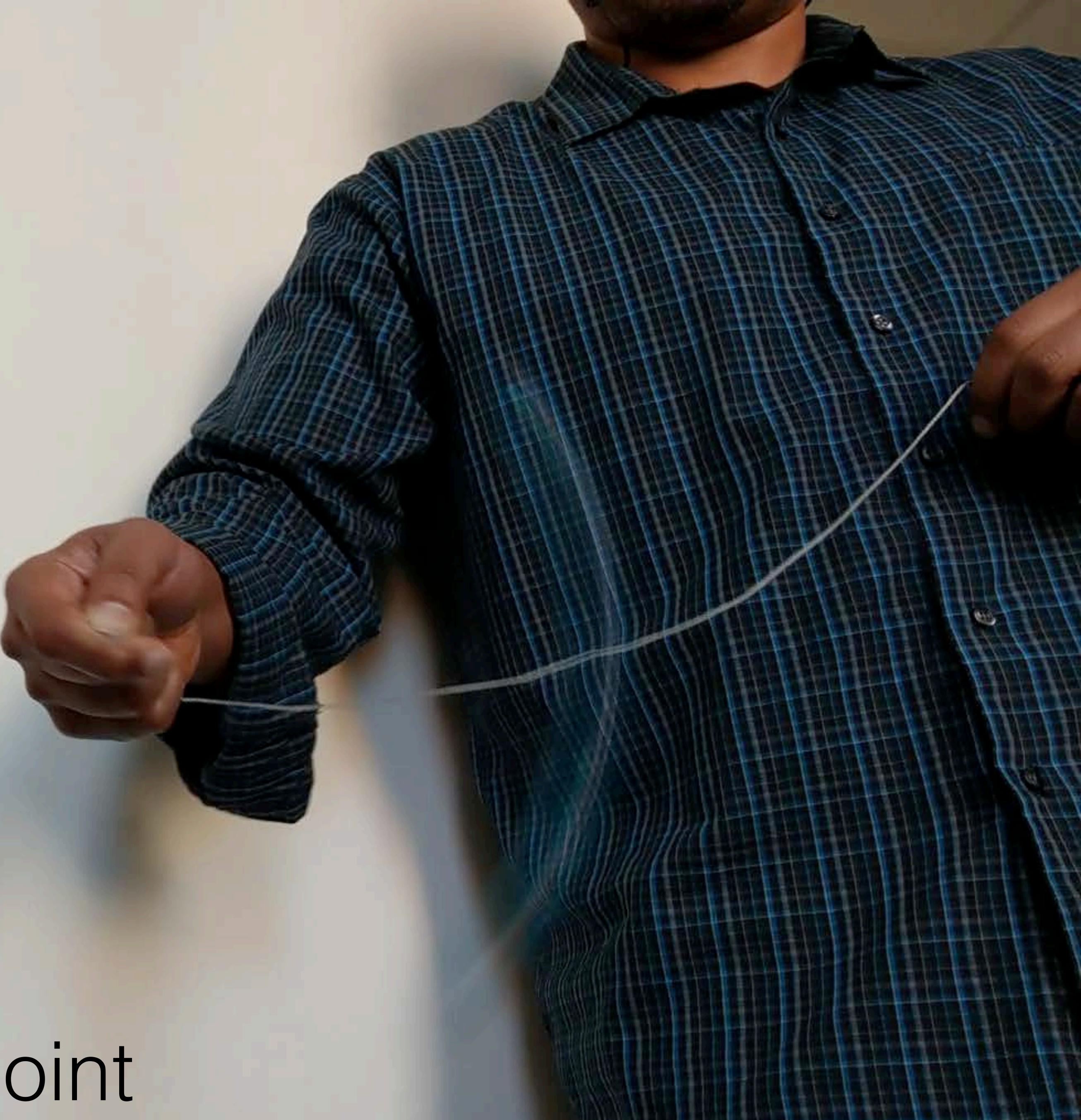
How are linear and angular velocity related?

How are linear and angular velocity related?

Consider the velocity of a point



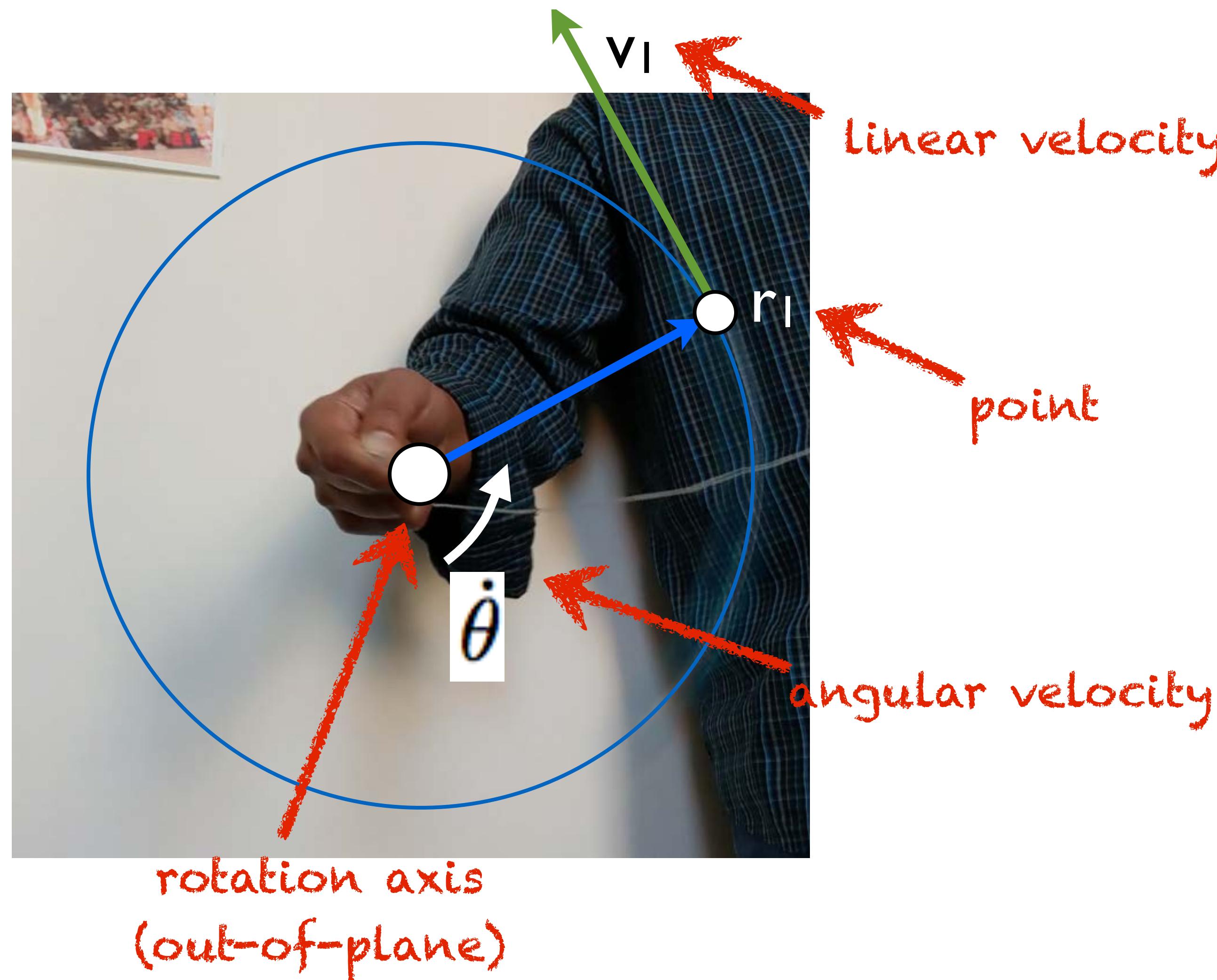
Consider the velocity of a point



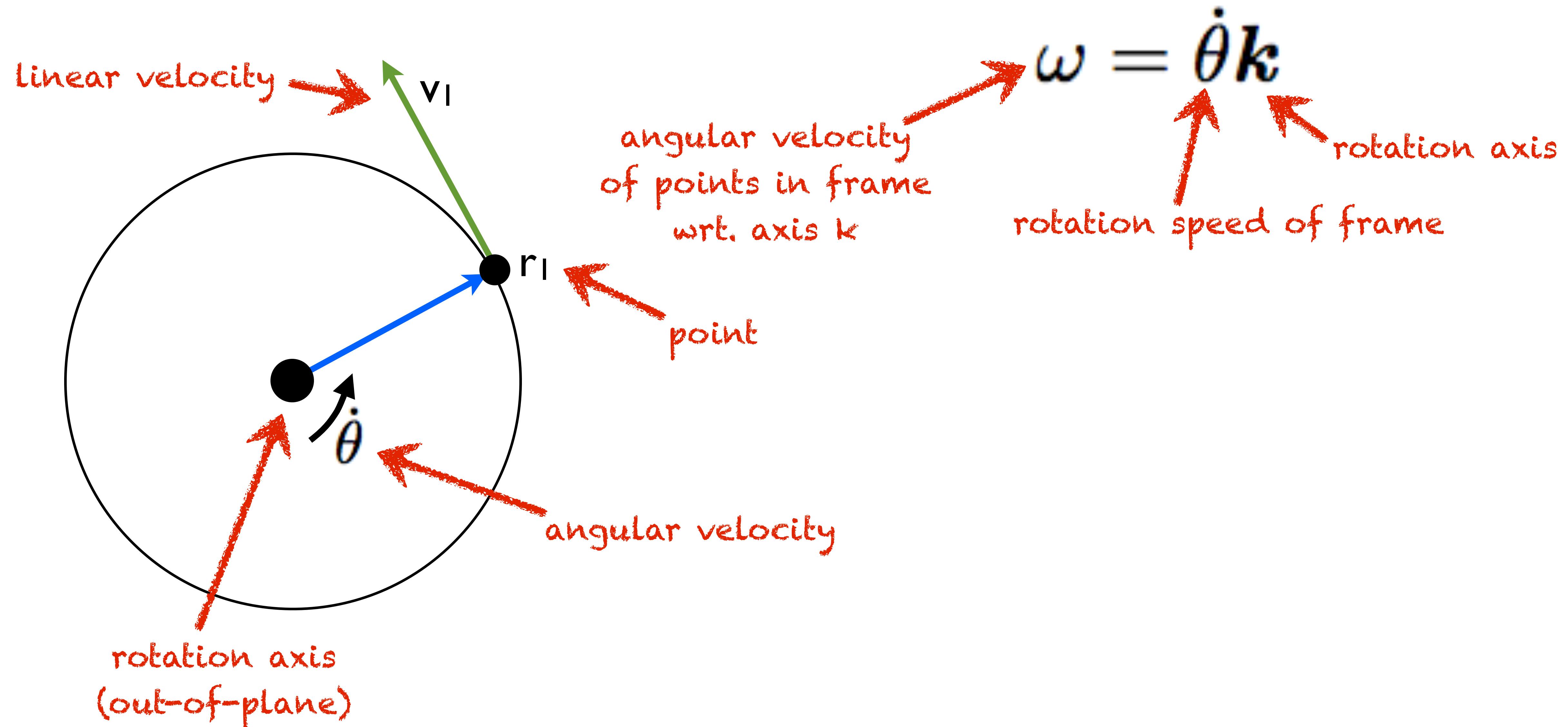
Consider the velocity of a point



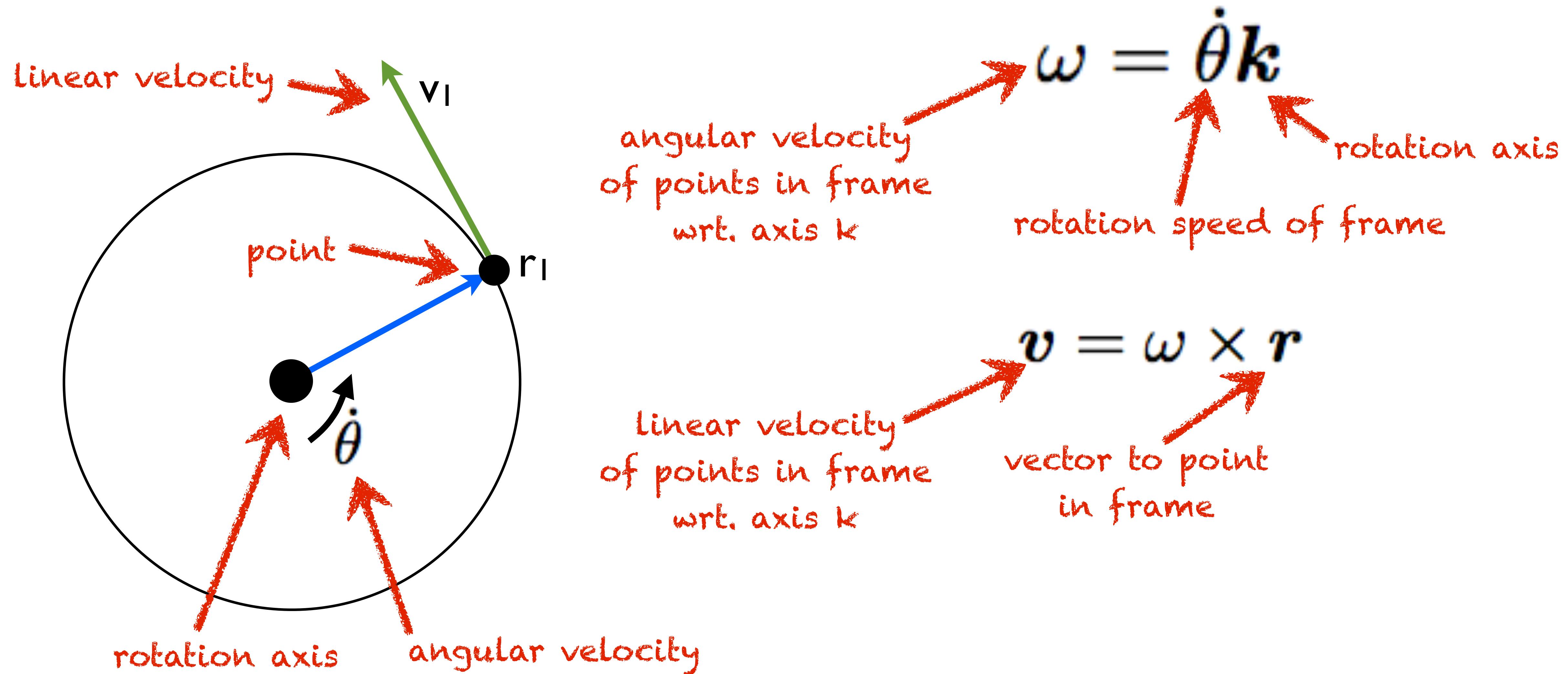
Velocity of Point Rotating in Fixed Frame



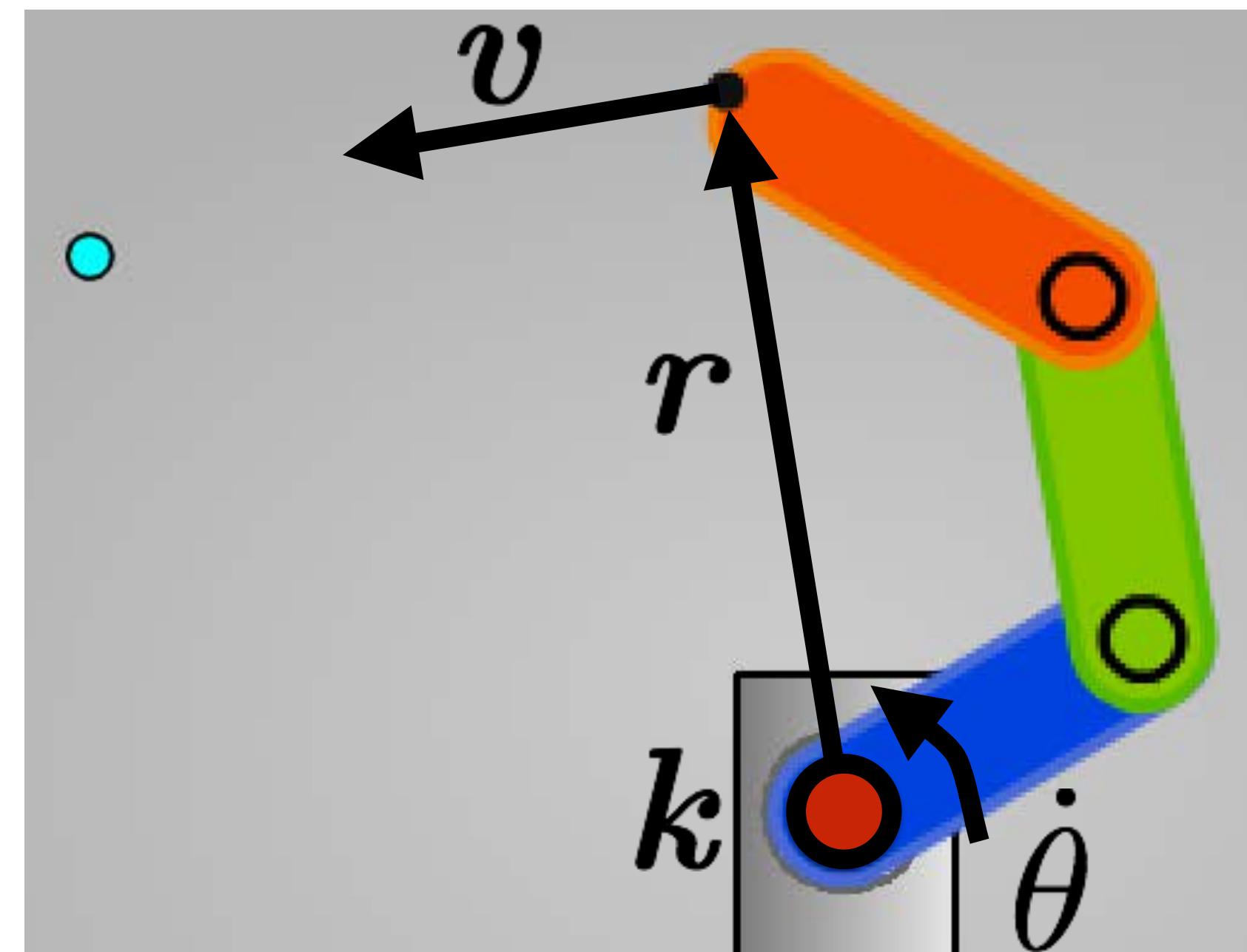
Velocity of Point Rotating in Fixed Frame



Velocity of Point Rotating in Fixed Frame



Velocity of Point Rotating in Fixed Frame



angular velocity
of points in frame
wrt. axis k

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

rotation speed of frame

Linear velocity
of points in frame
wrt. axis k

$$v = \omega \times r$$

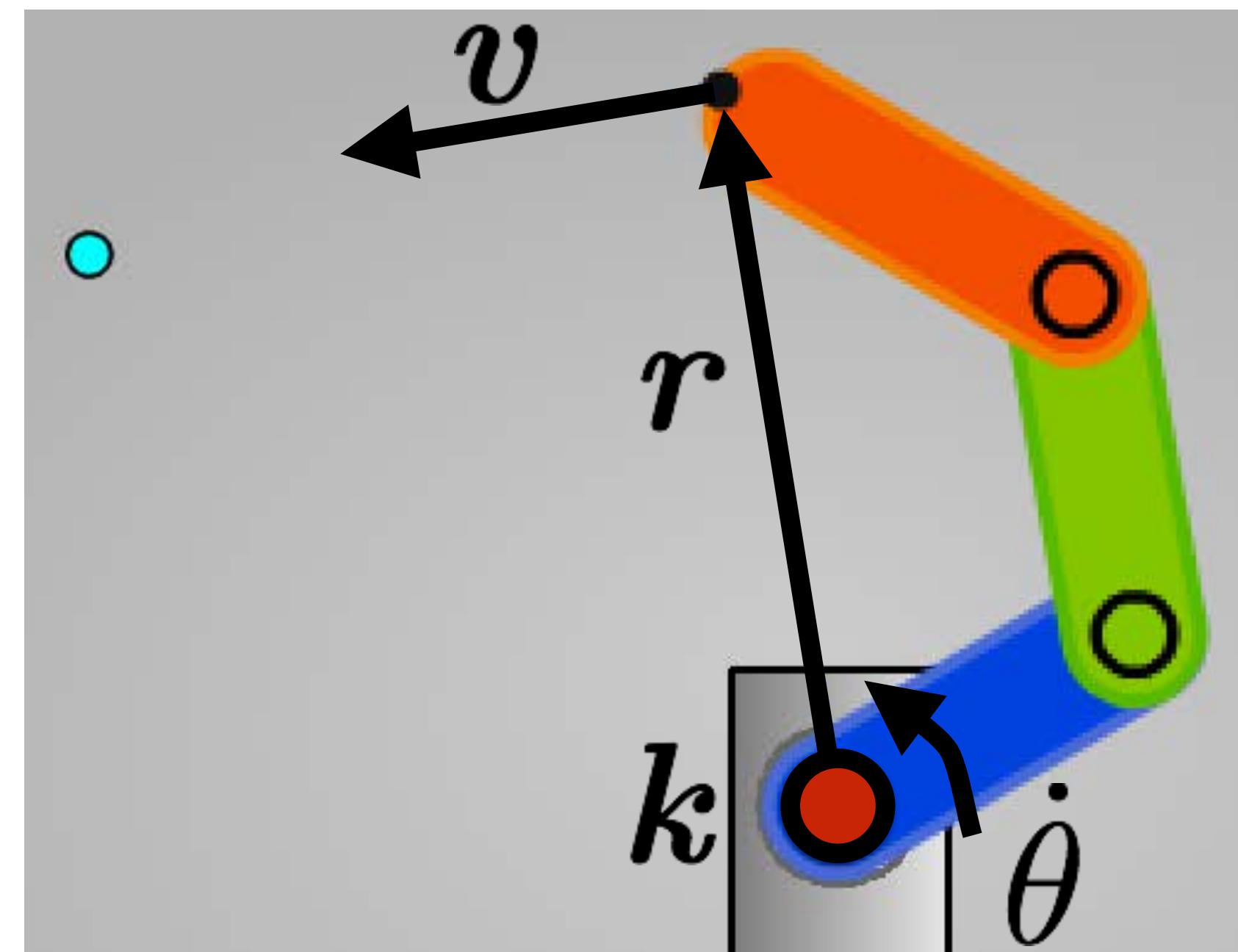
vector to point
in frame

endeffector
linear velocity

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

vector from
joint origin to
endeffector

joint rotation axis



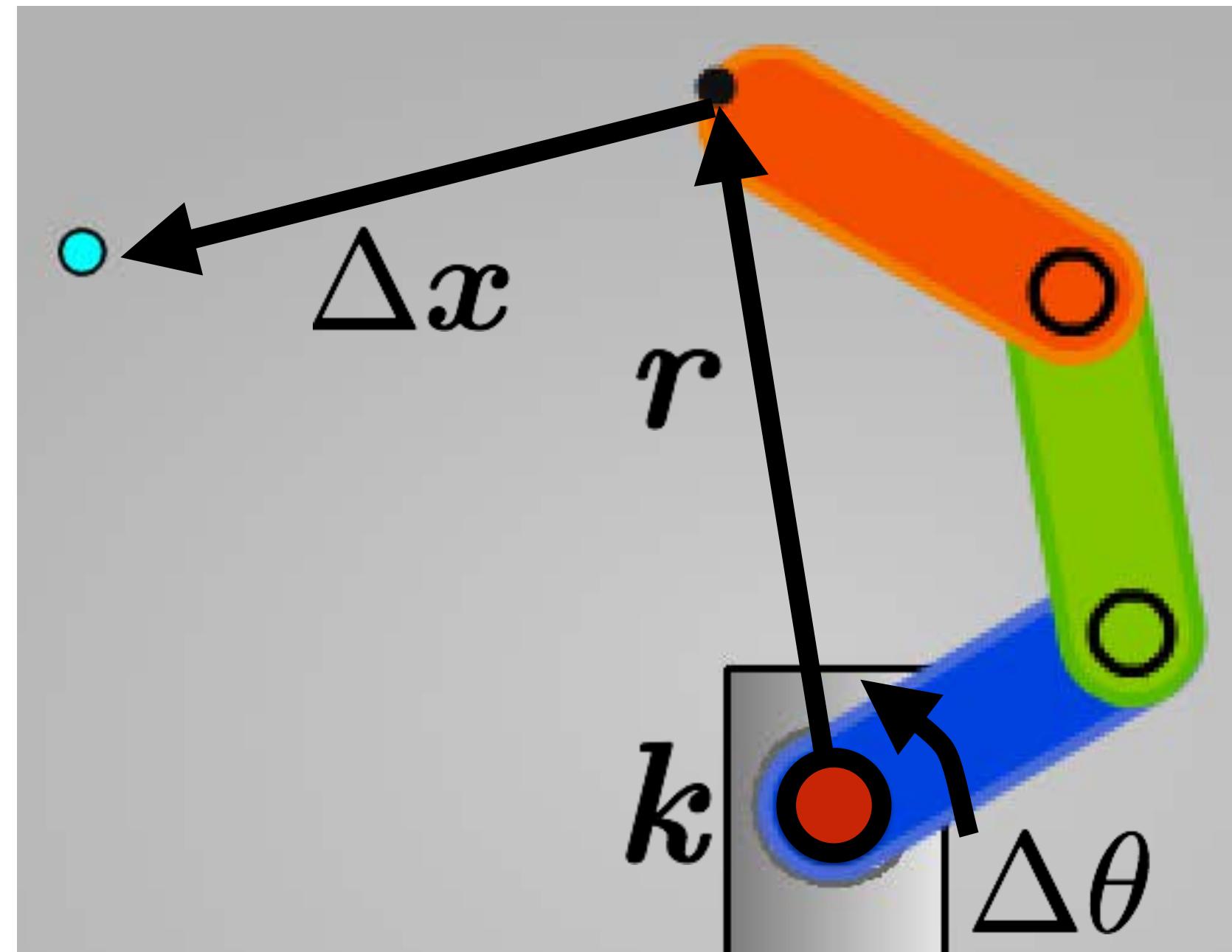
$$\vec{v} = \dot{\theta} k \times r$$

vector from joint origin to endeffector
endeffector Linear velocity
joint rotation axis

This is not what we wanted.

Why?

Jacobian Transpose



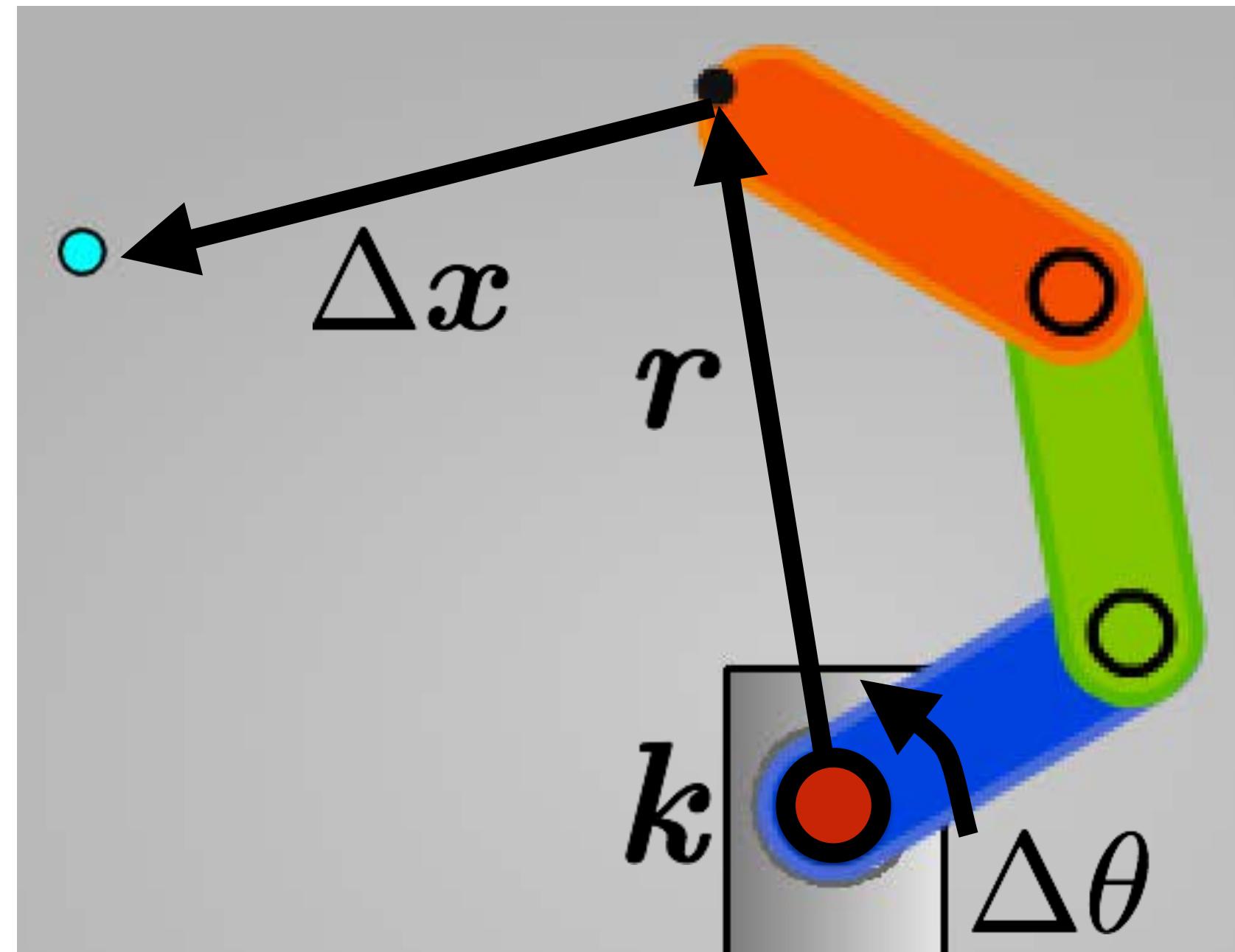
$$v = \dot{\theta}k \times r$$

vector from joint origin to endeffector
joint rotation axis
endeffector Linear velocity

This is not what we wanted.

How to obtain joint angular velocity from endeffector Linear velocity?

Jacobian Transpose



endeffector
Linear velocity

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

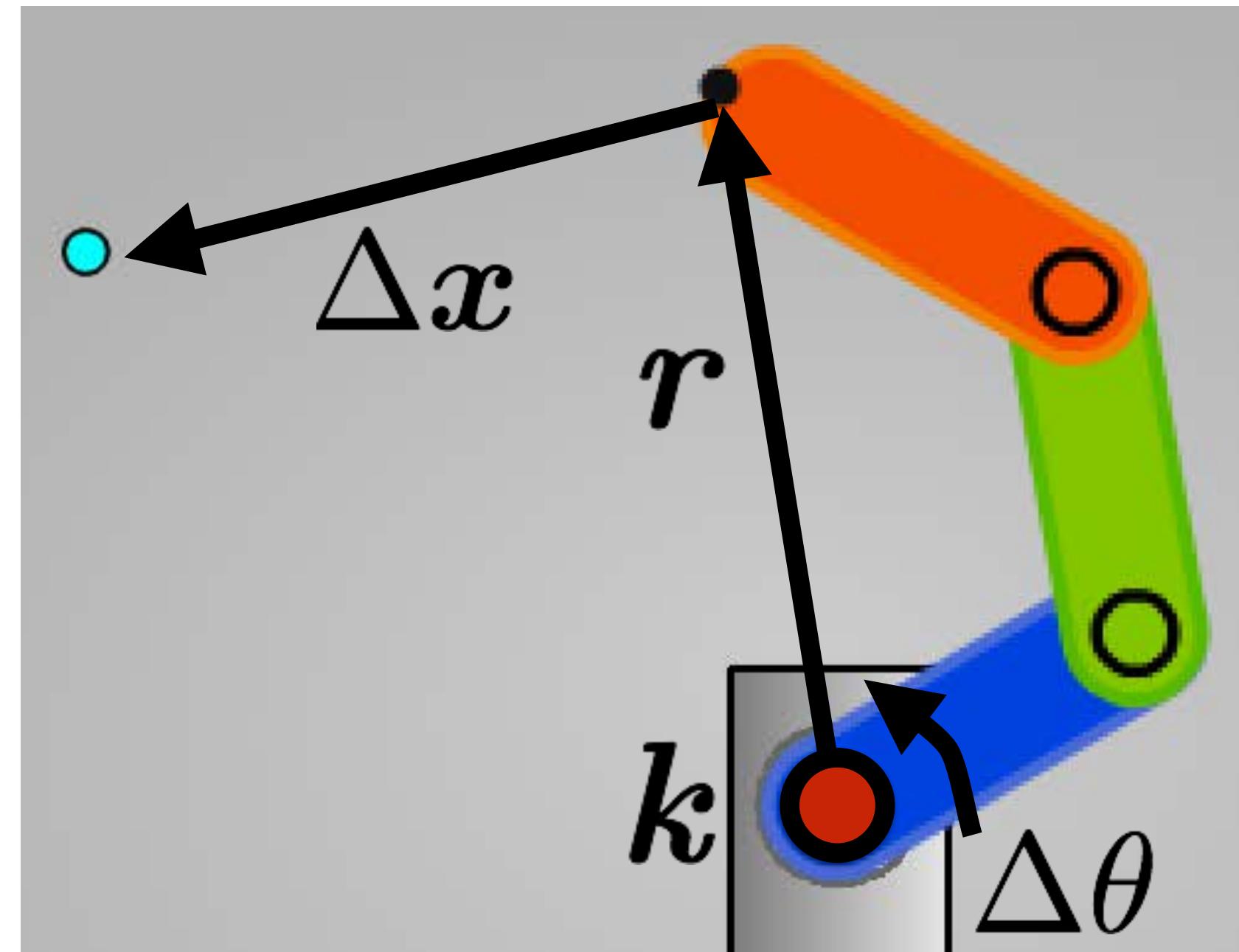
vector from joint origin to endeffector
joint rotation axis

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



$$\Delta\theta = \frac{(\underline{k} \times \underline{r})^T \Delta x}{\text{desired endeffector displacement}}$$

Annotations for the equation:

- joint rotation axis (points to the $\Delta\theta$ term)
- vector from joint origin to endeffector (points to the $\underline{k} \times \underline{r}$ term)
- desired endeffector displacement (points to the Δx term)
- Jacobian for joint i (points to the $(\underline{k} \times \underline{r})^T$ term)
- Angular displacement for joint i (points to the $\Delta\theta$ term)

Procedure (for each joint):

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

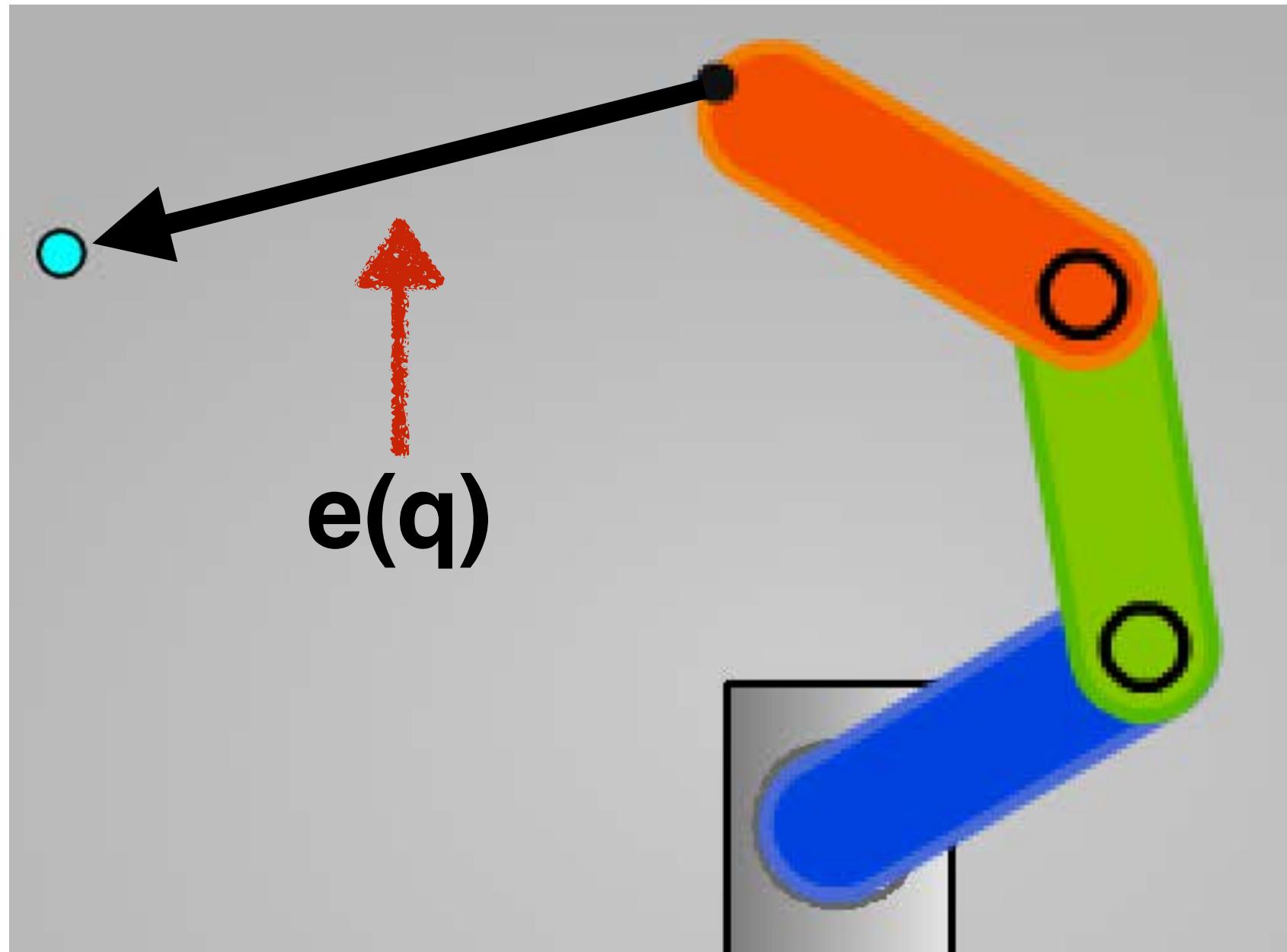
IK as Error Minimization

IK as Error Minimization

Gradient Descent Optimization



Inverse kinematics as error minimization

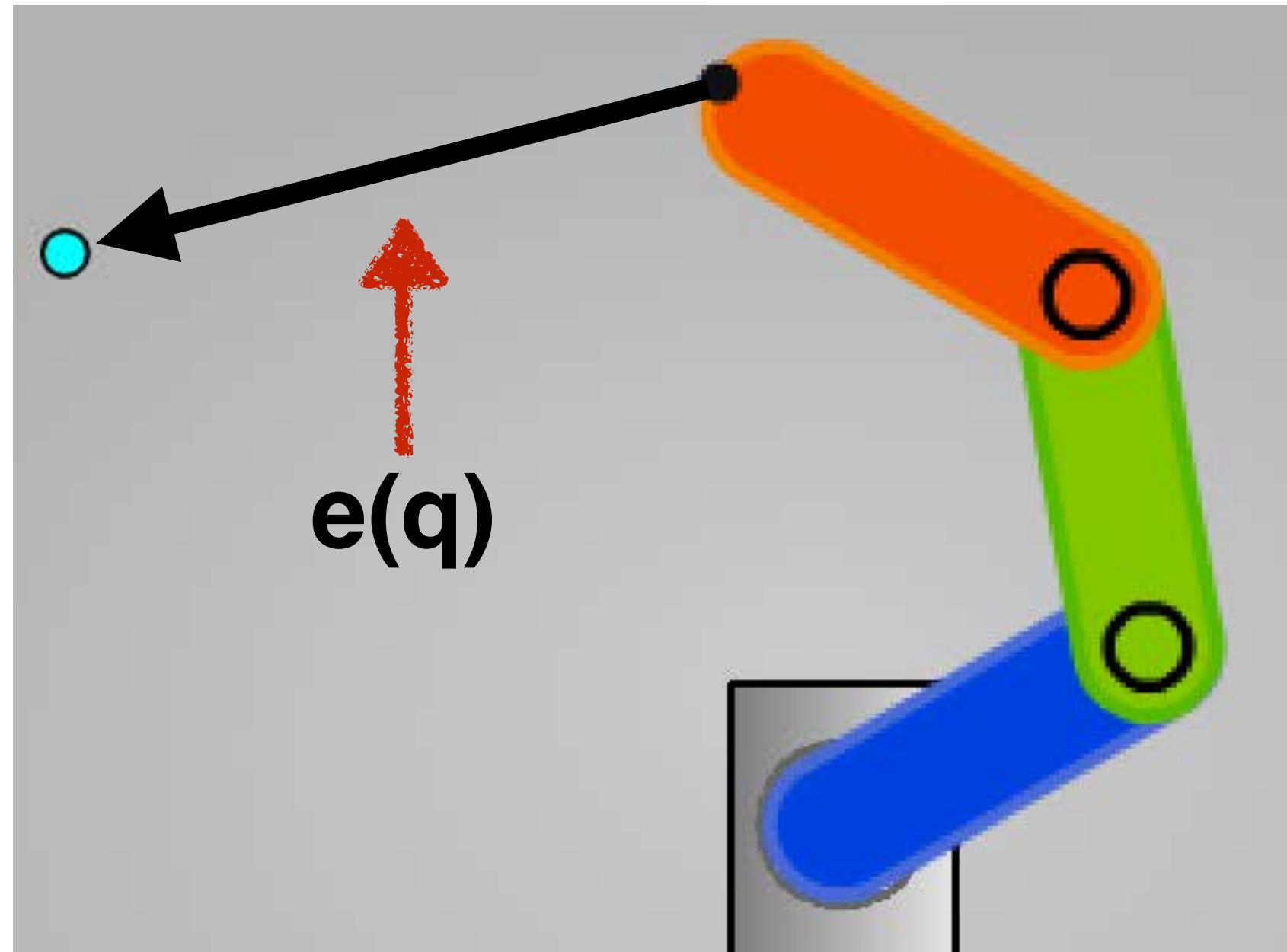


Define error function $e(\mathbf{q})$ as difference between current and desired endeffector poses

Error function parameterized by robot configuration \mathbf{q}

Find global minimum of $e(\mathbf{q})$: $\text{argmin}_{\mathbf{q}} e(\mathbf{q})$

Inverse kinematics as error minimization



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

Error function parameterized by robot configuration \mathbf{q}

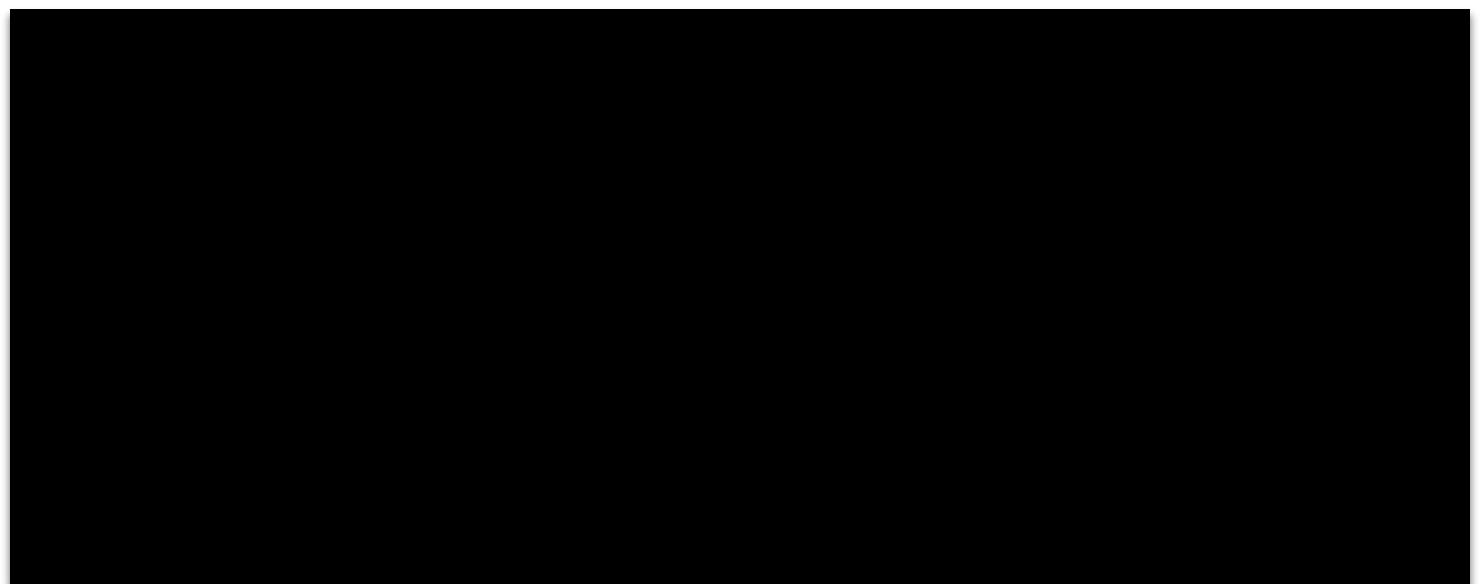
Find global minimum of $e(\mathbf{q})$: $\text{argmin}_{\mathbf{q}} e(\mathbf{q})$

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

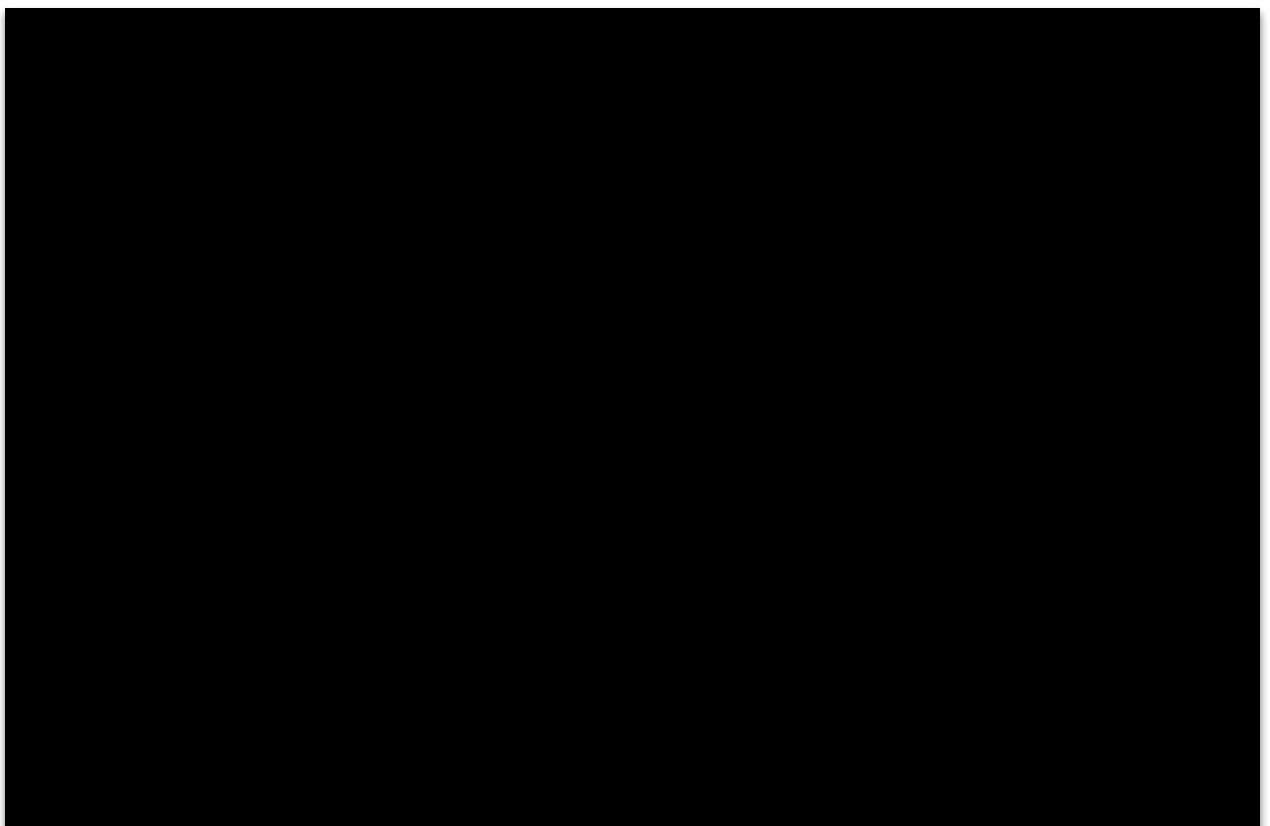
Example: Find global minimum of function

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

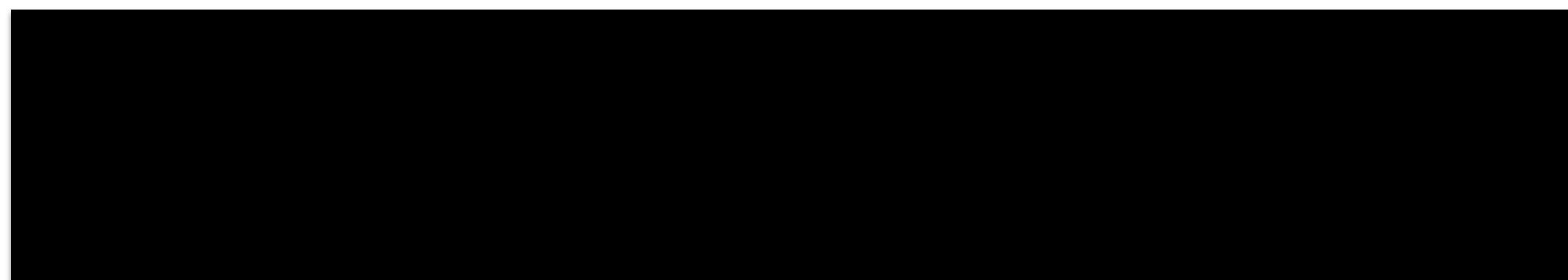
Take derivative



Solve for x where derivative is zero



Verify



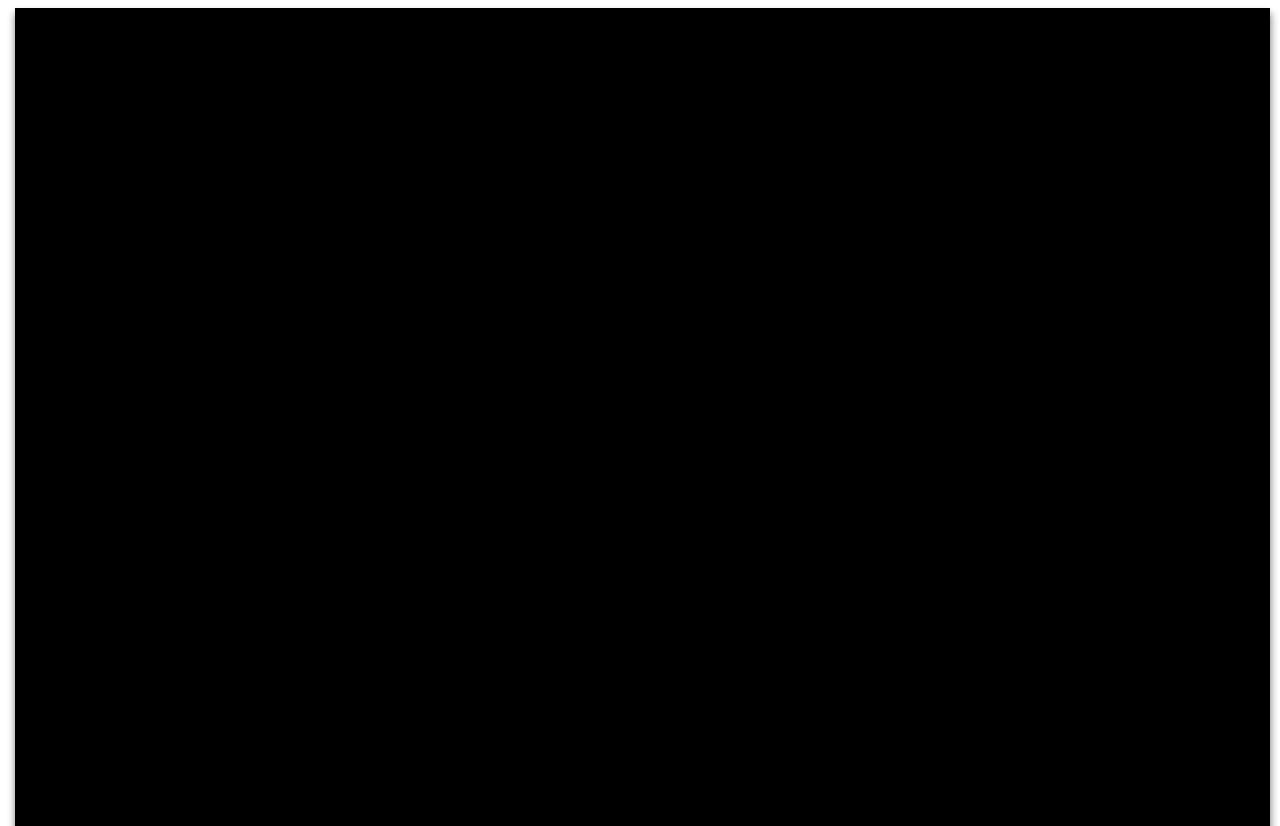
Example: Find global minimum of function

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

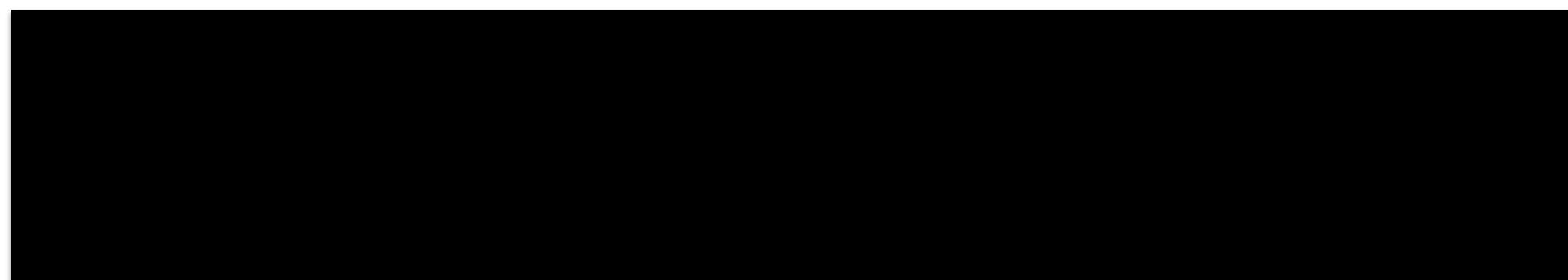
Take derivative

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

Solve for x where derivative is zero



Verify



Example: Find global minimum of function

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

Take derivative

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

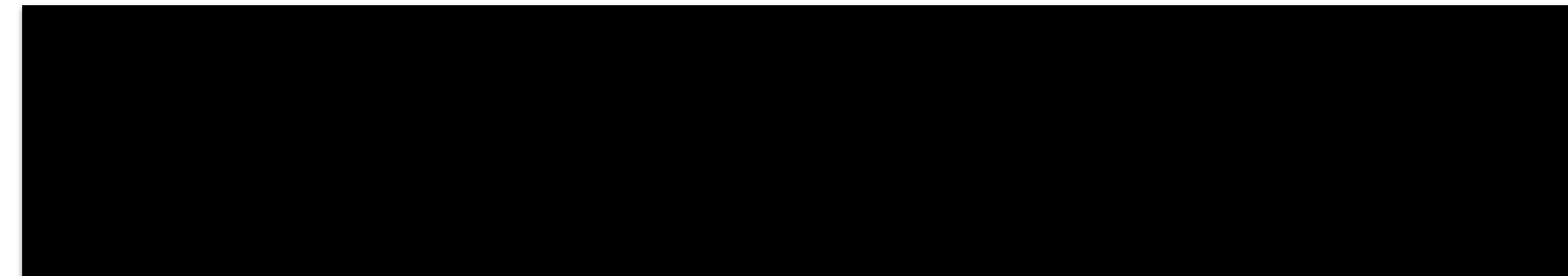
Solve for x where derivative is zero

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

$$6x - 12 = 0$$

$$x = 2$$

Verify



Example: Find global minimum of function

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

Take derivative

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

Solve for x where derivative is zero

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

$$6x - 12 = 0$$

$$x = 2$$

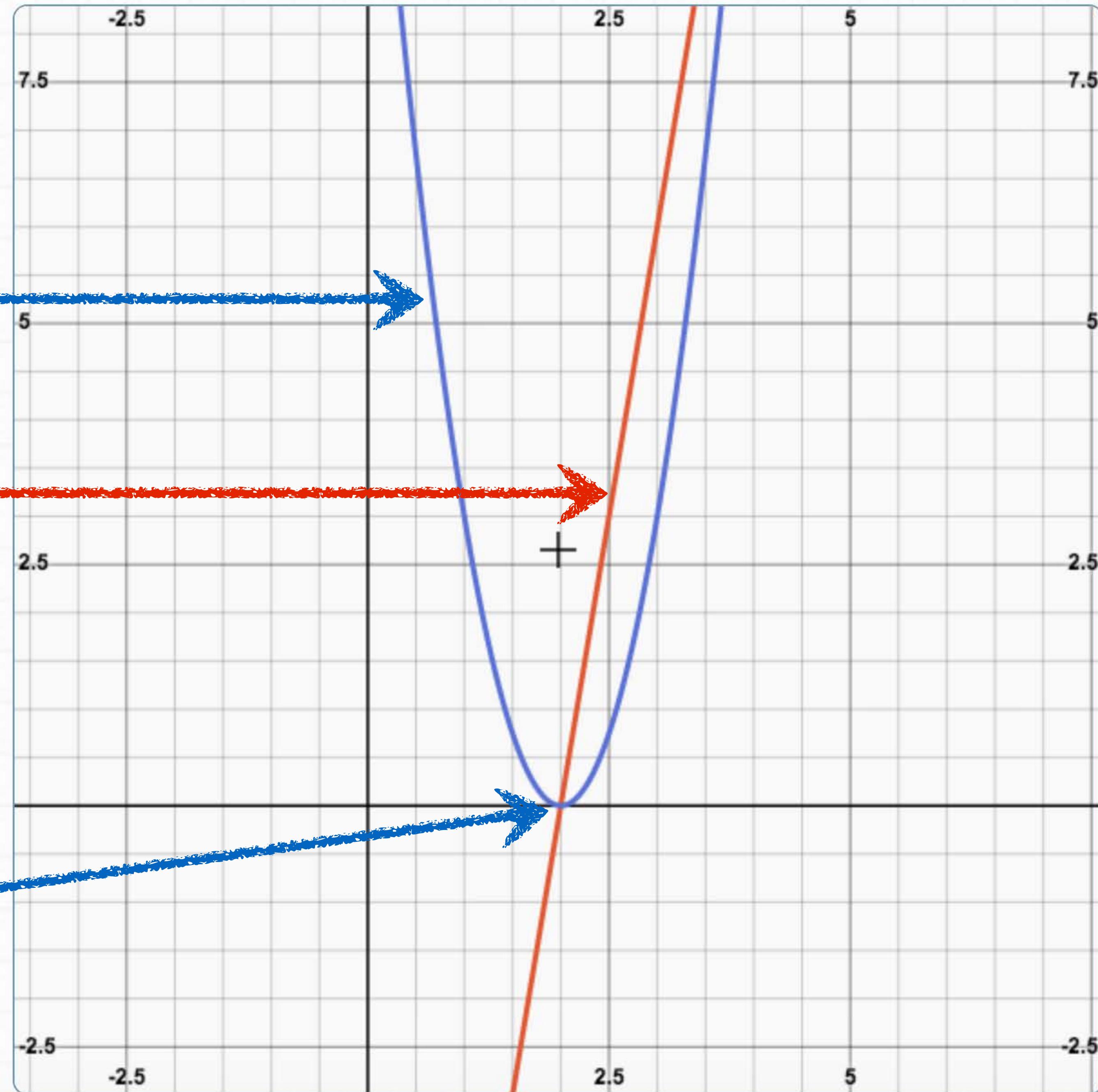
Verify $f(2) = 3((2) - 2)^2 = 0$



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

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

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



Toggle graphs:

- $f(x)$
- $f'(x)$

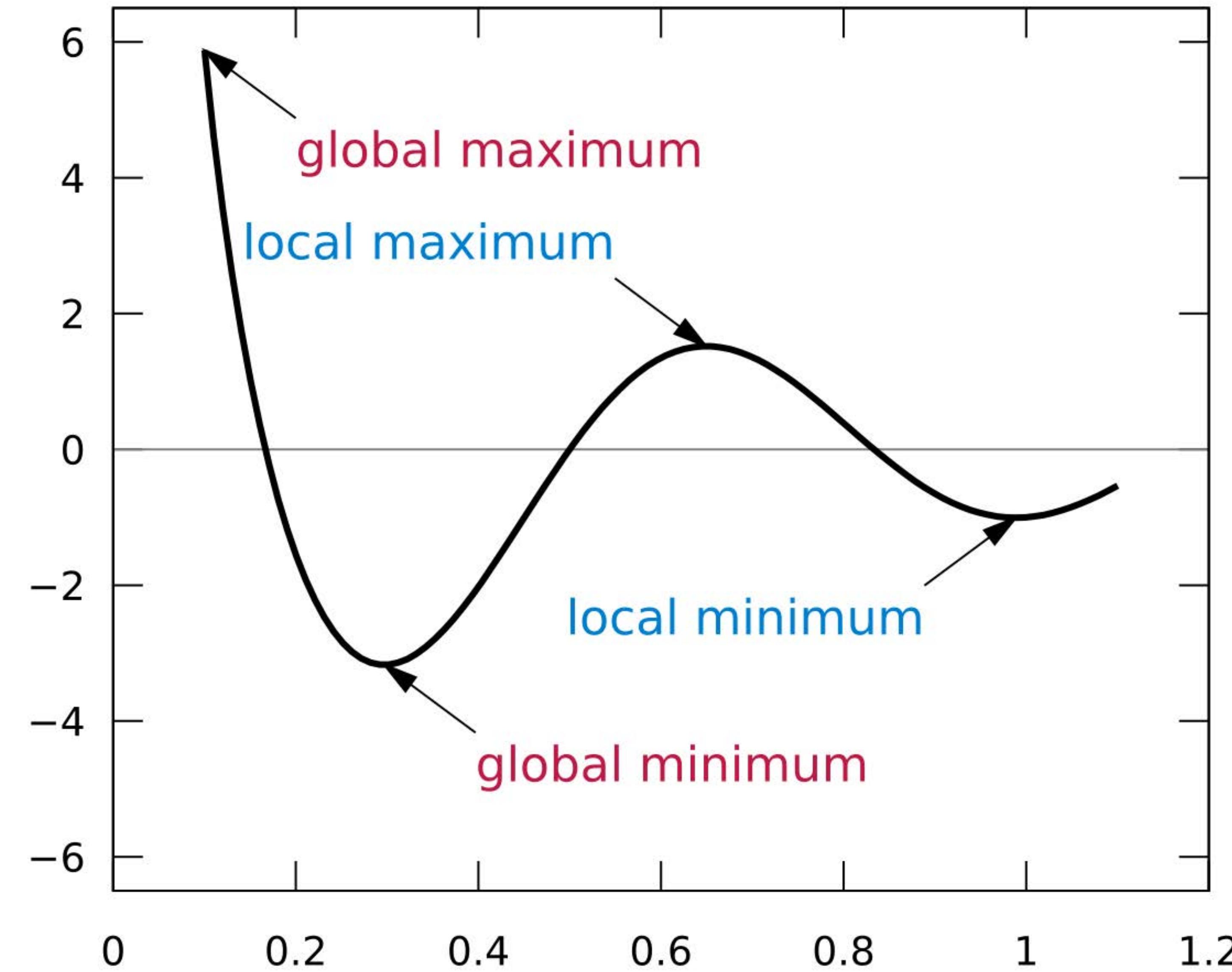
Table of values:

$x =$	<input type="text"/>
$f(x) =$	<input type="text"/>
$f'(x) =$	<input type="text"/>

Zoom mode:

- XY
- X
- Y
- 1:1

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



= Check your own answer
= 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](#)

Did You Know? ★

 Ad 

 **((SXM))** SEE WHAT YOU'VE BEEN HEARING. >

YOUR INPUT:
 $f(x) = \frac{\cos(3\pi x)}{x}$

Simplify **Roots/zeros**

FIRST DERIVATIVE:
 $\frac{d}{dx} [f(x)] = f'(x) = -\frac{3\pi \sin(3\pi x)}{x} - \frac{\cos(3\pi x)}{x^2}$

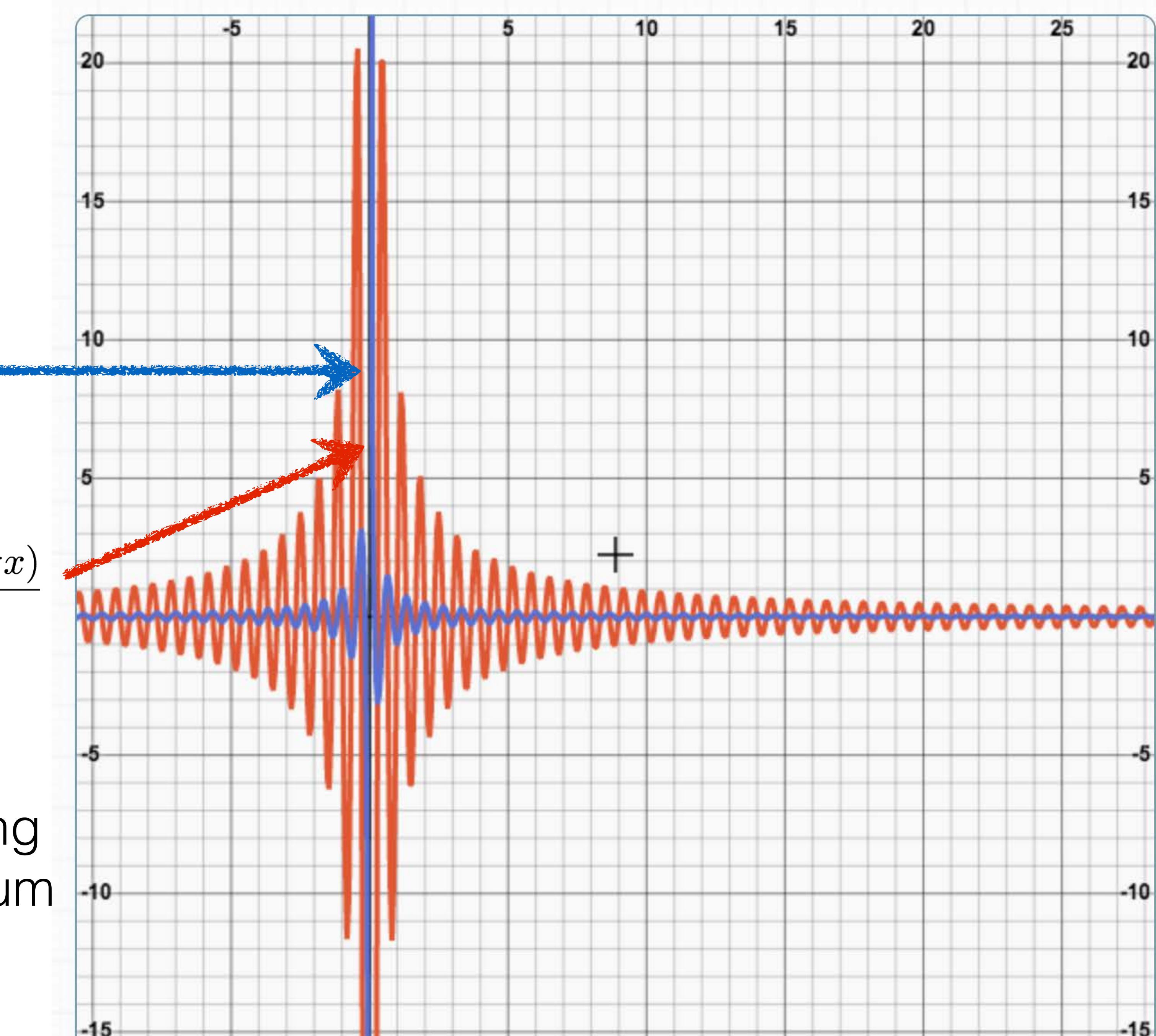
Simplify/rewrite:
 $-\frac{3\pi x \sin(3\pi x) + \cos(3\pi x)}{x^2}$

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



Toggle graphs:

- $f(x)$
- $f'(x)$

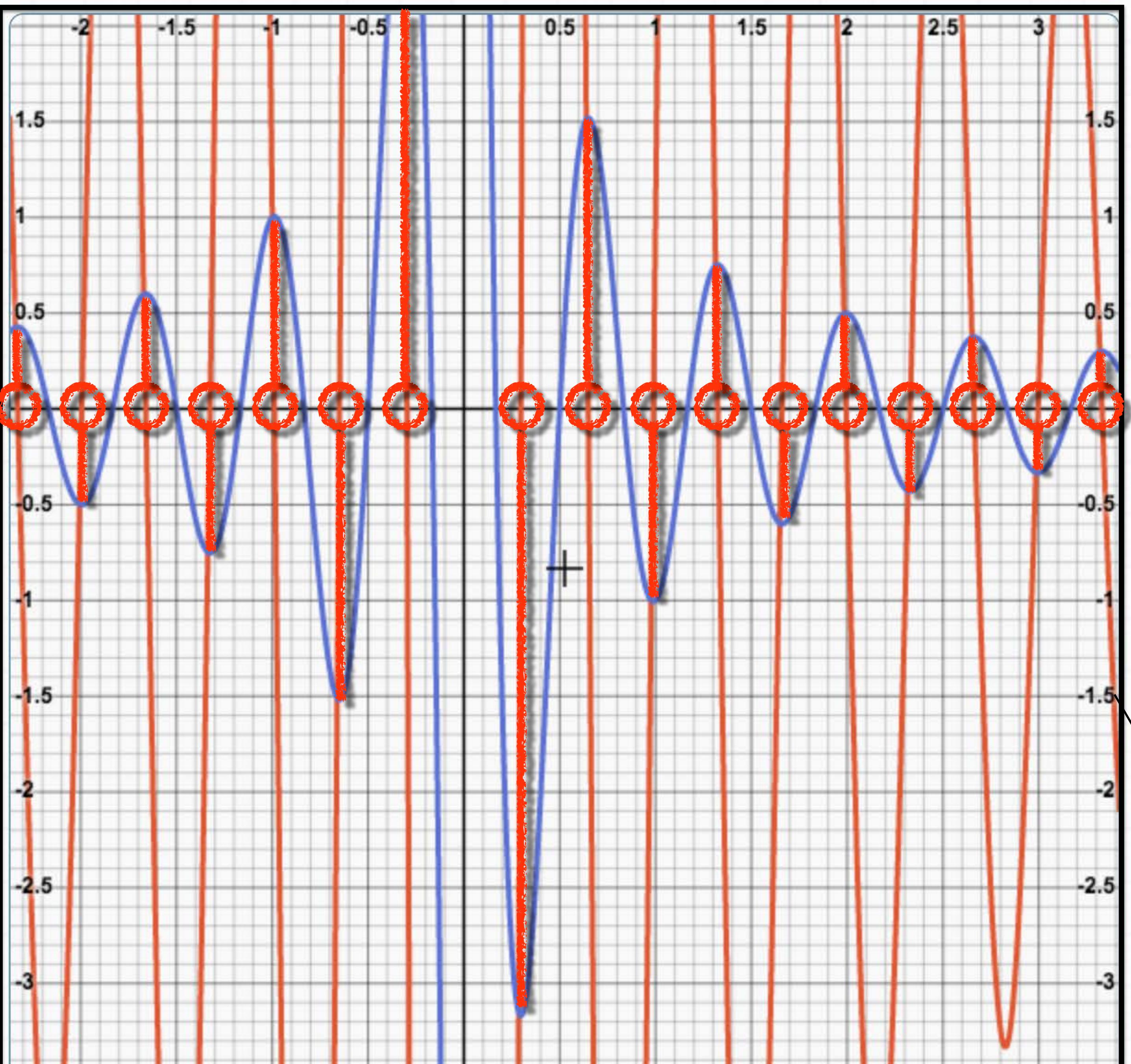
Table of values:

$x =$	<input type="text"/>
$f(x) =$	<input type="text"/>
$f'(x) =$	<input type="text"/>

Zoom mode:

- XY
- X
- Y
- 1:1**

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



Toggle graphs:

- $f(x)$
- $f'(x)$

Table of values:

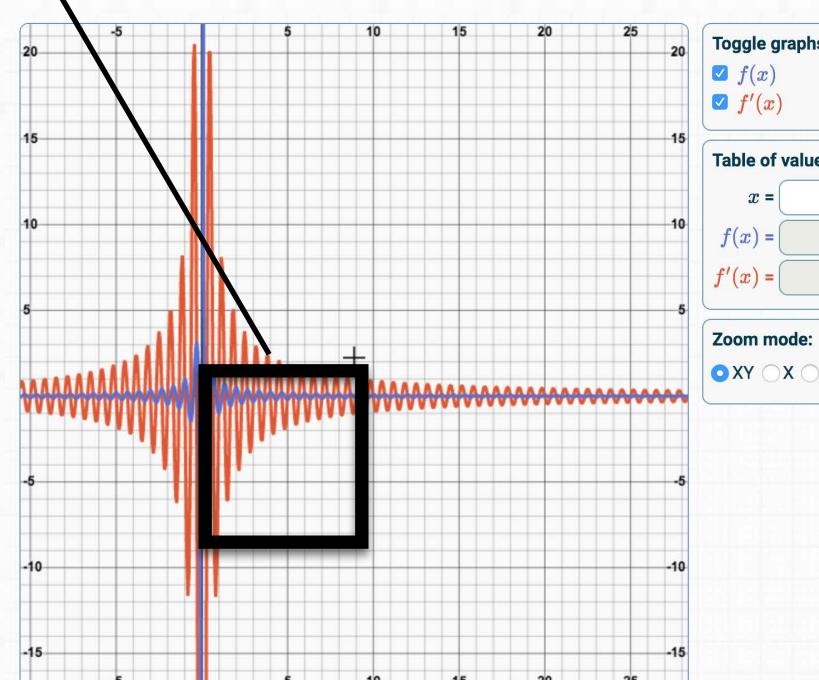
$x =$

$f(x) =$

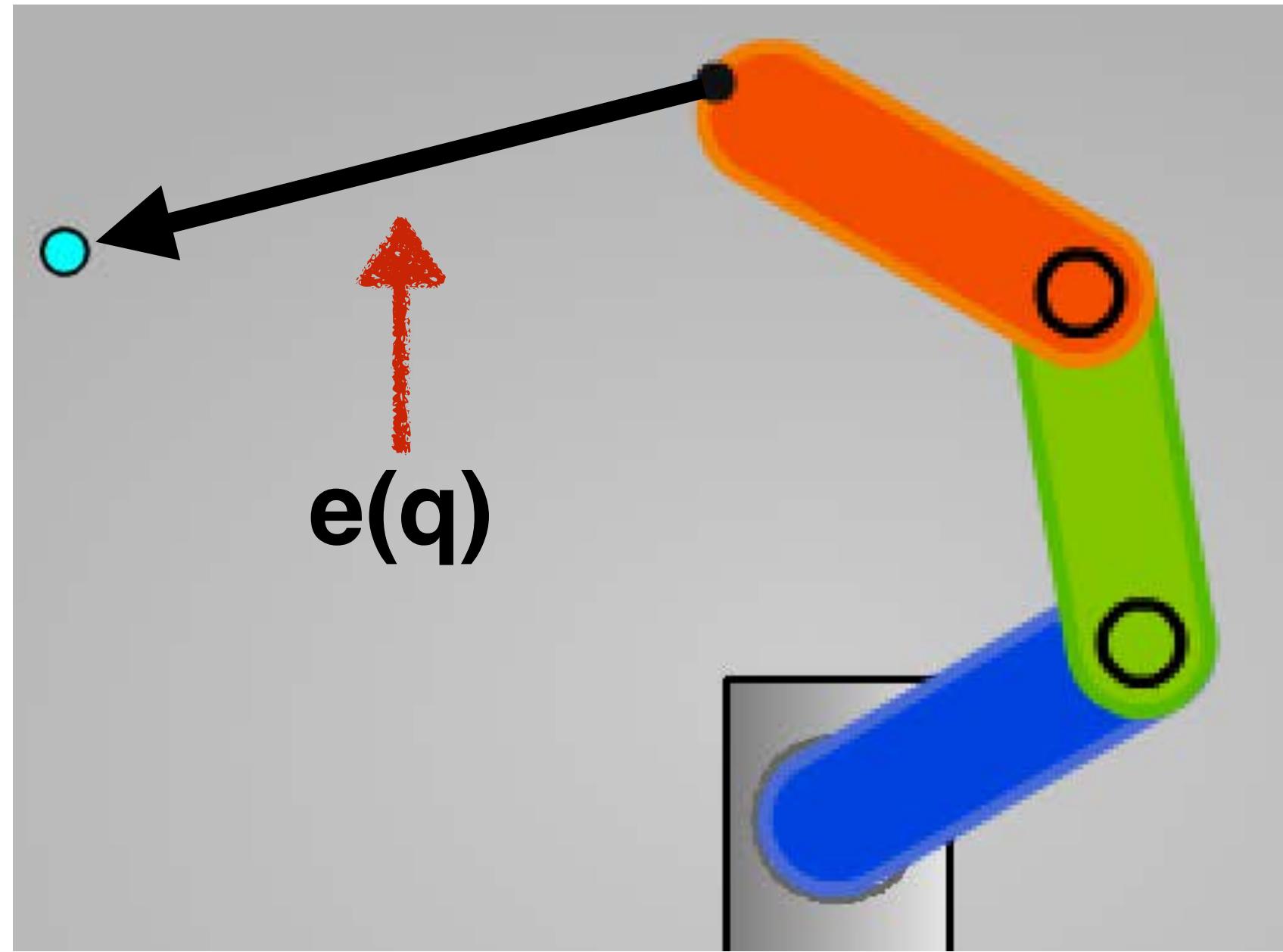
$f'(x) =$

Zoom mode:

XY X Y 1:1



Inverse kinematics as error minimization



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

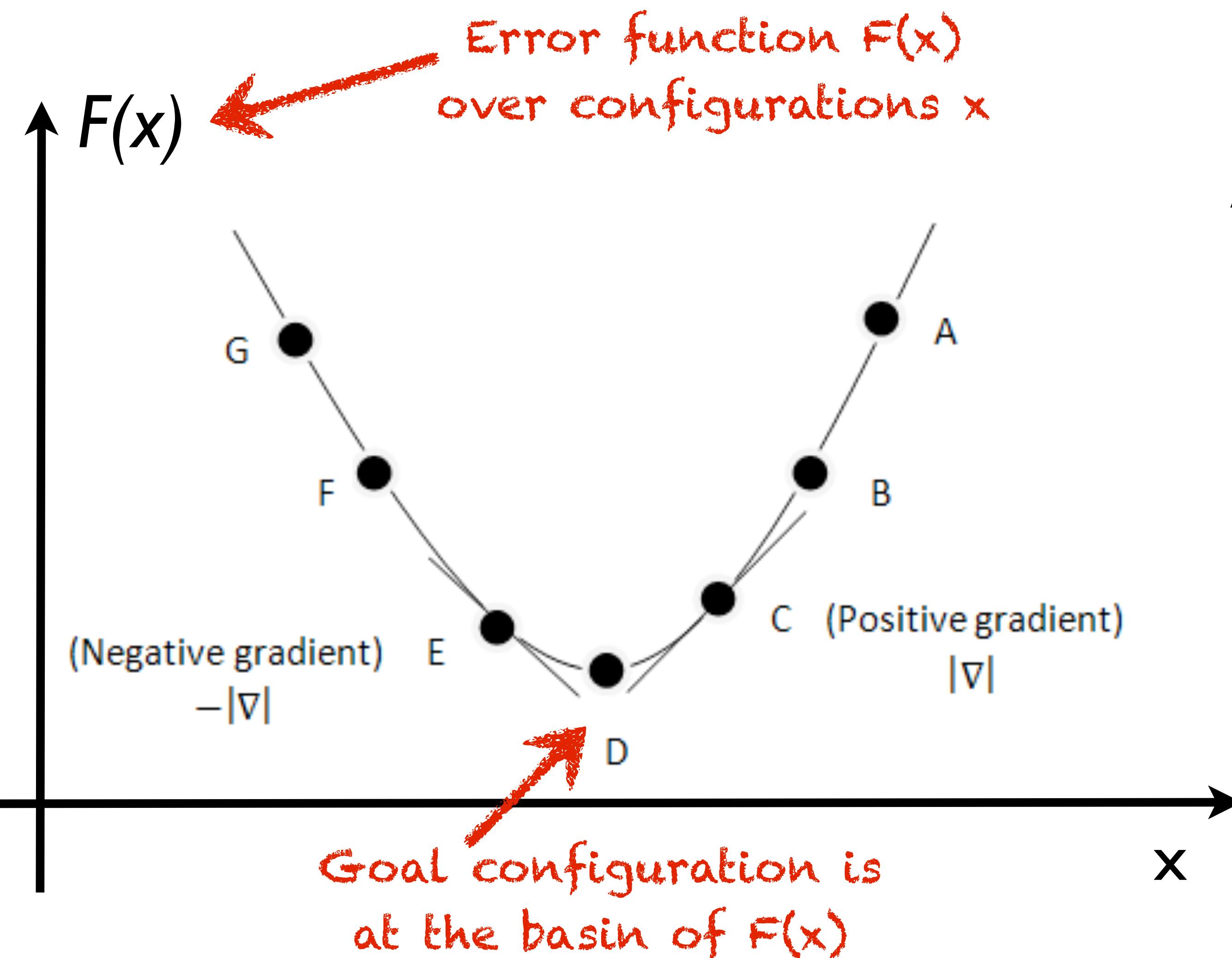
Error function parameterized by robot configuration \mathbf{q}

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

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

Gradient descent

From Wikipedia, the free encyclopedia



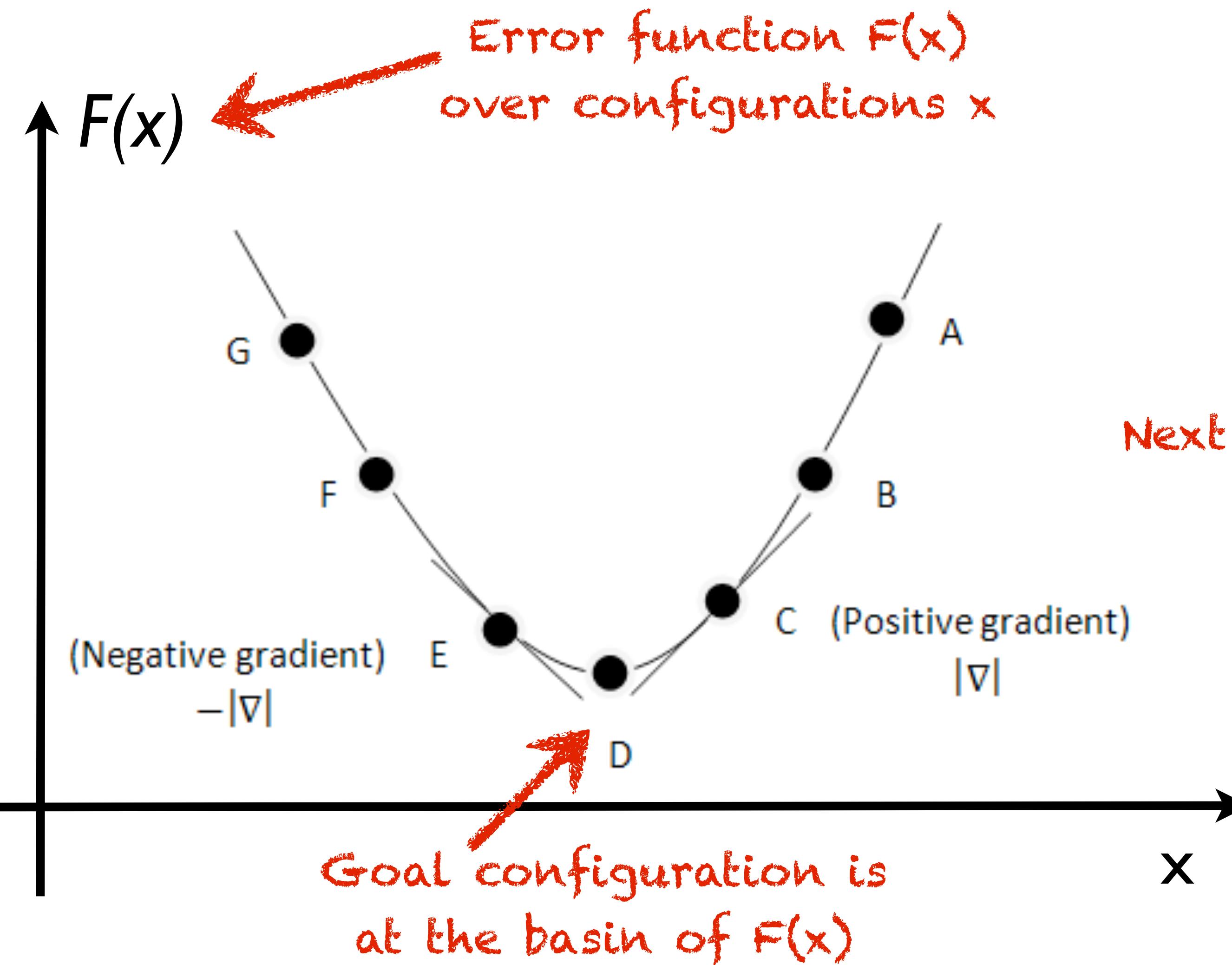
Assign initial solution guess \mathbf{x}_0

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

until $\|\mathbf{x}_i - \mathbf{x}_{i-1}\|$ is “small”

Gradient descent

From Wikipedia, the free encyclopedia



Current solution "Learning rate"

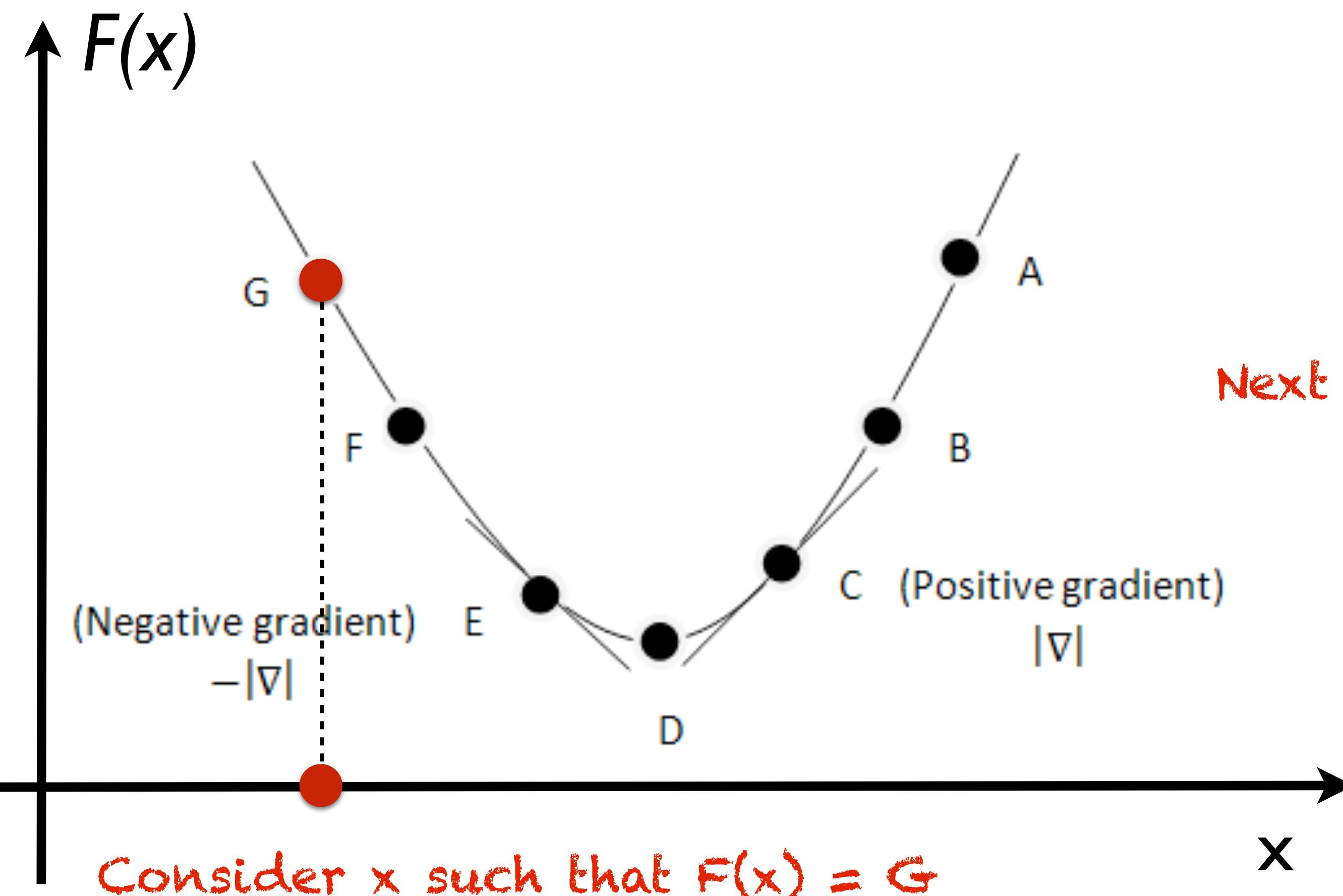
$$x_{i+1} = x_i - \gamma \nabla F(x_i)$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



Current solution

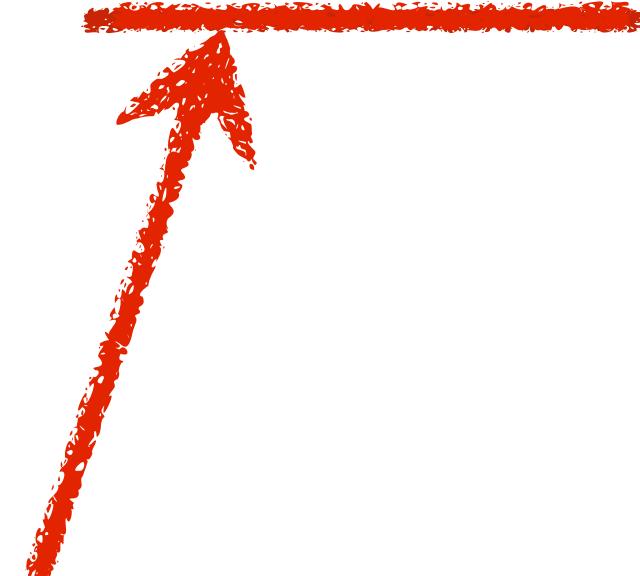
$$x_{i+1} = x_i - \gamma_i \nabla F(x_i)$$



Next solution

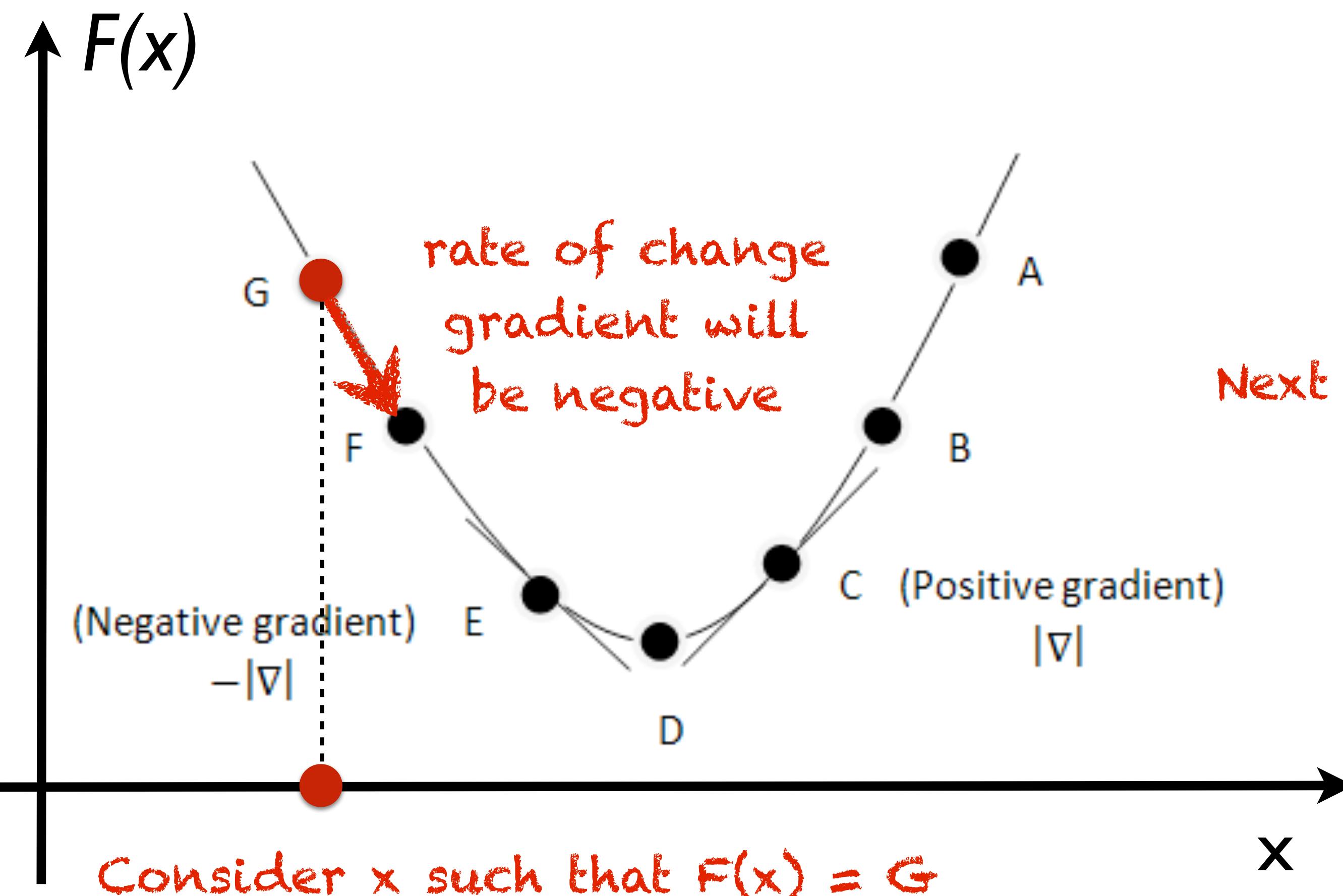
"Learning rate"

Derivative assumed to be direction of steepest ascent away from goal



Gradient descent

From Wikipedia, the free encyclopedia



Current solution "Learning rate"

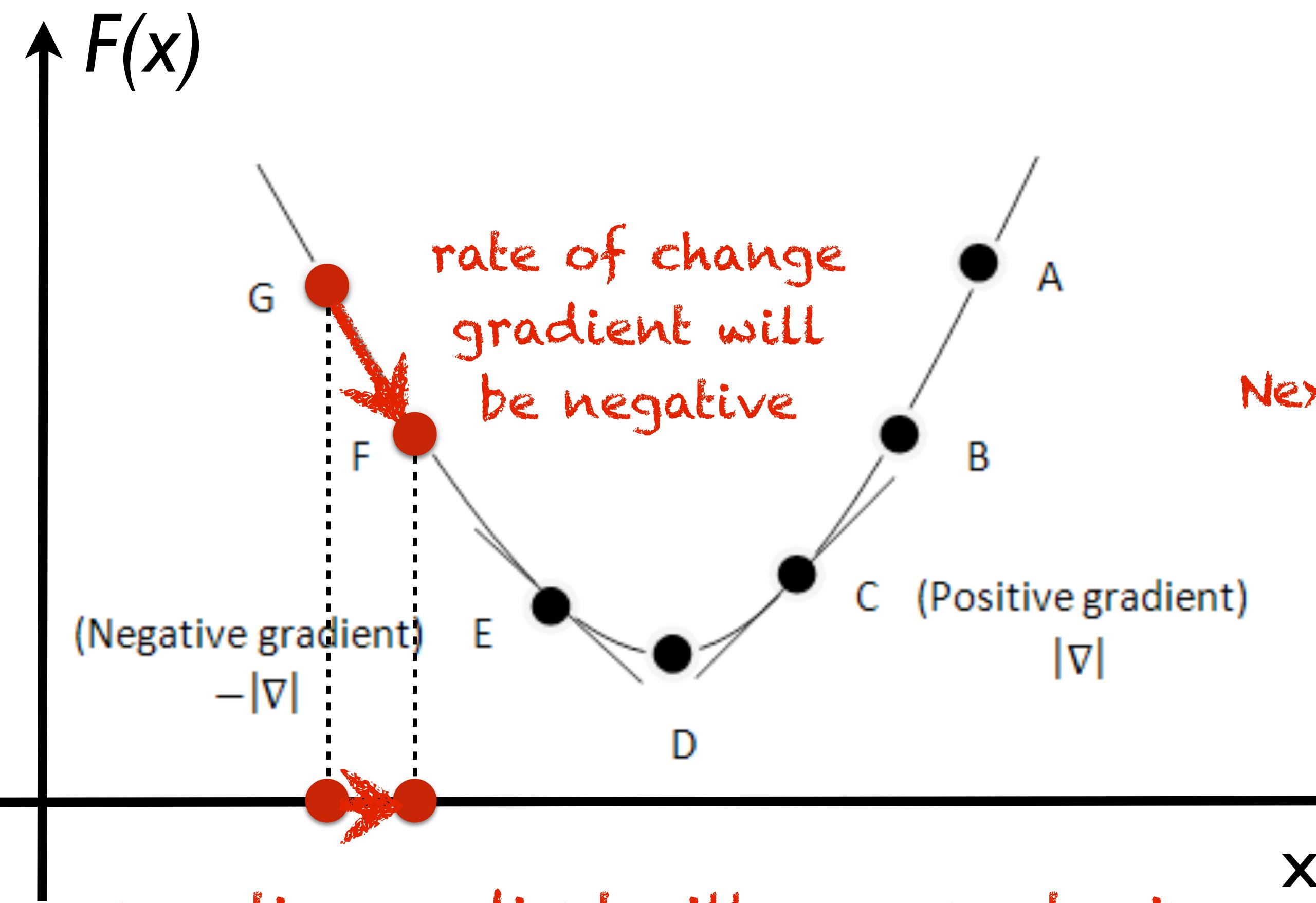
$$x_{i+1} = x_i - \gamma \nabla F(x_i)$$

Next solution

Derivative assumed to be direction
of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



Current solution "Learning rate"

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

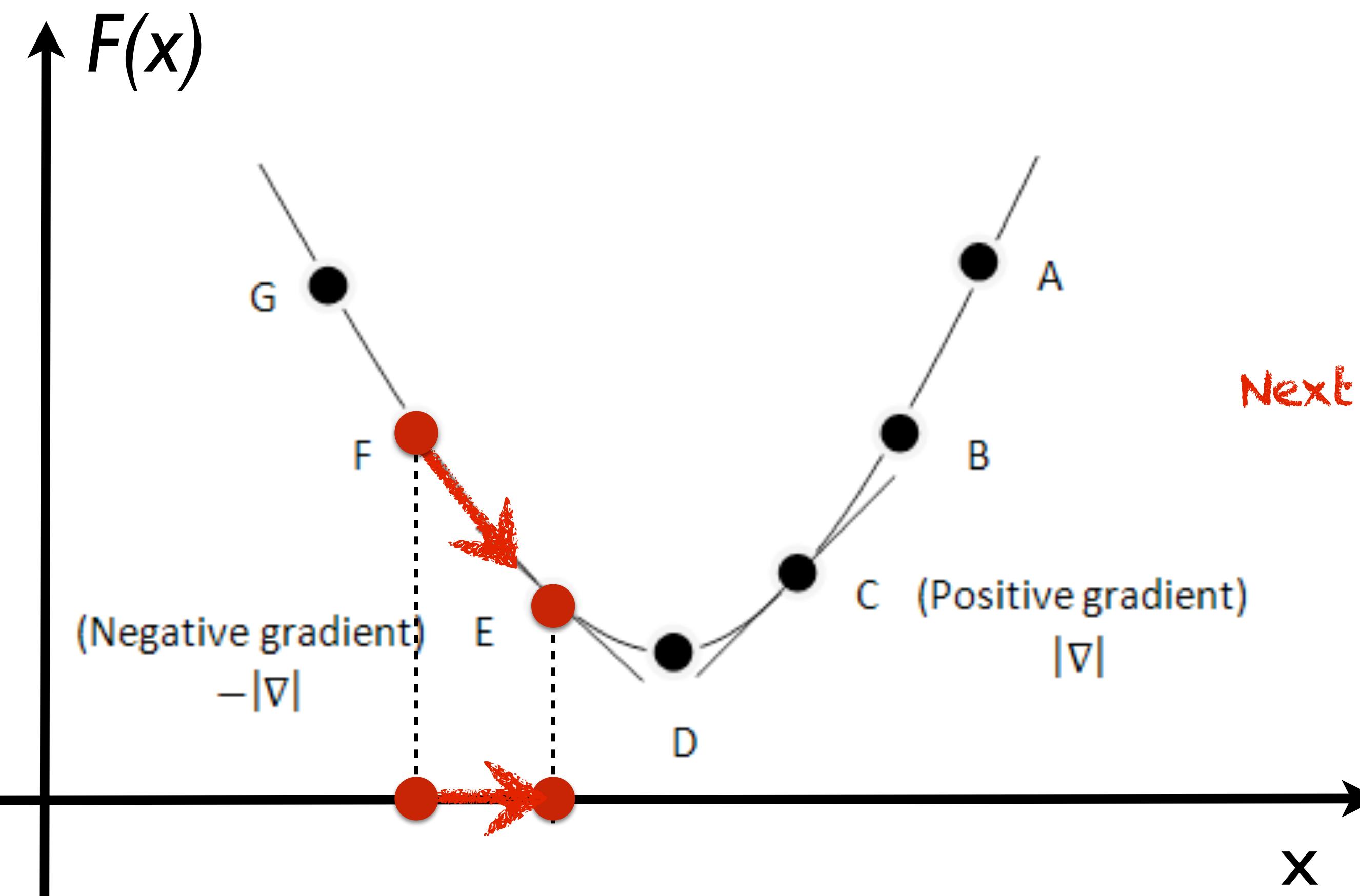
Next solution

Derivative assumed to be direction of steepest ascent away from goal

negative gradient will move next x in positive direction

Gradient descent

From Wikipedia, the free encyclopedia



Current solution "Learning rate"

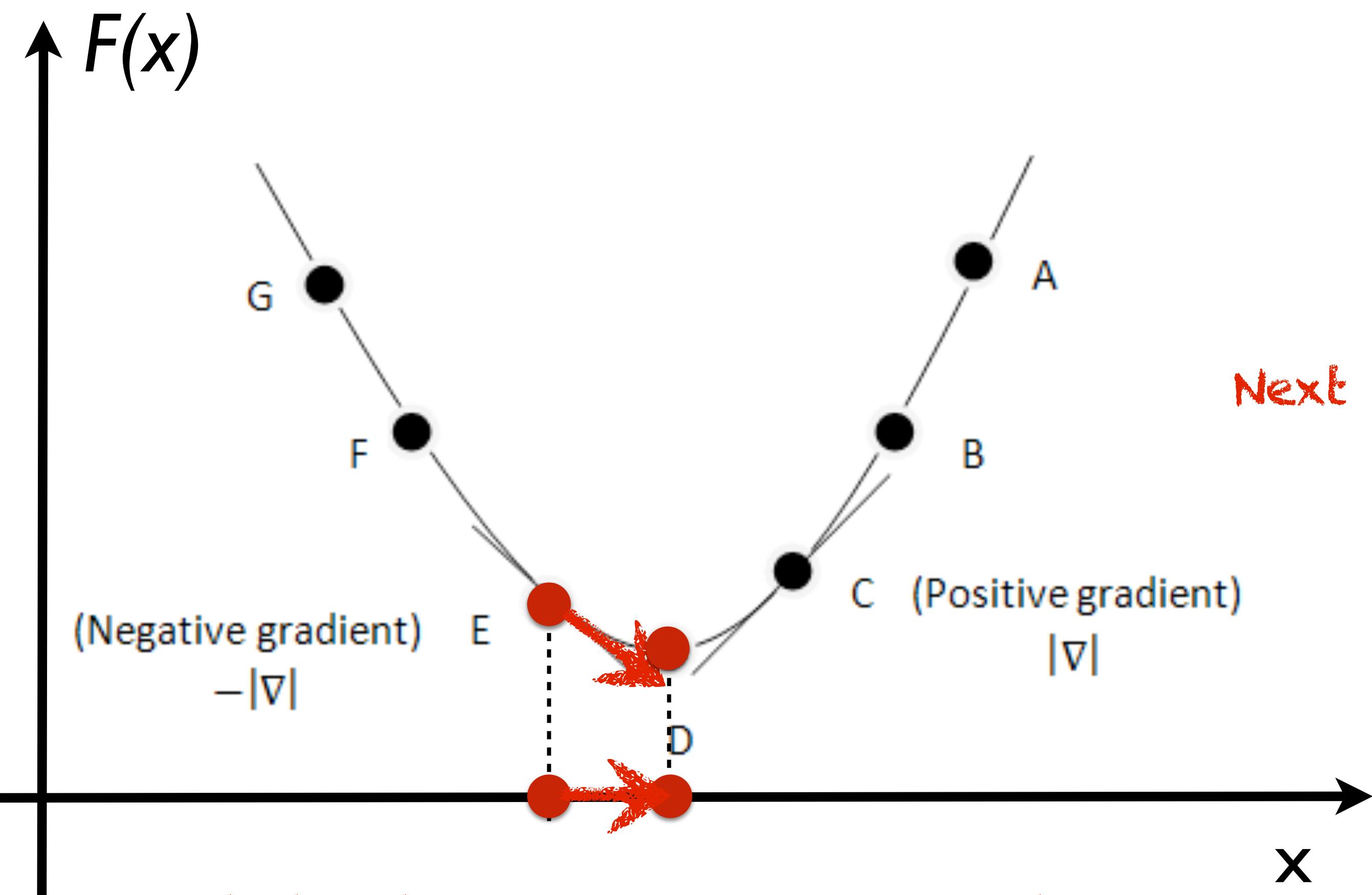
$$x_{i+1} = x_i - \gamma \nabla F(x_i)$$

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"

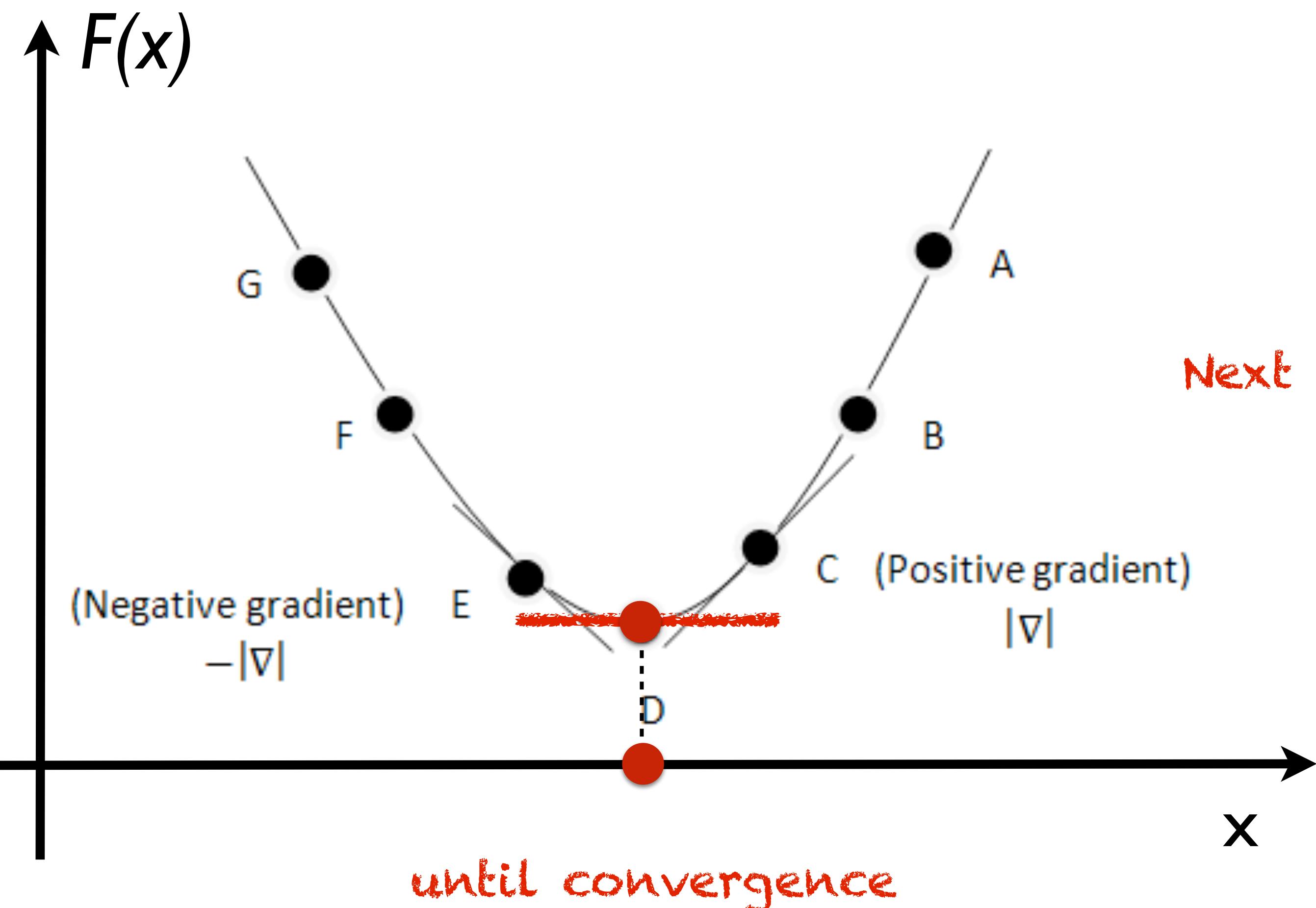
$$x_{i+1} = x_i - \gamma \nabla F(x_i)$$

Next solution

Derivative assumed to be direction of steepest ascent away from goal

Gradient descent

From Wikipedia, the free encyclopedia



Current solution "Learning rate"

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

Next solution

Derivative assumed to be direction of steepest ascent away from goal

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

Robot arm and its Jacobian

3D N-joint arm

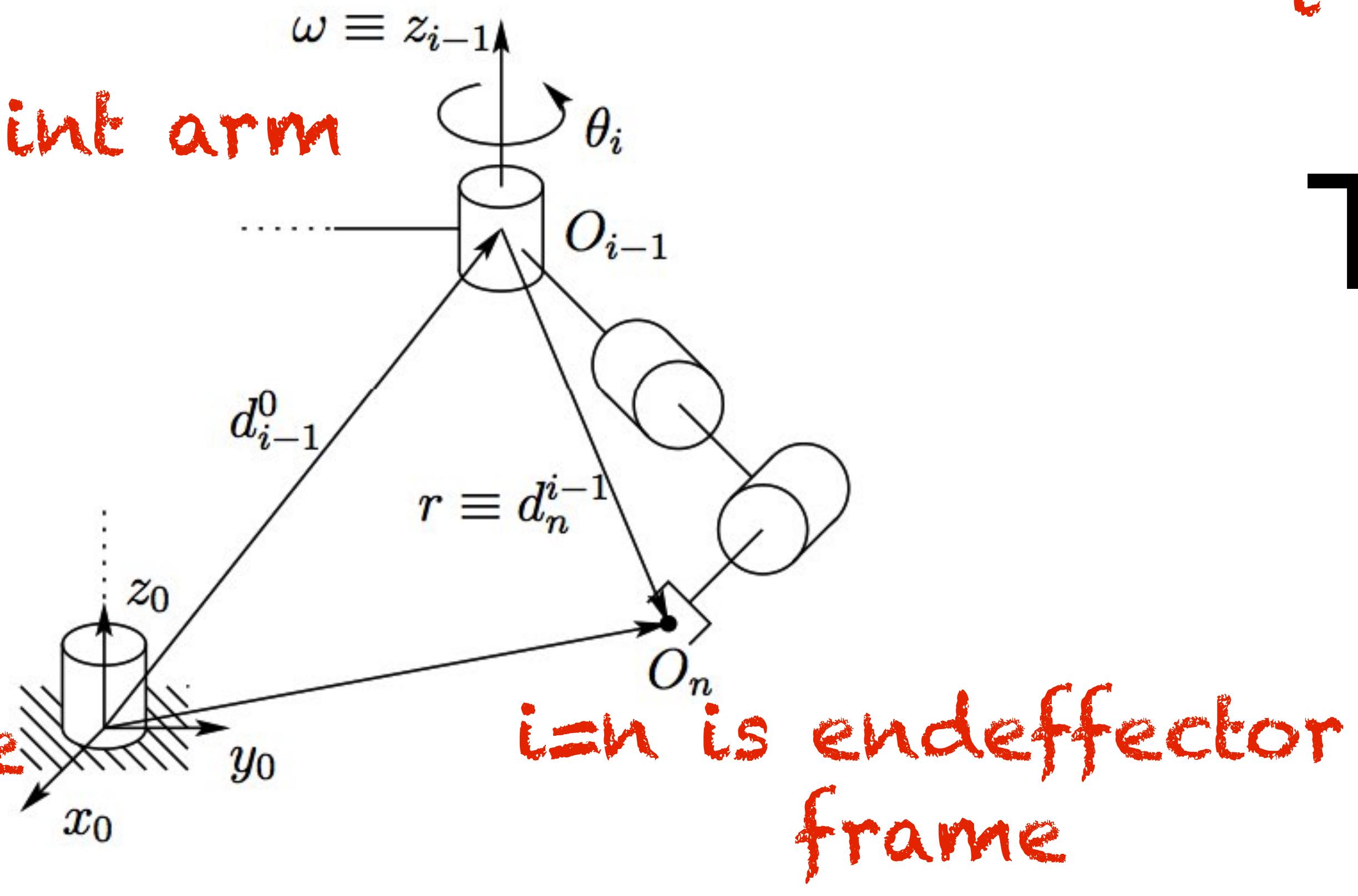


Figure 5.1: Motion of the end-effector due to link i .

$i-1^{\text{th}}$ frame maps to i^{th} column in

The Jacobian

A $6 \times N$ matrix

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

Robot arm and its Jacobian

Lets focus on
i-1th frame

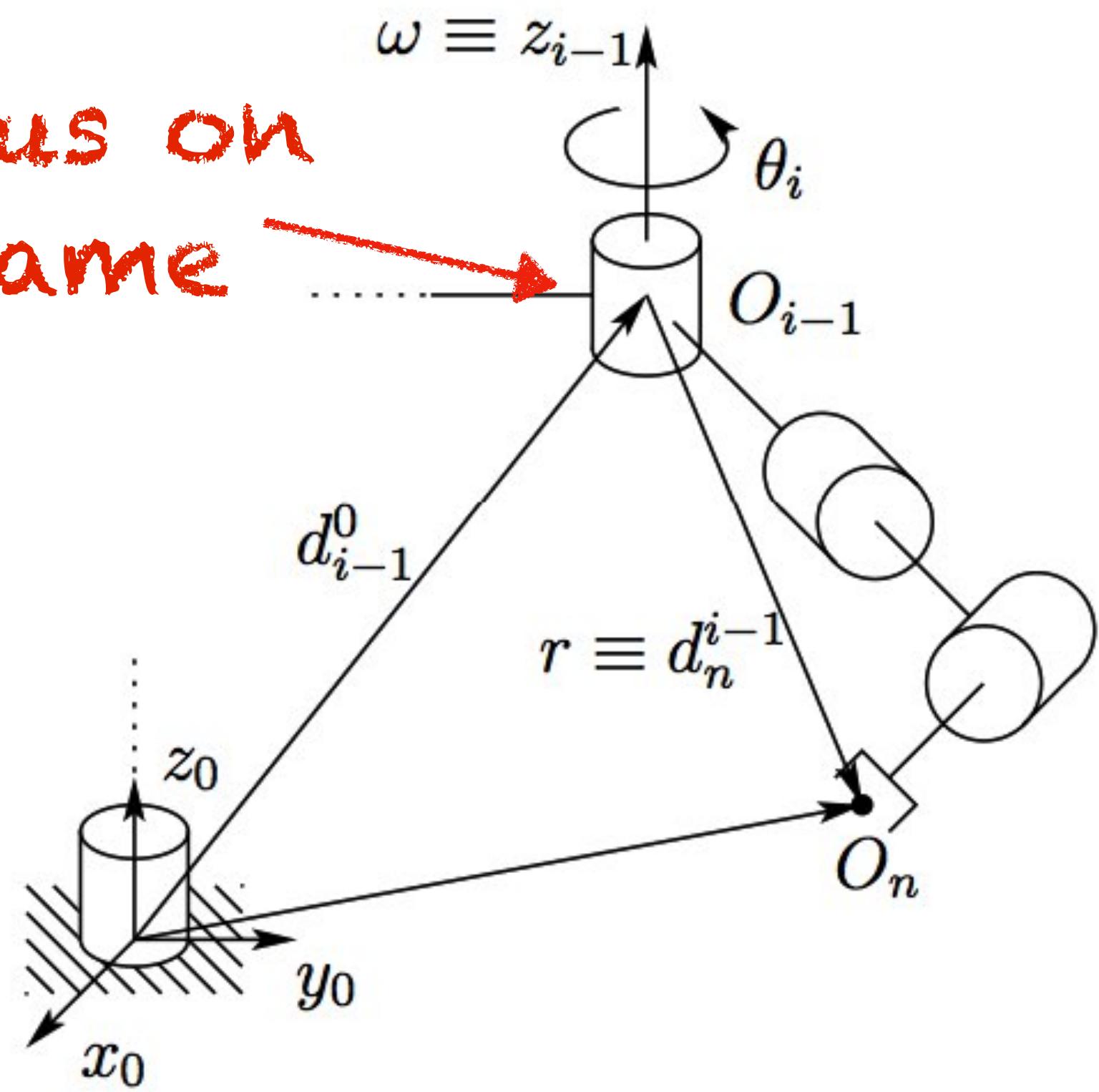


Figure 5.1: Motion of the end-effector due to link i .

*i-1th frame maps to *ith column in The Jacobian**

A $6 \times N$ matrix

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

*This will correspond to *ith column**

Robot arm and its Jacobian

Lets focus on
i-1th frame

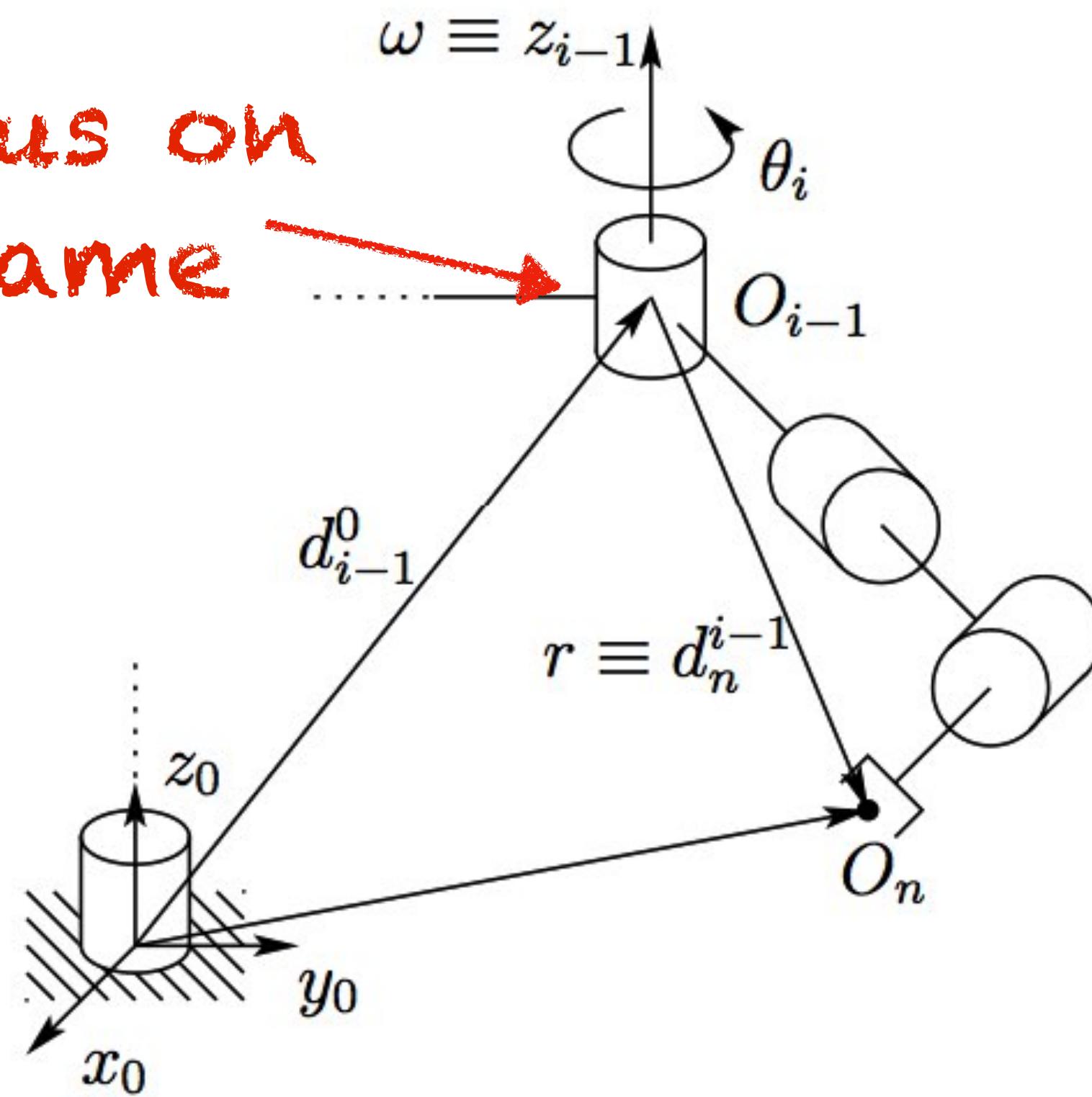


Figure 5.1: Motion of the end-effector due to link i .

J_i for a prismatic joint

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

J_i for a rotational joint

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

i-1th frame maps to ith column in The Jacobian

A $6 \times N$ matrix

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

consisting of two $3 \times N$ matrices

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

Robot arm and its Jacobian

Lets focus on
i-1th frame

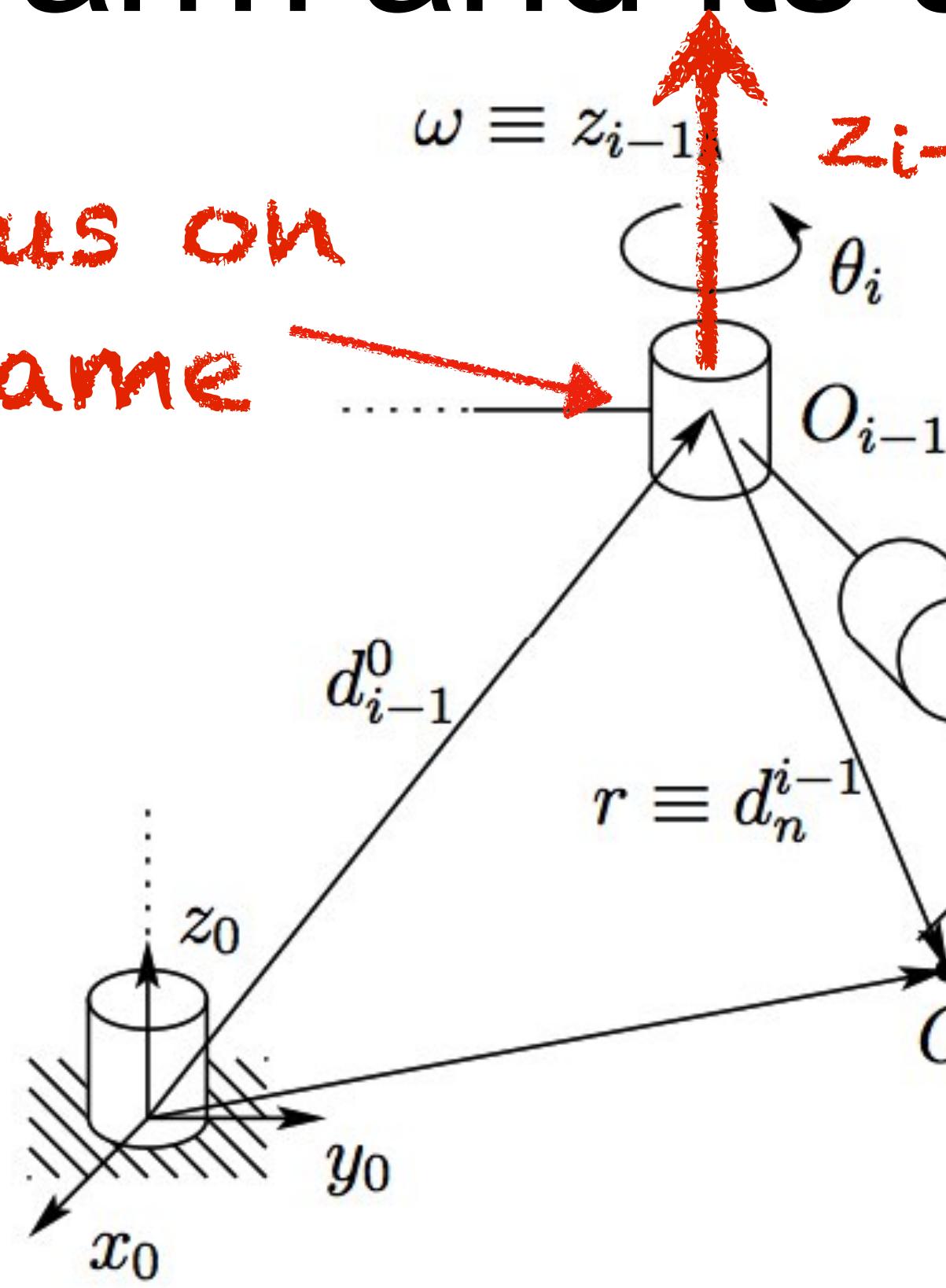


Figure 5.1: Motion of the end-effector due to link i .

z_{i-1} : joint axis

If the $i-1$ th joint is
prismatic

J_i for a prismatic joint

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

What is z_{i-1} capturing?

z_{i-1} is a 3×1 vector capturing the influence of this joint on the end-effector pose.

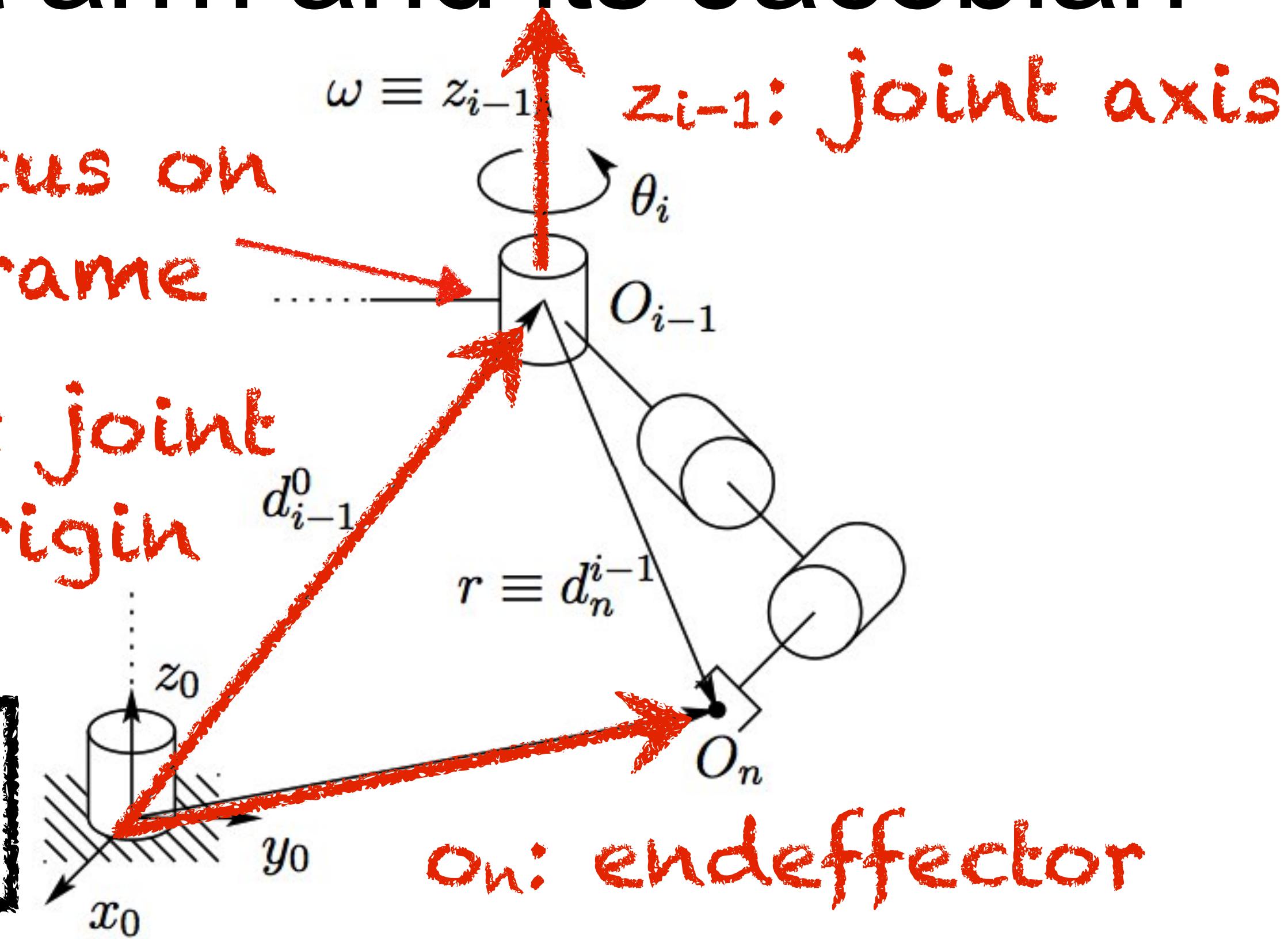
Only influences the translational (linear) component

Robot arm and its Jacobian

Lets focus on
 $i-1$ th frame

O_{i-1} : joint
origin

vectors in
base frame



If the $i-1$ th joint is
revolute

J_i for a rotational joint

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

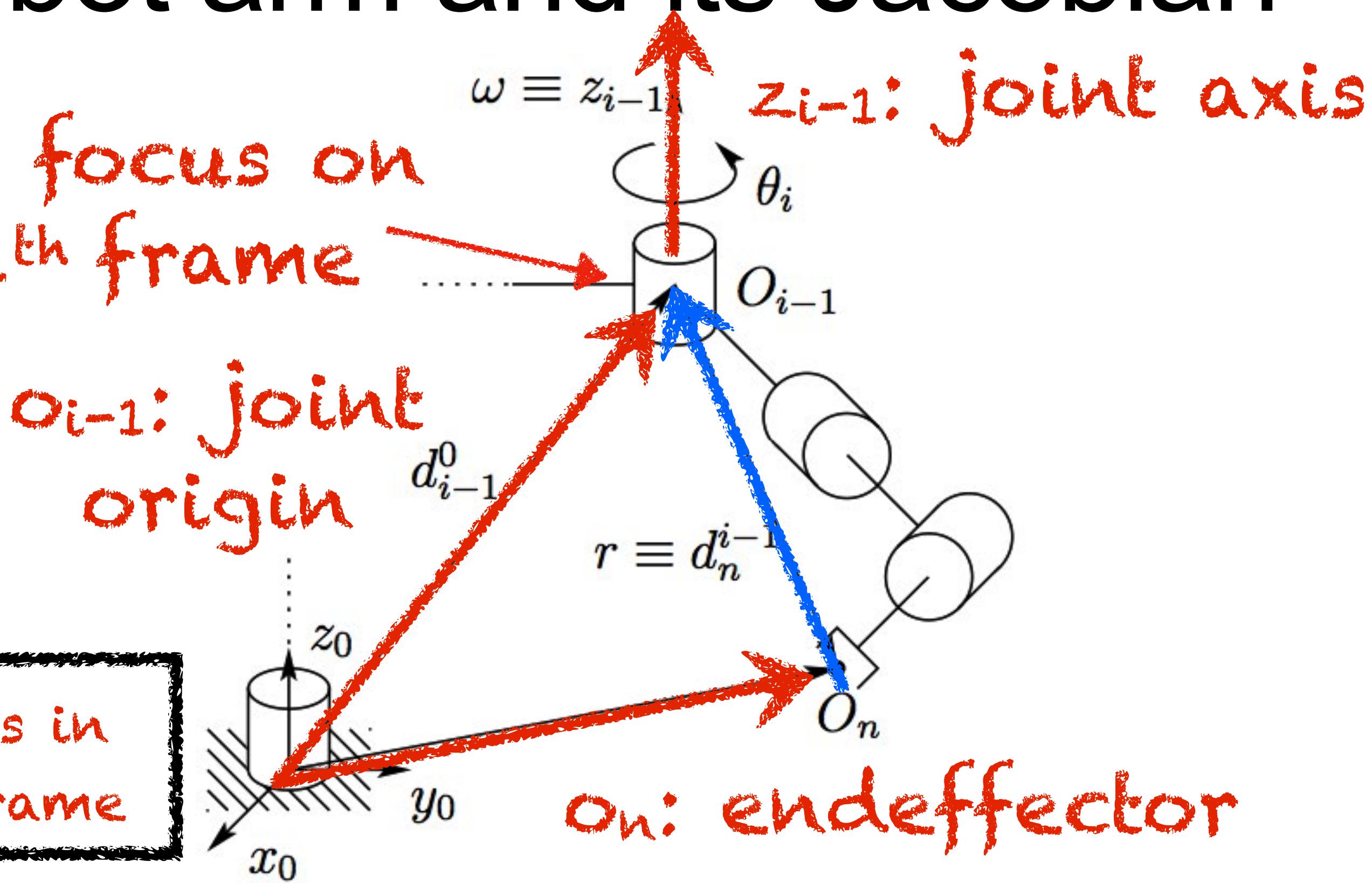
What is $z_{i-1} \times (O_n - O_{i-1})$
capturing?

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

Robot arm and its Jacobian

Lets focus on
 $i-1$ th frame

vectors in
base frame



If the $i-1$ th joint is
revolute

J_i for a rotational joint

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

What is $z_{i-1} \times (O_n - O_{i-1})$
capturing?

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

Robot arm and its Jacobian

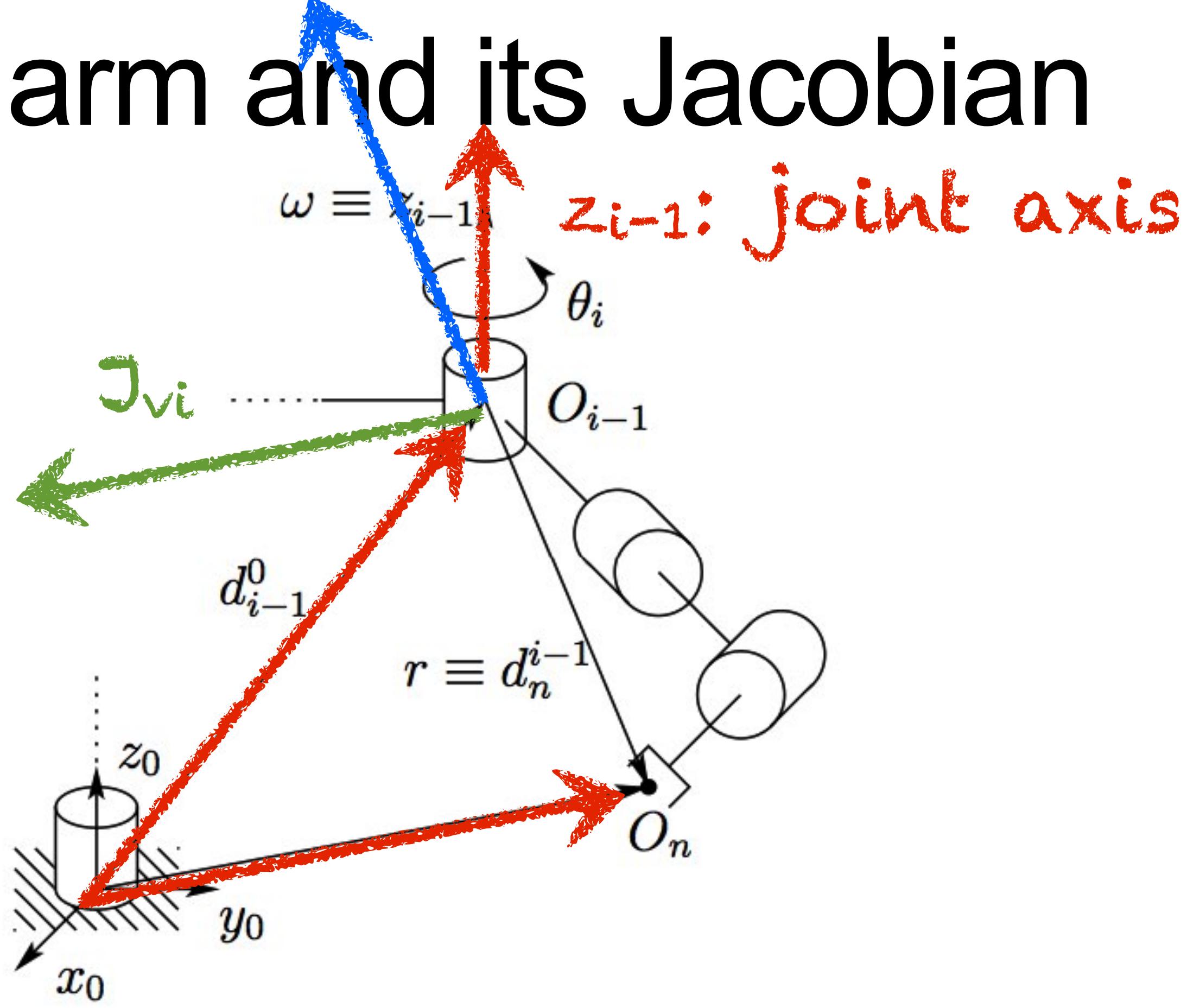


Figure 5.1: Motion of the end-effector due to link i .

If the $i-1$ th joint is revolute

J_i for a rotational joint

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

What is $z_{i-1} \times (O_n - O_{i-1})$ capturing?

The influence of this joint on the end-effector's translational component.

What is z_{i-1} capturing?

The influence of this joint on the end-effector's rotational component.

How did we get the Geometric Jacobian?

Velocity of Point Rotating on N-link Arm

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

Angular Velocity

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

assuming velocities expressed in the same frame

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

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

J_i for a rotational joint

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

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



Velocity of Point Rotating on N-link Arm

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

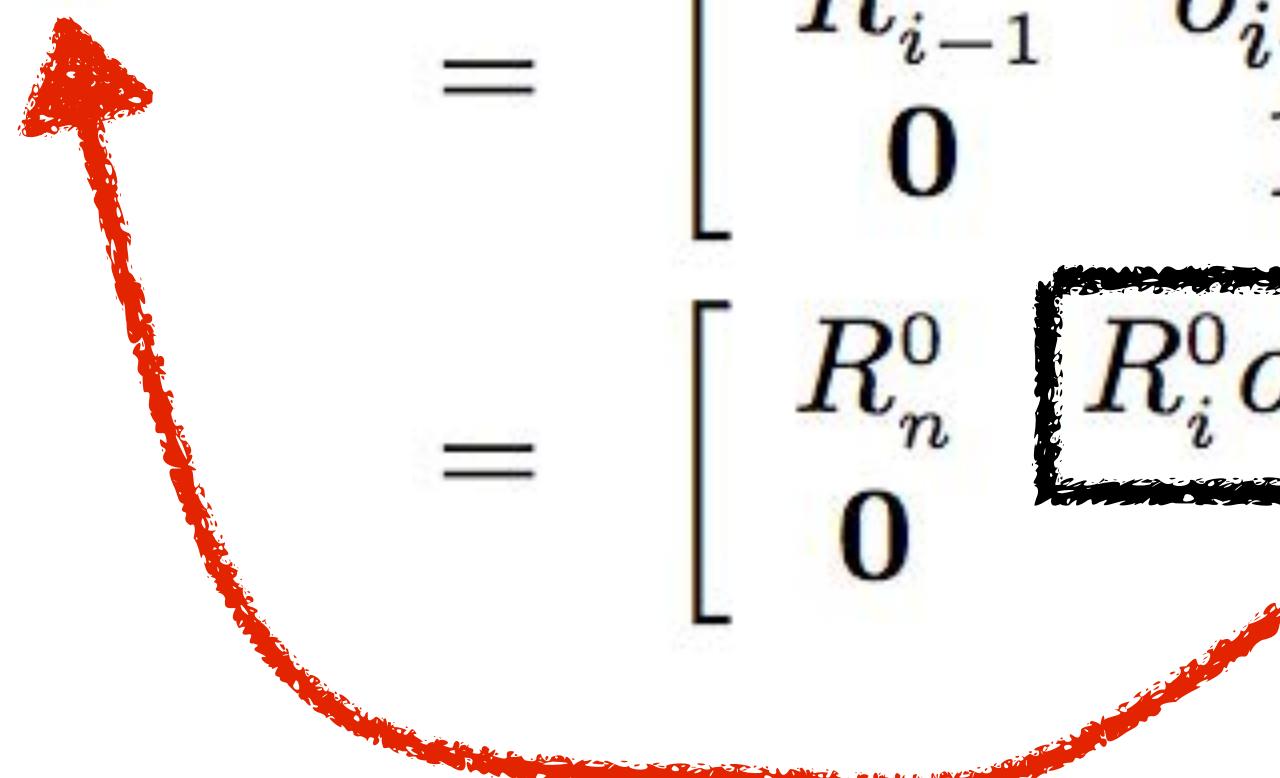
consider effect of all frames
(o..n) on endeffector

Velocity of Point Rotating on N-link Arm

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

Linear Velocity for Rotational Joint

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



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

position of endeffector frame

Velocity of Point Rotating on N-link Arm

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

Linear Velocity for Rotational Joint

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

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

take derivative wrt.
ith joint angle

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

Velocity of Point Rotating on N-link Arm

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

Linear Velocity for Rotational Joint

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

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

$$= \dot{\theta}_i S(z_{i-1}^0) R_i^0 o_n^i + \dot{\theta}_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1}$$

$$= \dot{\theta}_i z_{i-1}^0 \times (o_n^0 - o_{i-1}^0)$$

$$Jv_i = z_{i-1} \times (o_n - o_{i-1})$$

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

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 Procedure Restated

Geometric The ~~J~~Jacobian

A $6 \times N$ matrix

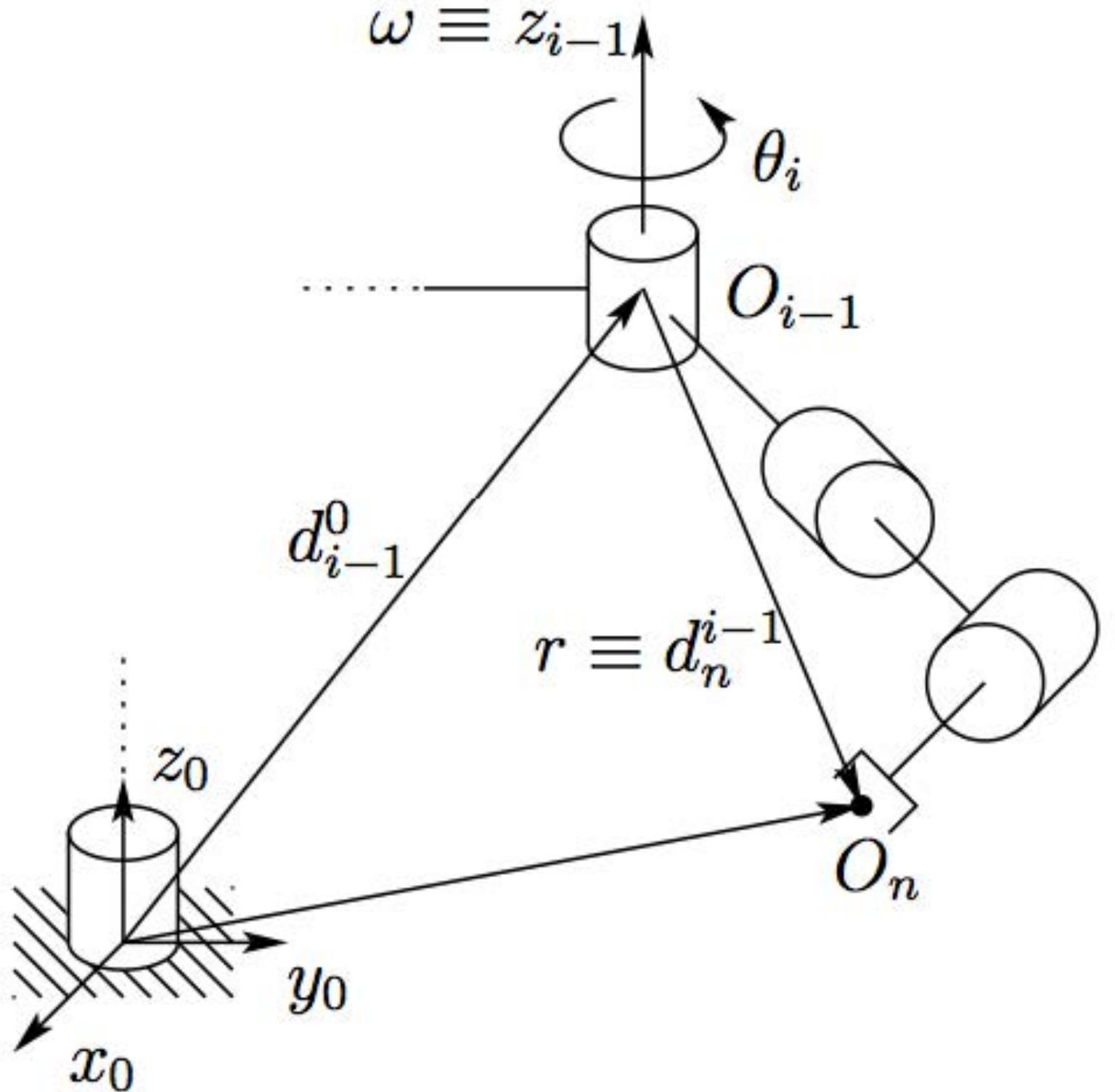
$$J = [J_1 \ J_2 \ \cdots \ J_n]$$

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 ~~J~~Jacobian

A $6 \times N$ matrix

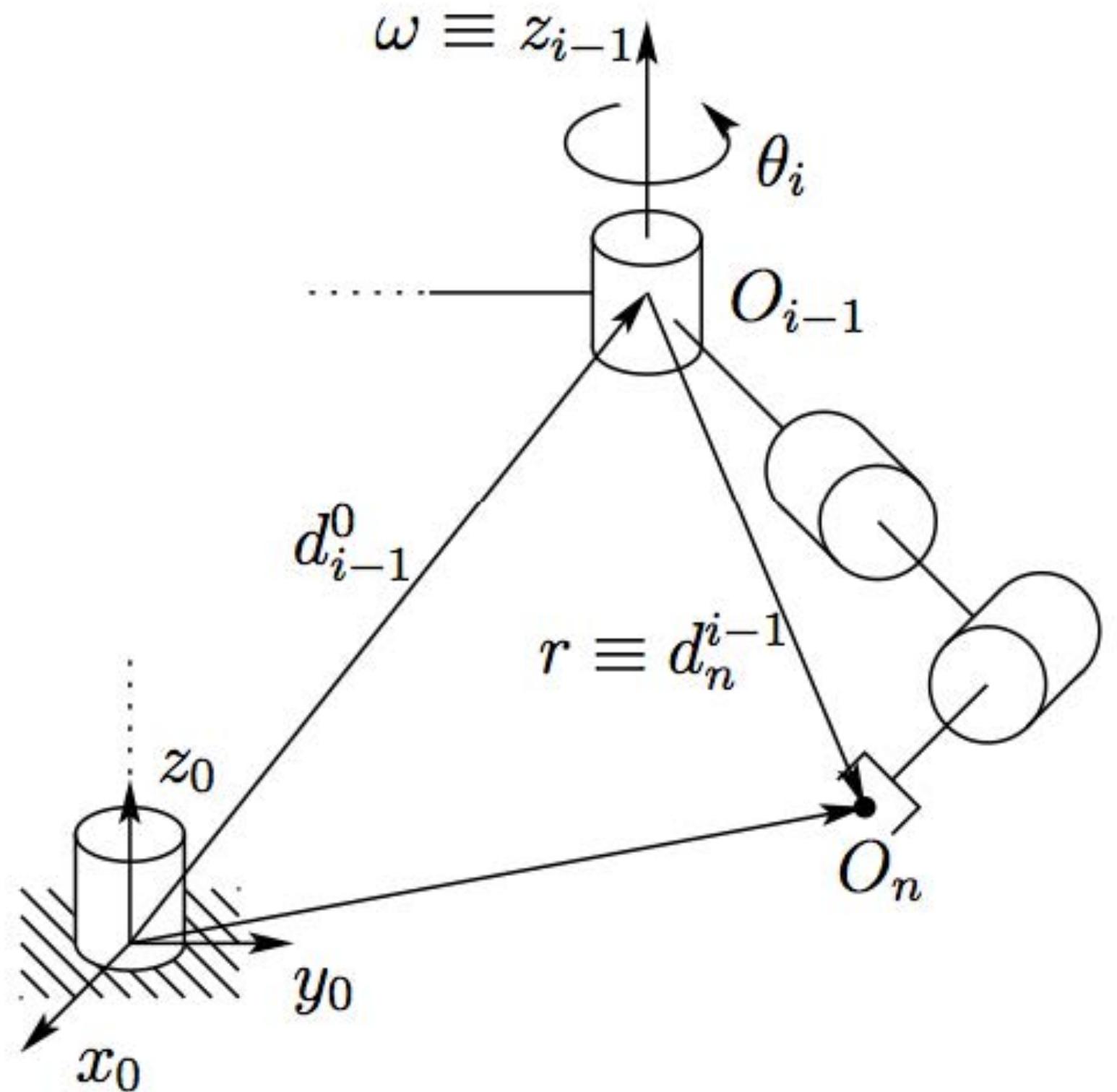
$$J = [J_1 \ J_2 \ \cdots \ J_n]$$

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

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



compute endpoint error

IK Procedure restated:

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

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

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

Geometric The \checkmark Jacobian

A $6 \times N$ matrix

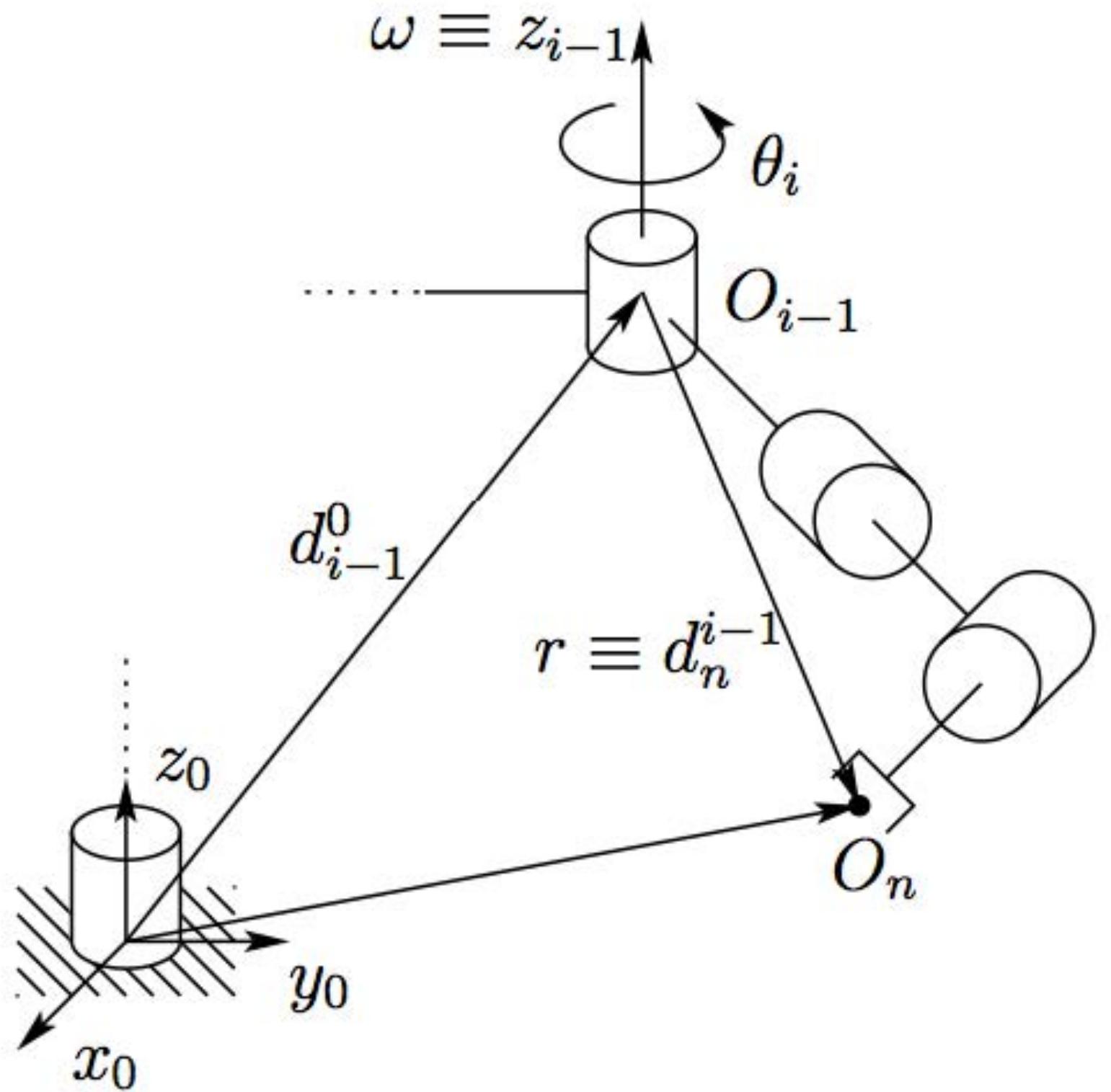
$$J = [J_1 \ J_2 \ \cdots \ J_n]$$

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

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



compute endpoint error

IK Procedure restated:

compute step direction $\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$

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

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

Geometric The \checkmark Jacobian

A $6 \times N$ matrix

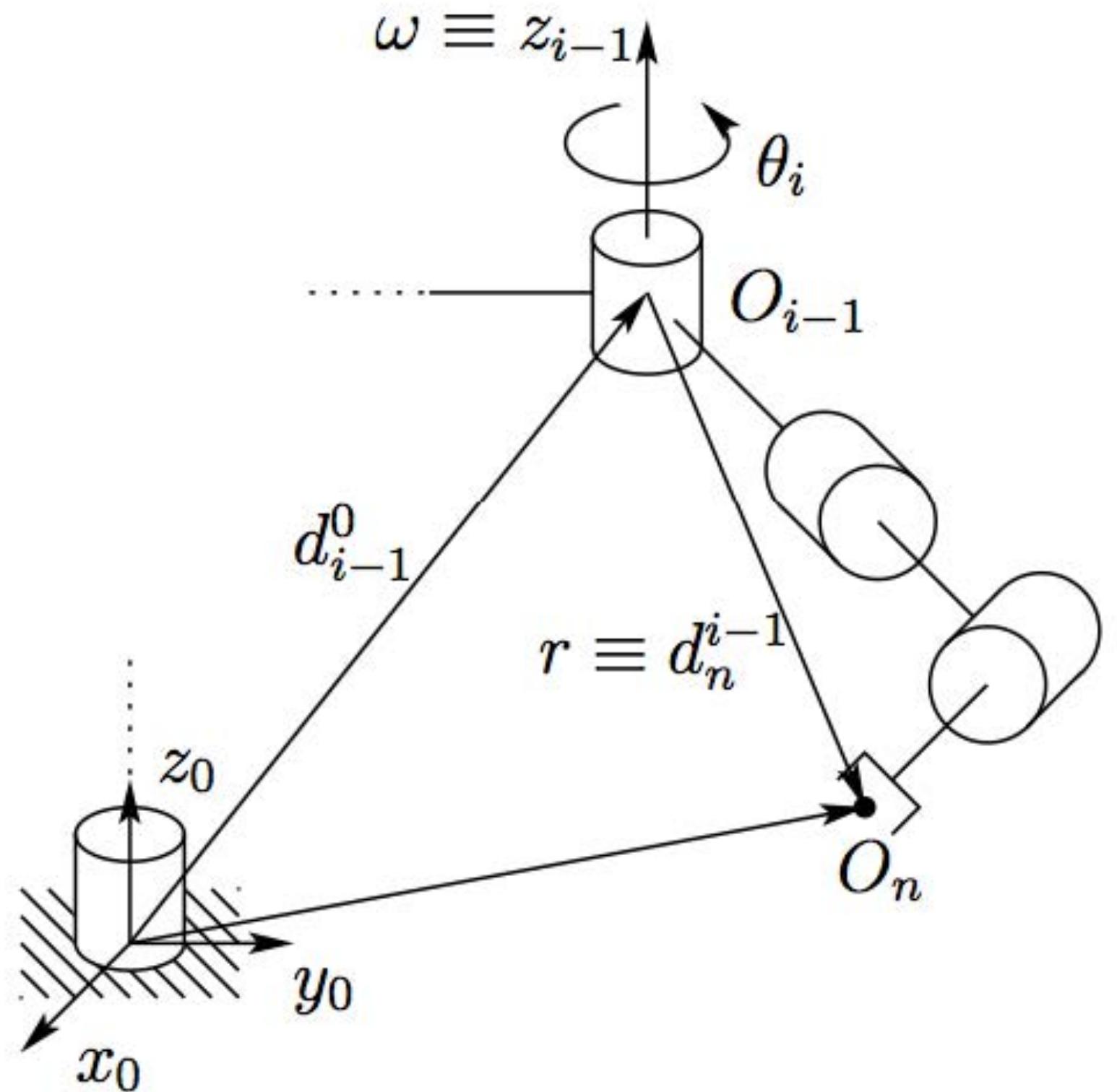
$$J = [J_1 \ J_2 \ \cdots \ J_n]$$

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

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



compute endpoint error

IK Procedure restated:

compute step direction $\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$

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

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

Geometric The Jacobian

A $6 \times N$ matrix

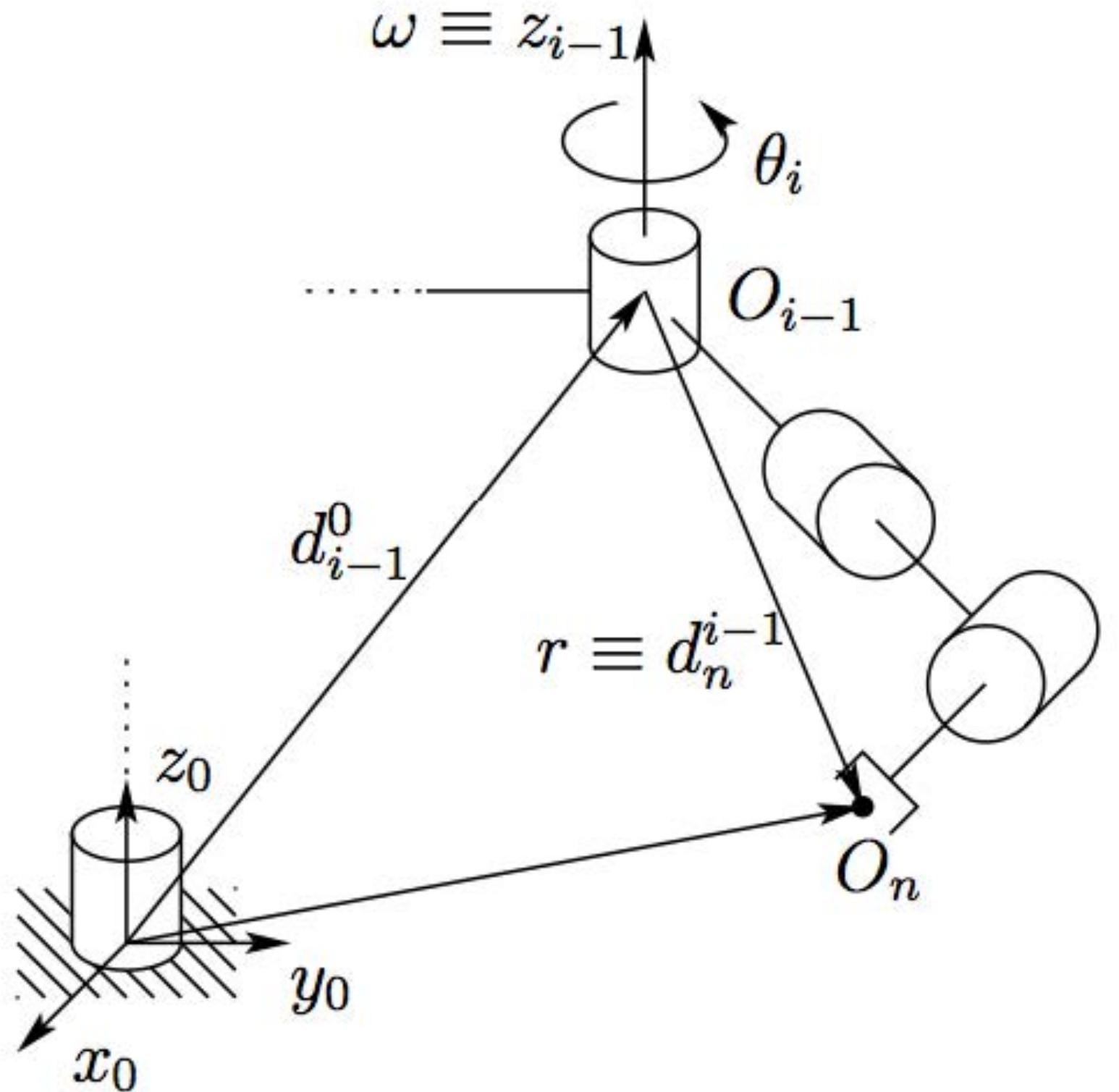
$$J = [J_1 \ J_2 \ \cdots \ J_n]$$

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

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



compute endpoint error

IK Procedure restated:

compute step direction

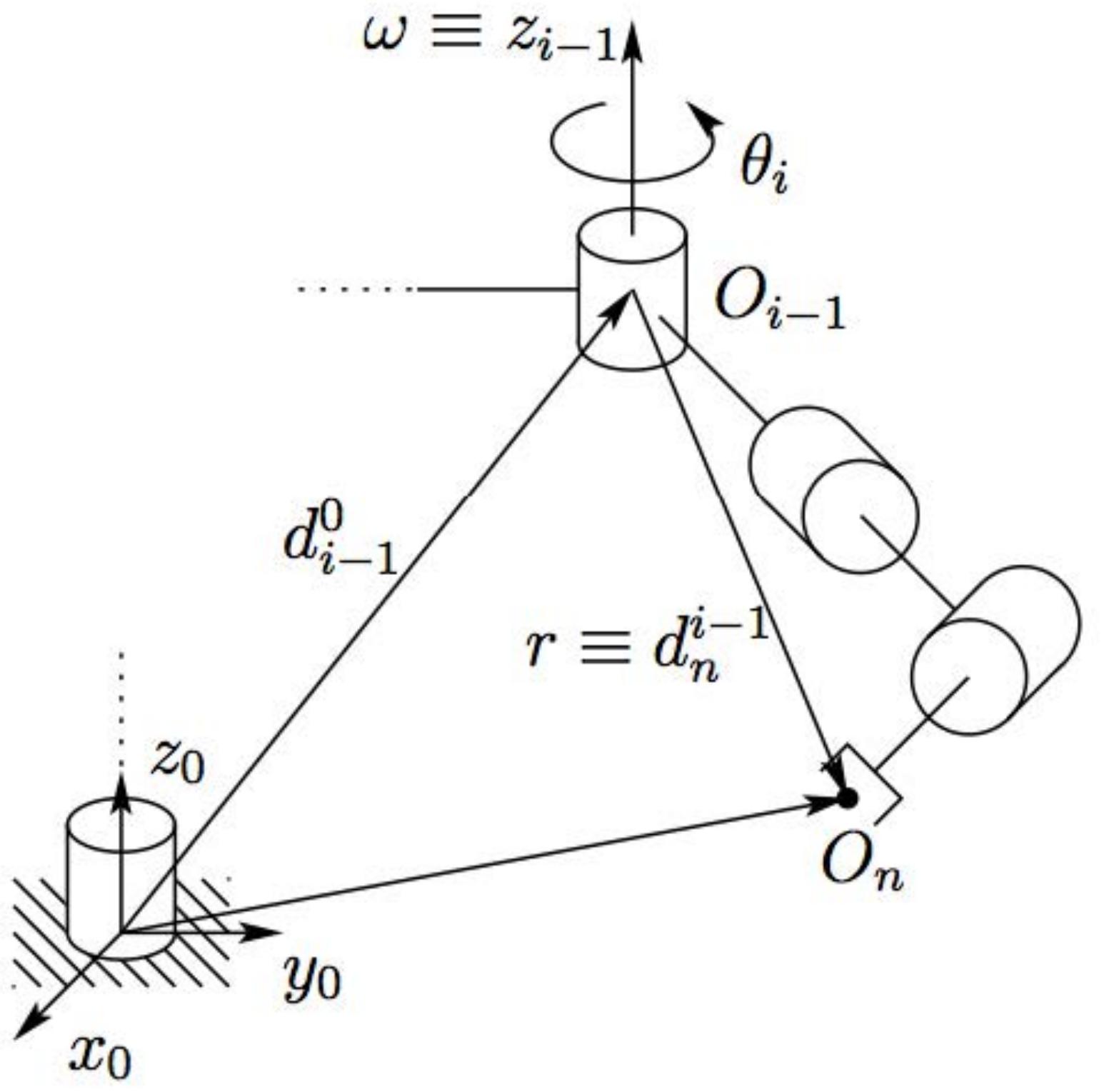
$$\Delta \mathbf{x}_n = \mathbf{x}_d - \mathbf{x}_n$$

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

perform step direction

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

repeat



The Jacobian

A $6 \times N$ matrix

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

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 $J(q)$?

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

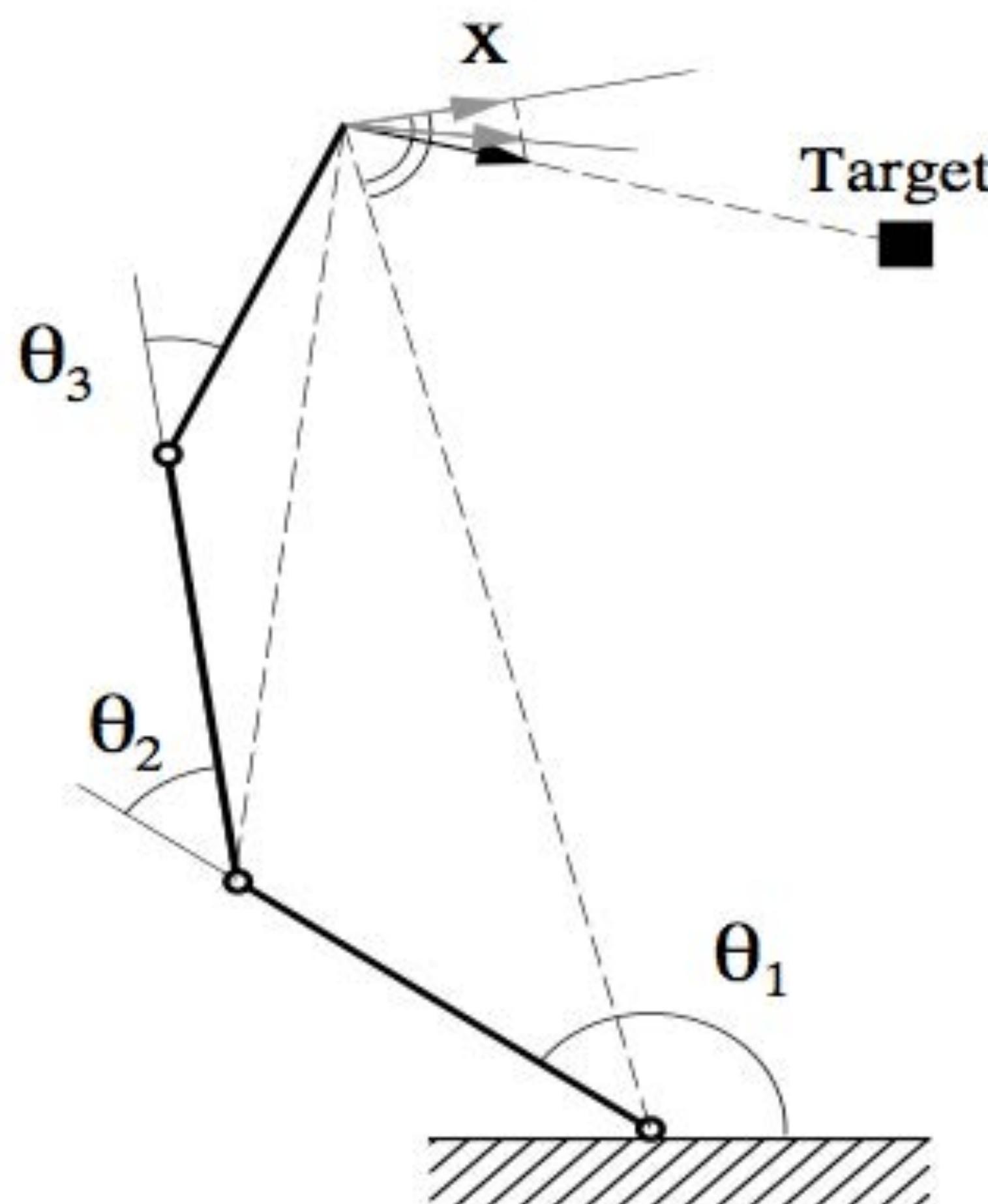
$$\Delta \mathbf{q}_n = J(\mathbf{q}_n)^{-1} \Delta \mathbf{x}_n$$

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

Jacobian Transpose revisited



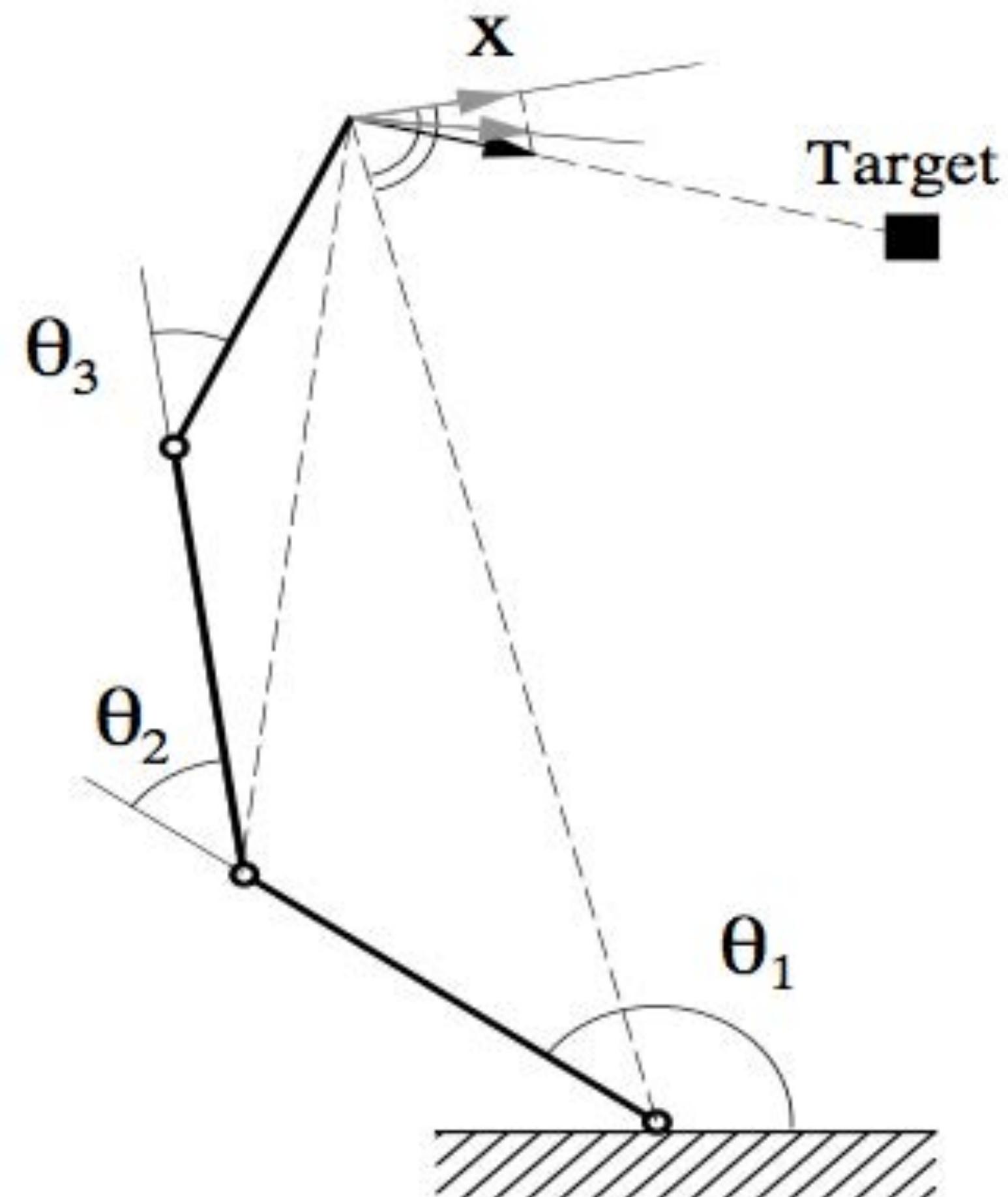
Jacobian Transpose



$$\Delta\theta = \alpha J^T(\theta) \Delta x$$

- Operating Principle:
 - Project difference vector Δx on those dimensions q which can reduce it the most
- Advantages:
 - Simple computation (numerically robust)
 - No matrix inversions
- Disadvantages:
 - Needs many iterations until convergence in certain configurations (e.g., Jacobian has very small coefficients)
 - Unpredictable joint configurations
 - Non conservative

Jacobian Transpose



$$\begin{aligned}\text{Minimize cost function } F &= \frac{1}{2}(\mathbf{x}_{target} - \mathbf{x})^T (\mathbf{x}_{target} - \mathbf{x}) \\ &= \frac{1}{2}(\mathbf{x}_{target} - f(\boldsymbol{\theta}))^T (\mathbf{x}_{target} - f(\boldsymbol{\theta}))\end{aligned}$$

with respect to $\boldsymbol{\theta}$ by gradient descent:

$$\begin{aligned}\Delta\boldsymbol{\theta} &= -\alpha \left(\frac{\partial F}{\partial \boldsymbol{\theta}} \right)^T \\ &= \alpha \left((\mathbf{x}_{target} - \mathbf{x})^T \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)^T \\ &= \alpha J^T(\boldsymbol{\theta})(\mathbf{x}_{target} - \mathbf{x}) \\ &= \alpha J^T(\boldsymbol{\theta})\Delta\mathbf{x}\end{aligned}$$

Error Minimization by Jacobian Transpose

$$\begin{aligned} J(\mathbf{q})\Delta\mathbf{q} &= \Delta\mathbf{x} \\ \Delta\mathbf{q} &= J(\mathbf{q})^{-1}\Delta\mathbf{x} \\ \arg \min_{\Delta\mathbf{q}} &\|J(\mathbf{q})\Delta\mathbf{q} - \Delta\mathbf{x}\|^2 \end{aligned}$$

Jacobian gives mapping from configuration displacement to endeffector displacement

Inverse of Jacobian maps endeffector displacement to configuration displacement

But, inverse of Jacobian is rarely an option. Why?

Instead, find configuration displacement that minimizes endeffector error squared

Error Minimization by Jacobian Transpose

$$\arg \min_{\Delta q} \| J(q) \Delta q - \Delta x \|^2$$

Instead, find configuration displacement that minimizes endeffector error squared

$$\begin{aligned} C &= (J(q) \Delta q - \Delta x)^2 \\ &= (J(q) \Delta q - \Delta x)^T (J(q) \Delta q - \Delta x) \\ &= \Delta q^T J(q)^T J(q) \Delta q - \Delta q^T J(q)^T \Delta x - \Delta x^T J(q) \Delta q + \Delta x^T \Delta x \\ &= \Delta q^T J(q)^T J(q) \Delta q - 2 \Delta q^T J(q)^T \Delta x + \Delta x^T \Delta x \end{aligned}$$

Define cost function expressing squared error

Error Minimization by Jacobian Transpose

$$C = \Delta\mathbf{q}^T J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2\Delta\mathbf{q}^T J(\mathbf{q})^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \Delta\mathbf{x}$$

Define cost function
expressing squared error

$$\frac{dC}{d\Delta\mathbf{q}} = 2J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2J(\mathbf{q})^T \Delta\mathbf{x} + 0$$

Take cost derivative wrt.
change in configuration

$$\left. \frac{dC}{d\Delta\mathbf{q}} \right|_{\Delta\mathbf{q}=0} = 2J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2J(\mathbf{q})^T \Delta\mathbf{x} \Big|_{\Delta\mathbf{q}=0}$$

Evaluate at convergence point, where
change in configuration is zero

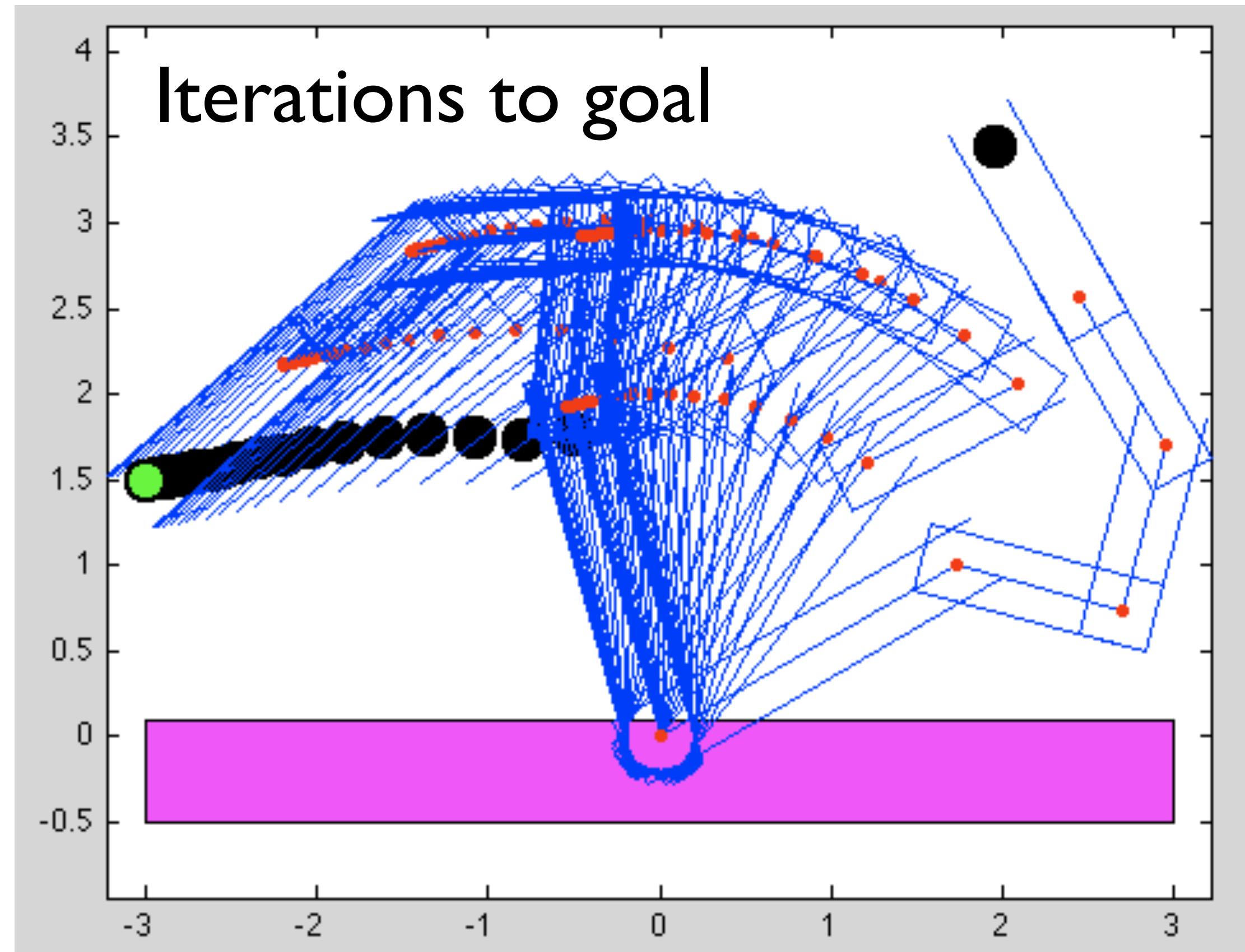
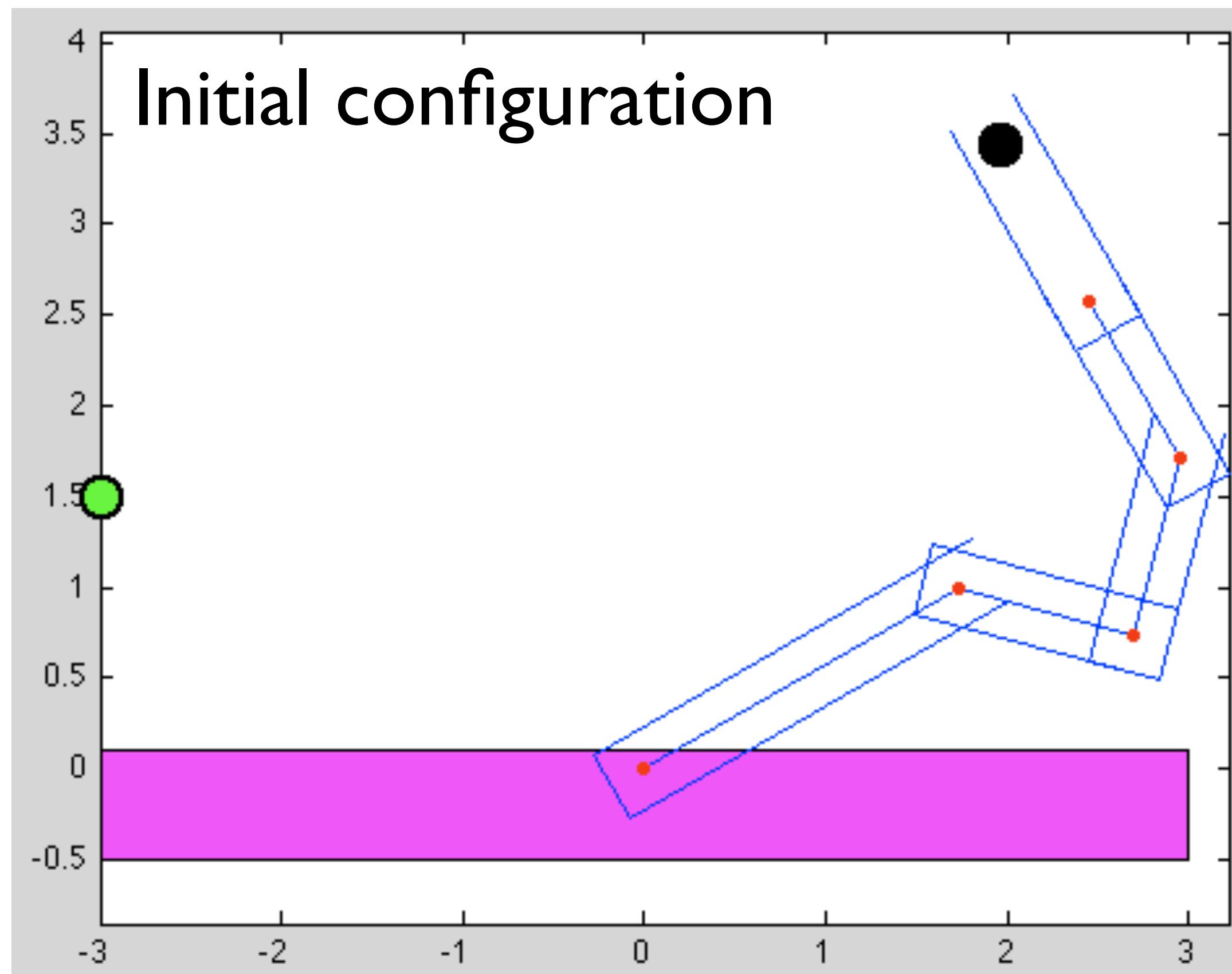
$$= 2J(\mathbf{q})^T \Delta\mathbf{x}$$

$$= \boxed{\gamma J(\mathbf{q})^T \Delta\mathbf{x}}$$

step length (gamma) chosen
as update step scale

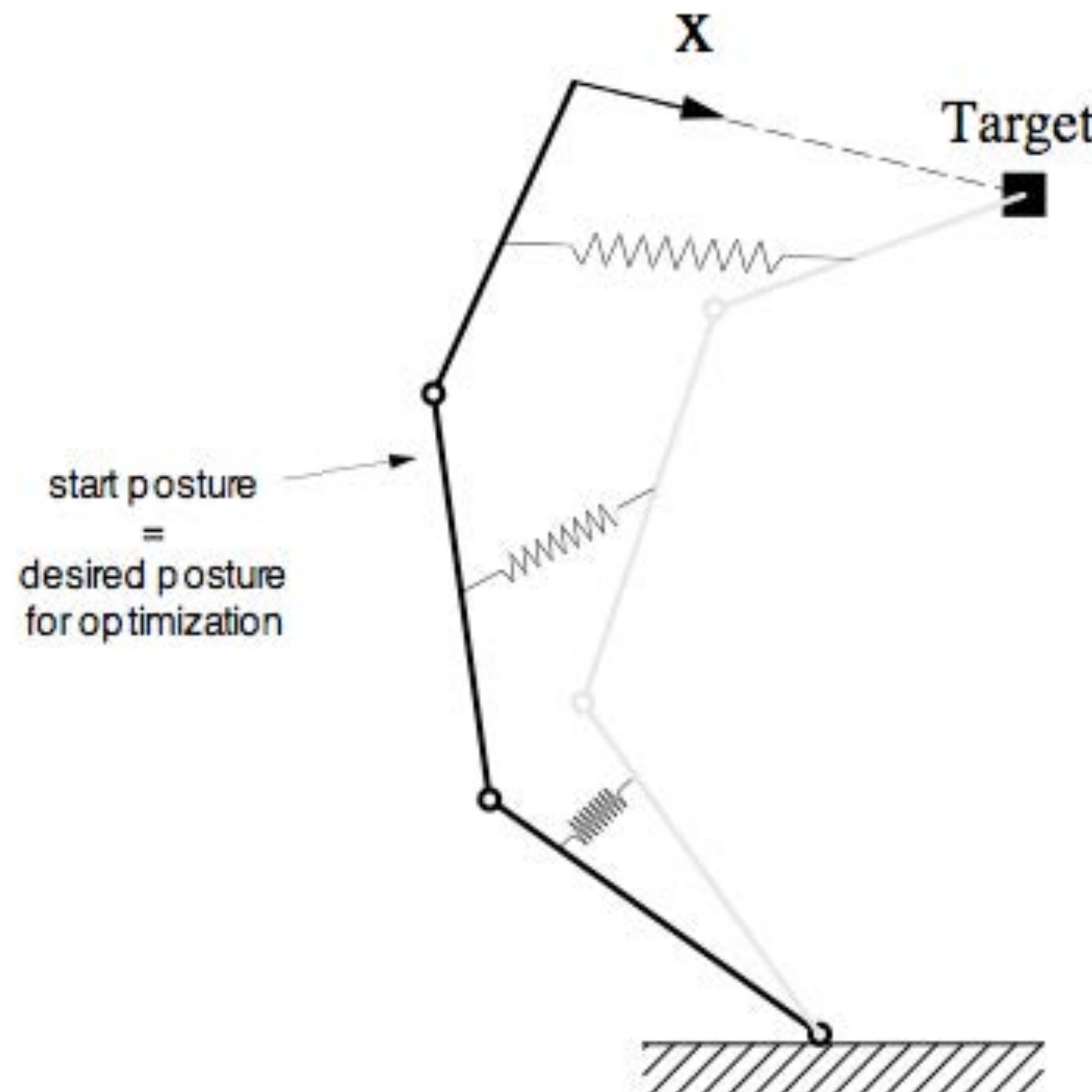


Matlab 5-link arm example: Jacobian transpose



Jacobian Pseudoinverse

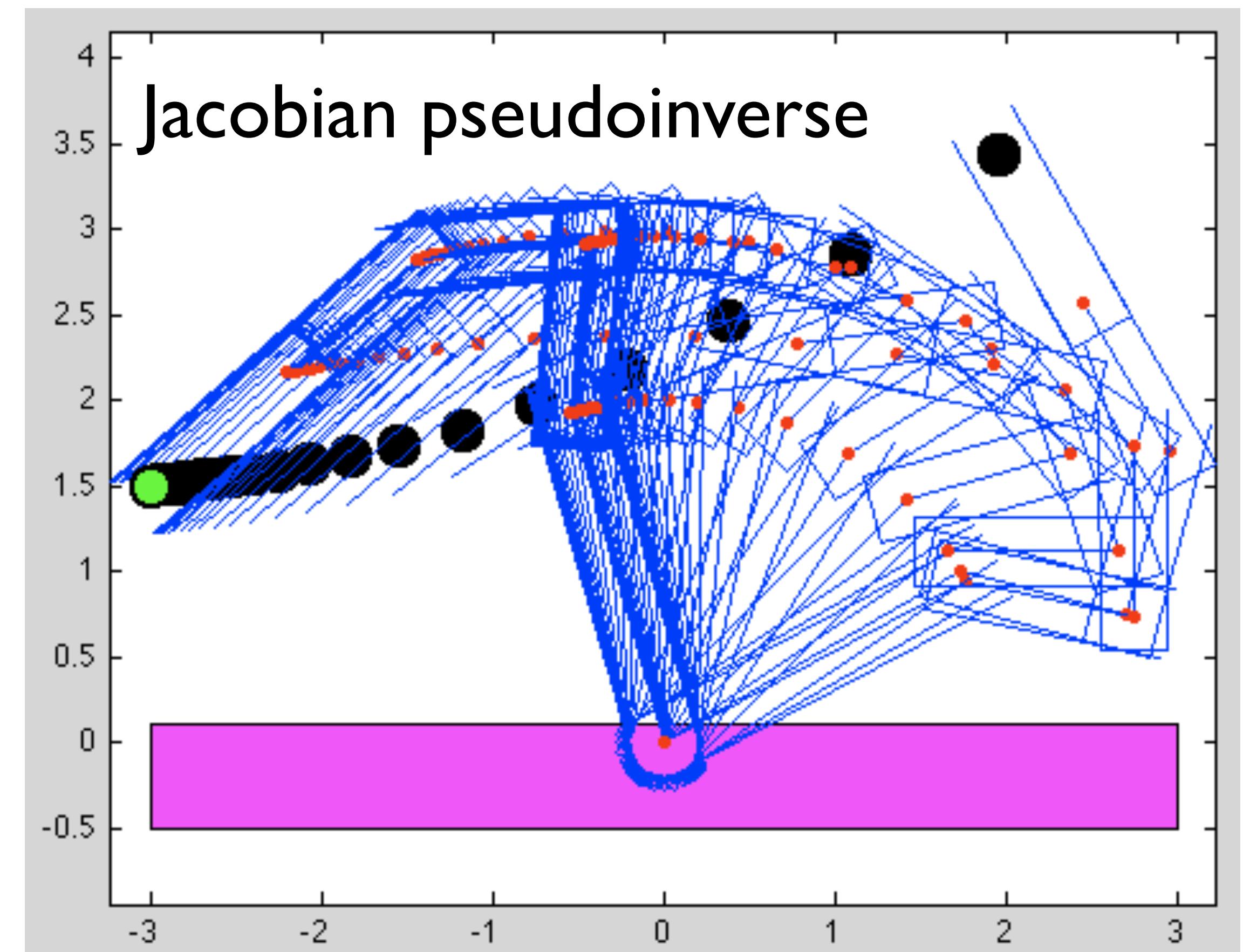
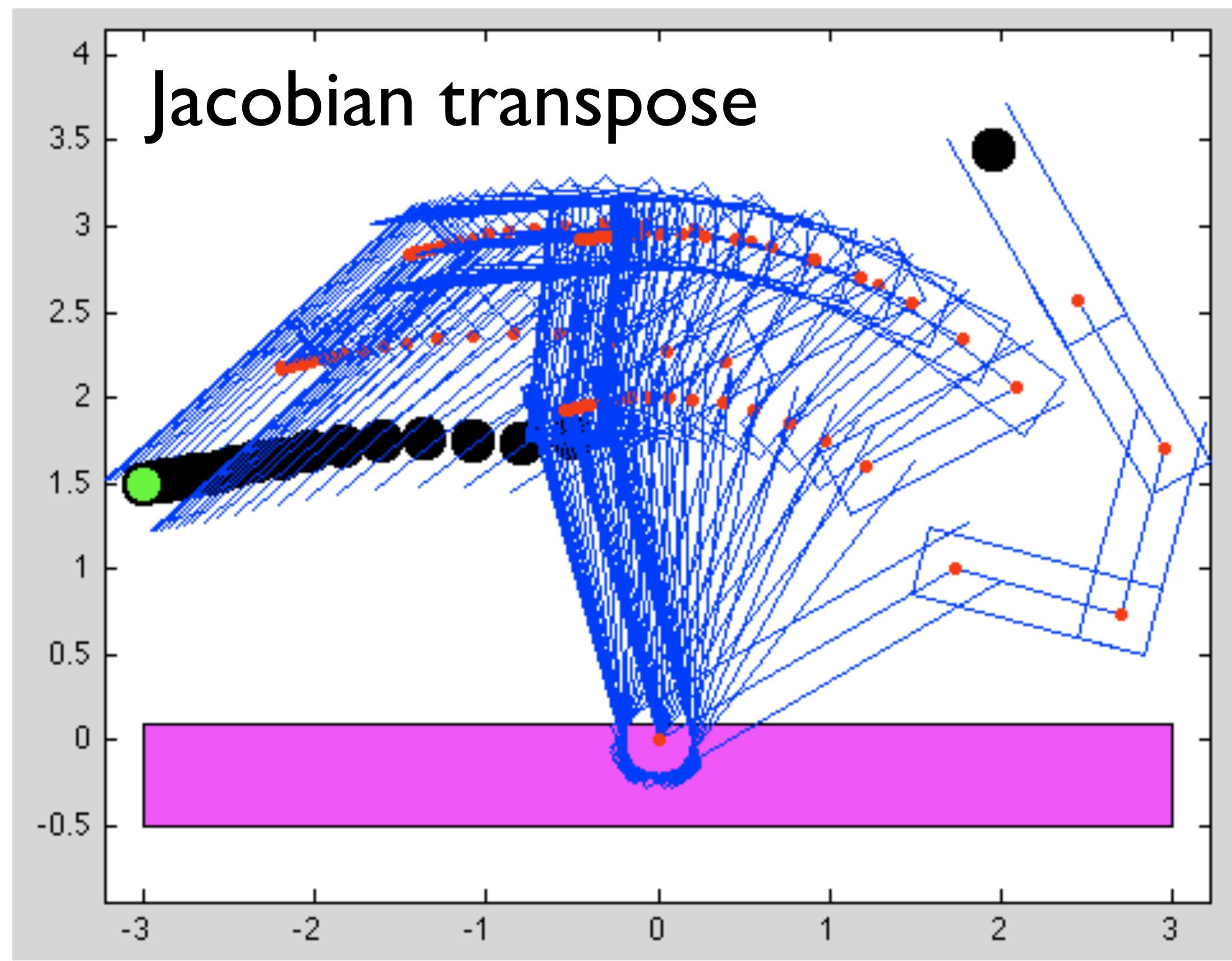
Pseudo Inverse



$$\Delta\theta = \alpha J^T(\theta) (J(\theta)J^T(\theta))^{-1} \Delta x$$

- Operating Principle:
 - Shortest path in q-space
- Advantages:
 - Computationally fast (second order method)
- Disadvantages:
 - Matrix inversion necessary (numerical problems)
 - Unpredictable joint configurations
 - Non conservative

Matlab 5-link arm example: Jacobian Pseudoinverse



Error Minimization by Jacobian Pseudoinverse

Define cost function
expressing squared error

$$C = \Delta\mathbf{q}^T J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2\Delta\mathbf{q}^T J(\mathbf{q})^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \Delta\mathbf{x}$$

$$\frac{dC}{d\Delta\mathbf{q}} = 2J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2J(\mathbf{q})^T \Delta\mathbf{x} + 0$$

Take cost derivative

Set to zero and solve for configuration displacement

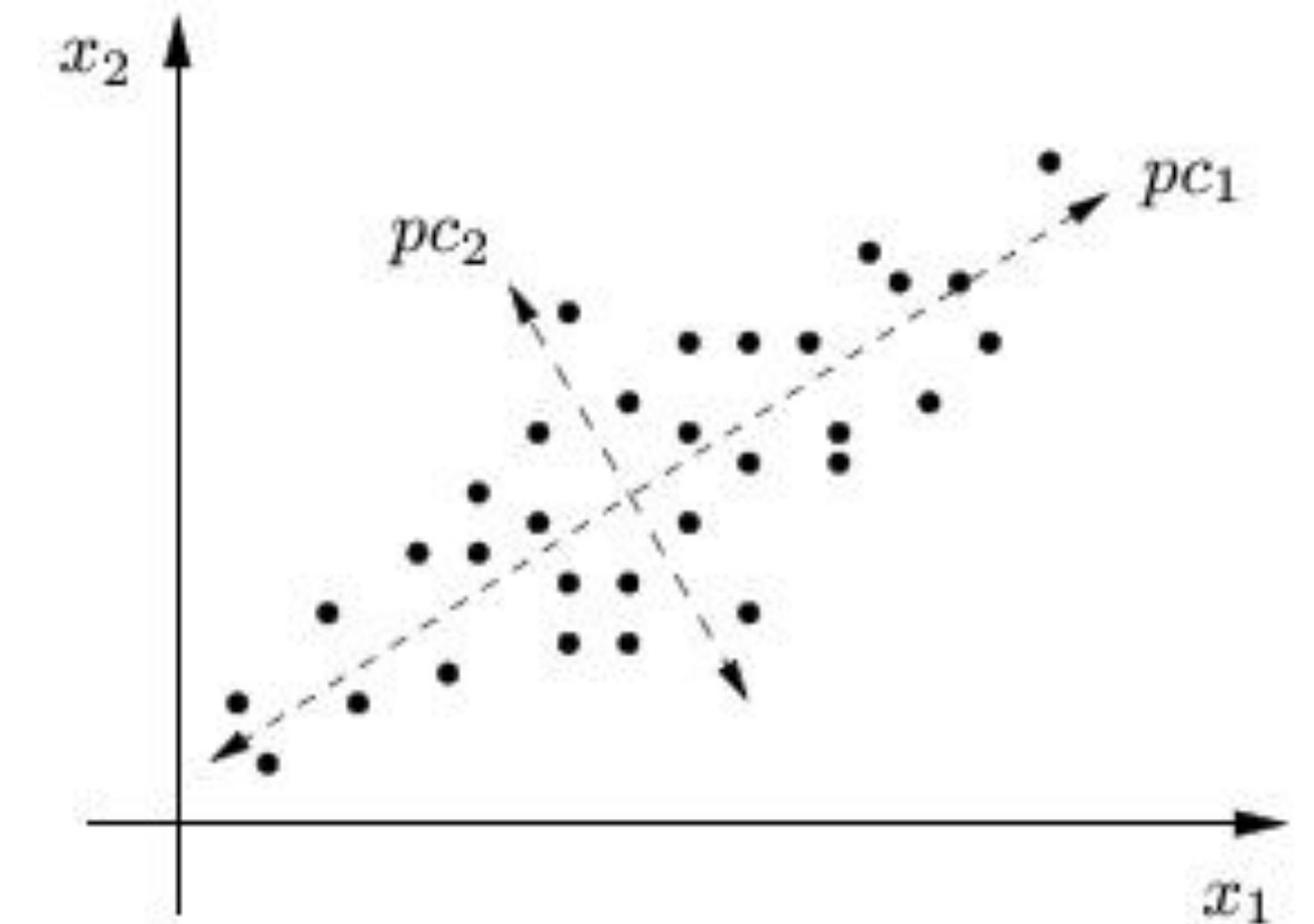
$$0 = 2J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} - 2J(\mathbf{q})^T \Delta\mathbf{x}$$

$$J(\mathbf{q})^T J(\mathbf{q}) \Delta\mathbf{q} = J(\mathbf{q})^T \Delta\mathbf{x} \quad \text{Normal form}$$

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



Pseudoinverse, More Generally



- Pseudoinverse of matrix A : $A^+ = (A^T A)^{-1} A^T$ approximates solution to linear system $Ax=b$
- The pseudoinverse A^+ is a least squares “best fit” approximate solution of an overdetermined system $Ax=b$, where there are more equations (m) than unknowns (n), or vice versa
- Often used for data fitting, as a singular value decomposition

Which Pseudoinverse

- For matrix A with dimensions $N \times M$ with full rank
- Left pseudoinverse, for when $N > M$, (i.e., “tall”, less than than 6 DoFs)

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

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

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



Maybe there is a simpler
approach to IK?

Next lecture:
IK continued ...
Manipulation New Frontiers