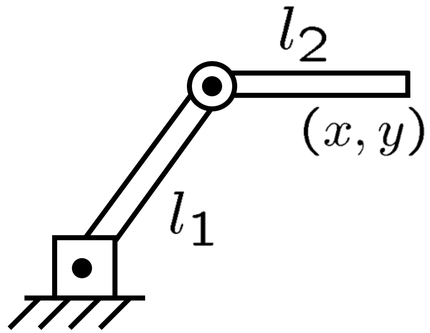
Recap

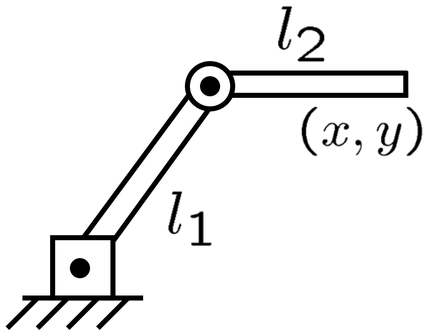
Forward kinematics

Forward kinematics



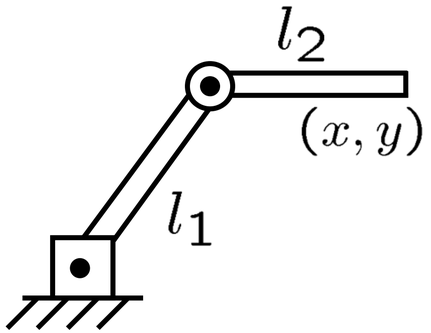
Forward kinematics

$$\mathbf{x} = (x, y)^T$$
$$\mathbf{q} = (\theta_1, \theta_2)^T$$



Forward kinematics

$$\mathbf{x} = (x, y)^T$$
$$\mathbf{q} = (\theta_1, \theta_2)^T$$

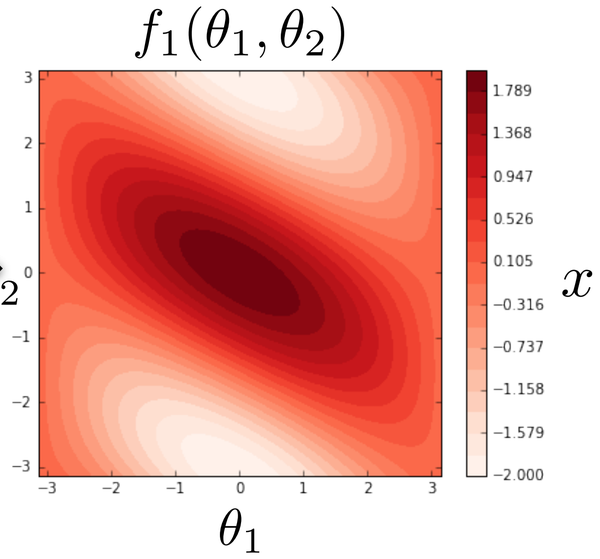


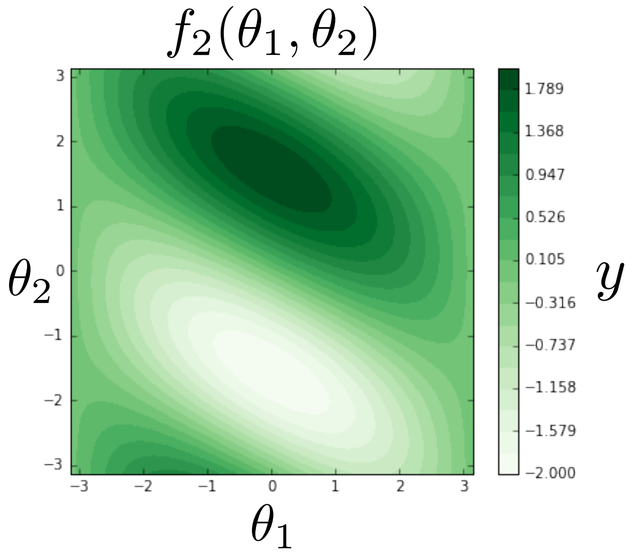
Nonlinear System !!!

Forward kinematics

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

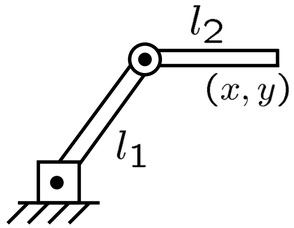
$$x = f_1(\mathbf{q}) = l_1 c_1 + l_2 c_{12}$$
$$y = f_2(\mathbf{q}) = l_1 s_1 + l_2 s_{12}$$





Inverse kinematics

Inverse kinematics



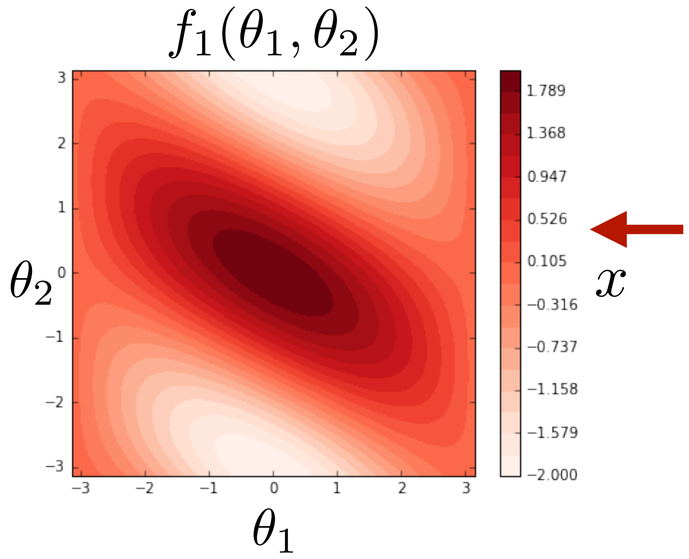
Forward kinematics

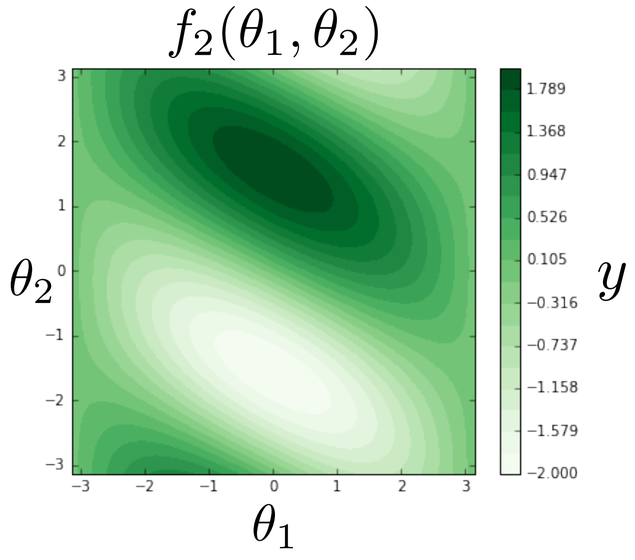
$$\mathbf{x} = f(\mathbf{q})$$

Inverse kinematics

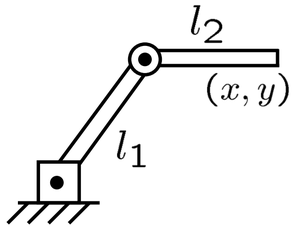
$$\mathbf{q} = f^{-1}(\mathbf{x})$$

Forward kinematics





Inverse kinematics



Forward kinematics

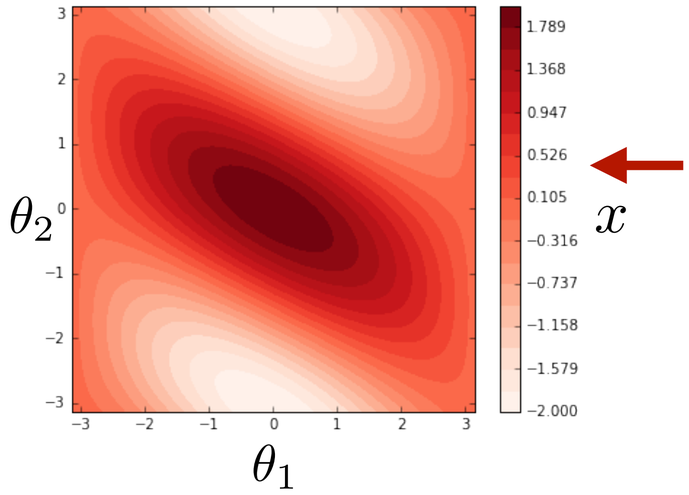
$$\mathbf{x} = f(\mathbf{q})$$

Inverse kinematics

$$\mathbf{q} = f^{-1}(\mathbf{x})$$

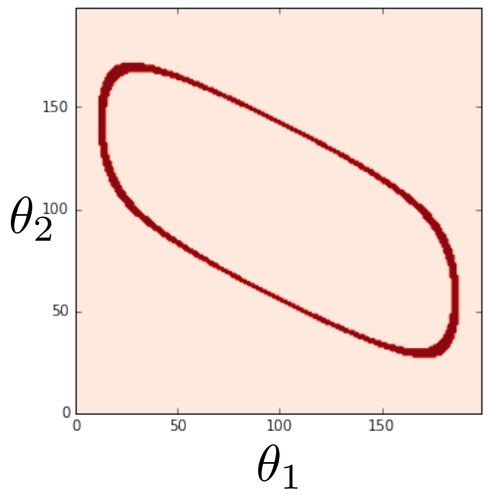
Forward kinematics

$f_1(\theta_1, \theta_2)$

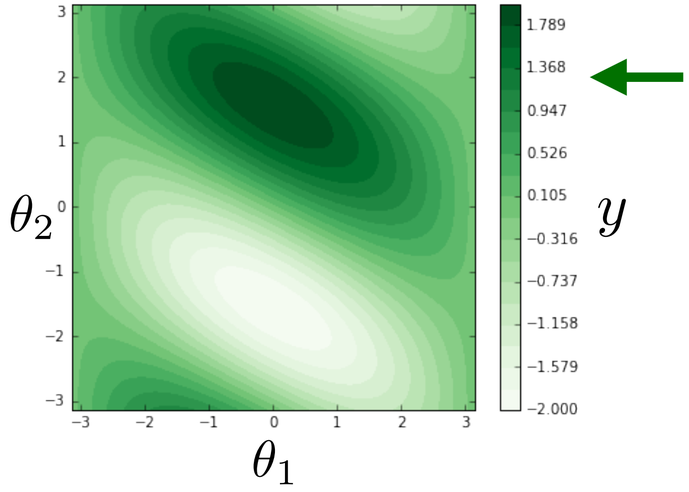


Inverse kinematics

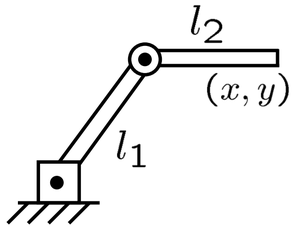
$(\theta_1, \theta_2)^T = f_1^{-1}(0.4)$



$f_2(\theta_1, \theta_2)$



Inverse kinematics



Forward kinematics

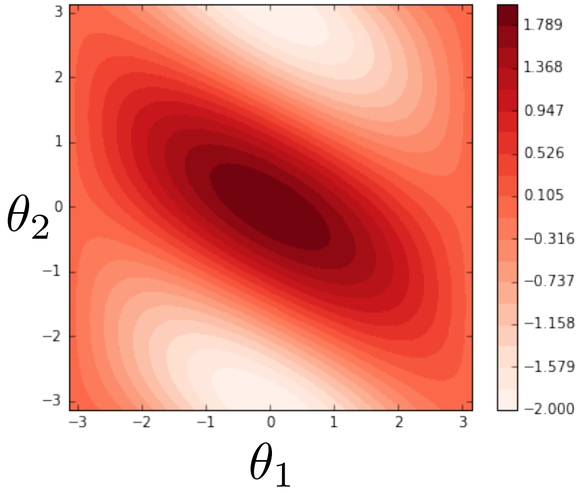
$$\mathbf{x} = f(\mathbf{q})$$

Inverse kinematics

$$\mathbf{q} = f^{-1}(\mathbf{x})$$

Forward kinematics

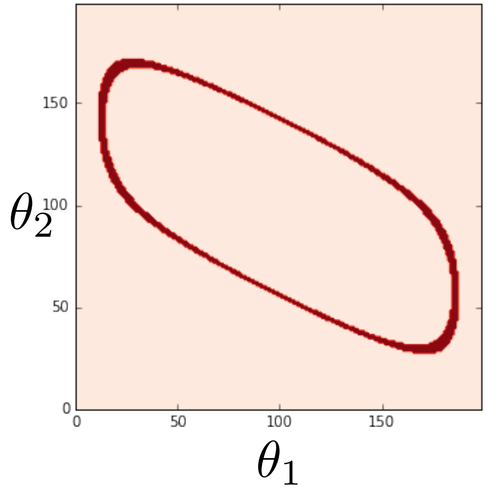
$f_1(\theta_1, \theta_2)$



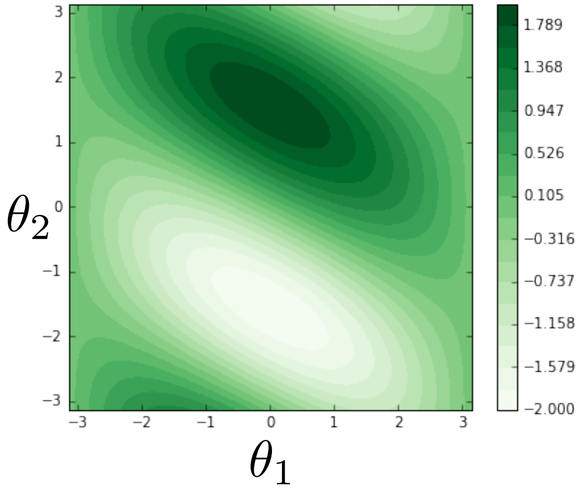
x

Inverse kinematics

$(\theta_1, \theta_2)^T = f_1^{-1}(0.4)$

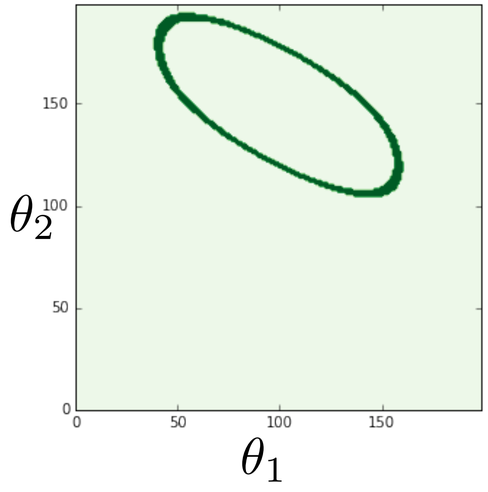


$f_2(\theta_1, \theta_2)$

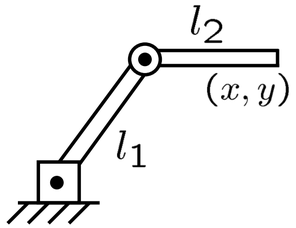


y

$(\theta_1, \theta_2)^T = f_2^{-1}(1.2)$



Inverse kinematics



Forward kinematics

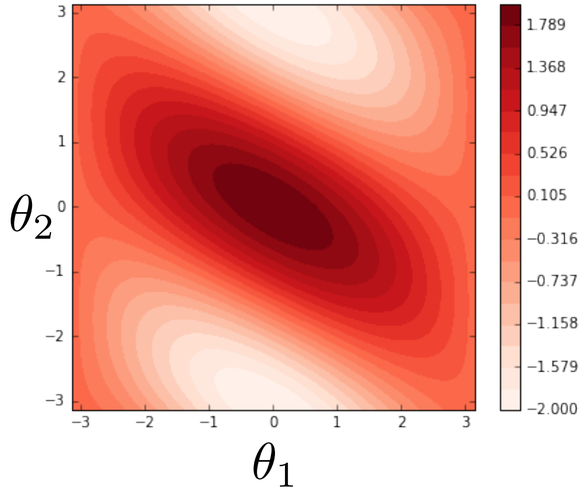
$$\mathbf{x} = f(\mathbf{q})$$

Inverse kinematics

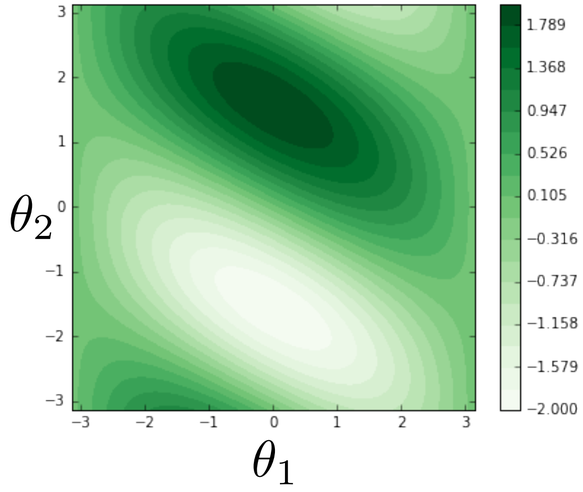
$$\mathbf{q} = f^{-1}(\mathbf{x})$$

Forward kinematics

$f_1(\theta_1, \theta_2)$

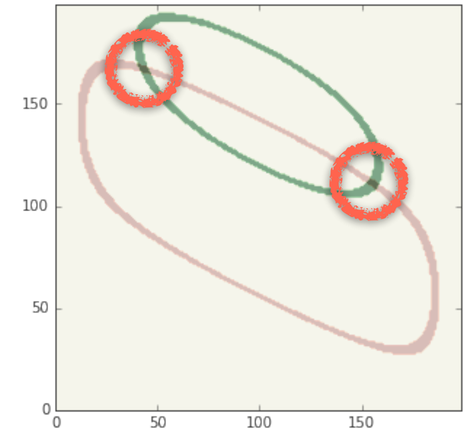


$f_2(\theta_1, \theta_2)$



Inverse kinematics

$$(\theta_1, \theta_2)^T = f_1^{-1}(0.4)$$
$$(\theta_1, \theta_2)^T = f_2^{-1}(1.2)$$



Inverse kinematics

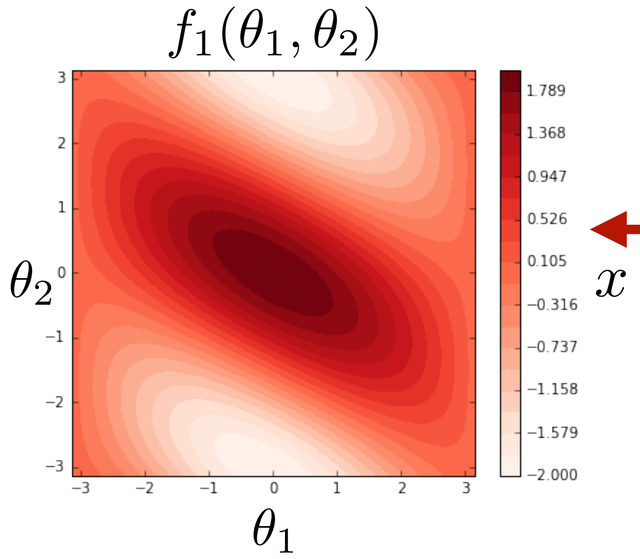
Forward kinematics

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

Inverse kinematics

$$\mathbf{q} = f^{-1}(\mathbf{x})$$

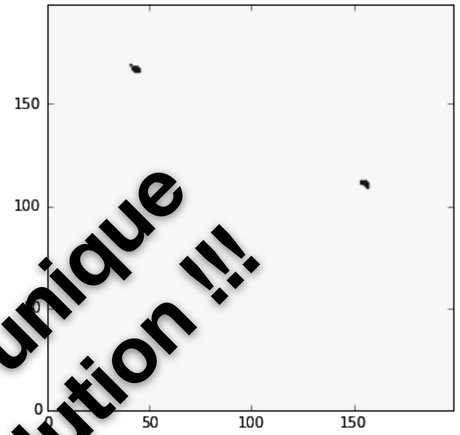
Forward kinematics



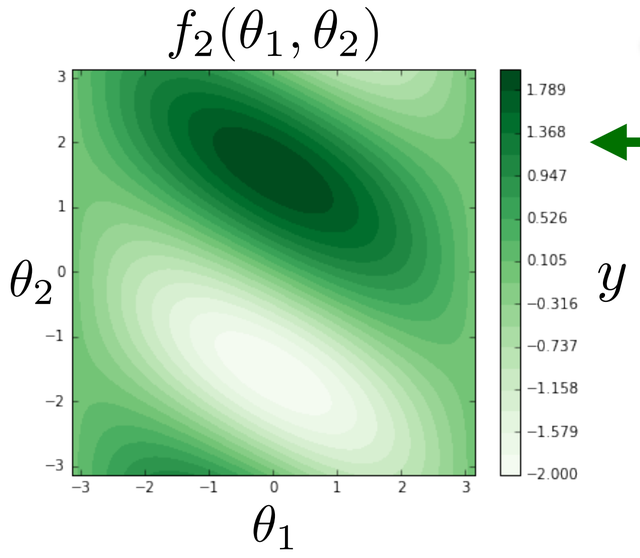
x

Inverse kinematics

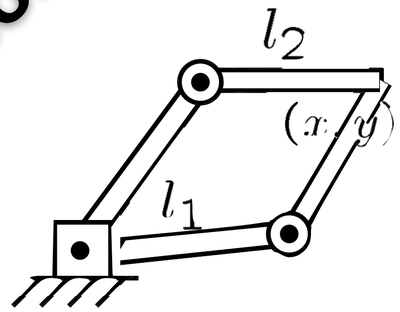
$$(\theta_1, \theta_2)^T = f^{-1}((0.4, 1.2)^T)$$



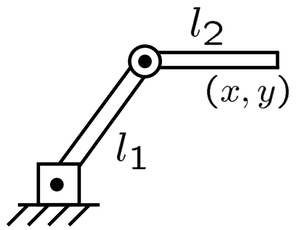
No unique solution !!!



y



Inverse kinematics



Forward kinematics

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

$$x = f_1(\mathbf{q}) = l_1 c_1 + l_2 c_{12}$$

$$y = f_2(\mathbf{q}) = l_1 s_1 + l_2 s_{12}$$

Inverse kinematics

$$\mathbf{q} = f^{-1}(\mathbf{x})$$

Analytical solution:

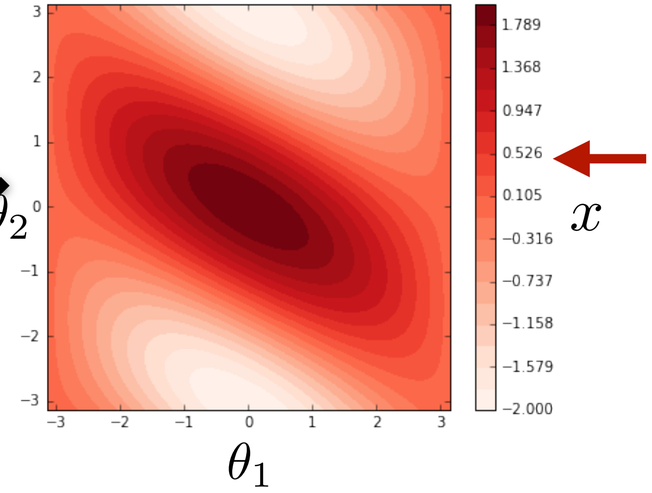
$$\theta_1 = \begin{matrix} \pm\cos^{-1}\left(\frac{\sqrt{-y^2\sin^2(x)+2\sin^2(x)+2\sin^2(x)\cos(x)}+y\cos(x)+y}{\sin^2(x)+\cos^2(x)+2\cos(x)+1}\right) \\ \pm\cos^{-1}\left(\frac{-\sqrt{-y^2\sin^2(x)+2\sin^2(x)+2\sin^2(x)\cos(x)}+y\cos(x)+y}{\sin^2(x)+\cos^2(x)+2\cos(x)+1}\right) \end{matrix}$$

$$\theta_2 = \begin{matrix} \pm\cos^{-1}\left(\frac{y\sin(x)-2\sqrt{-y^2\cos^4\left(\frac{x}{2}\right)+2\cos(x)\cos^4\left(\frac{x}{2}\right)+2\cos^4\left(\frac{x}{2}\right)}}{\sin^2(x)+\cos^2(x)+2\cos(x)+1}\right) \\ \pm\cos^{-1}\left(\frac{y\sin(x)-2\sqrt{-y^2\cos^4\left(\frac{x}{2}\right)+2\cos(x)\cos^4\left(\frac{x}{2}\right)+2\cos^4\left(\frac{x}{2}\right)}}{\sin^2(x)+\cos^2(x)+2\cos(x)+1}\right) \end{matrix}$$

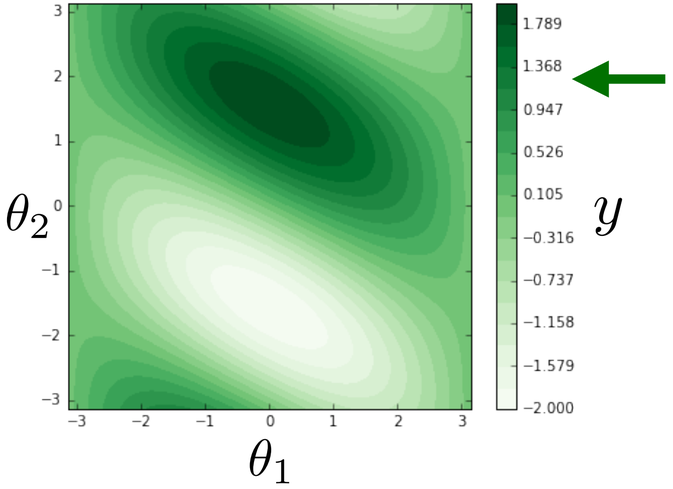
Super complicated !!!

Forward kinematics

$f_1(\theta_1, \theta_2)$



$f_2(\theta_1, \theta_2)$

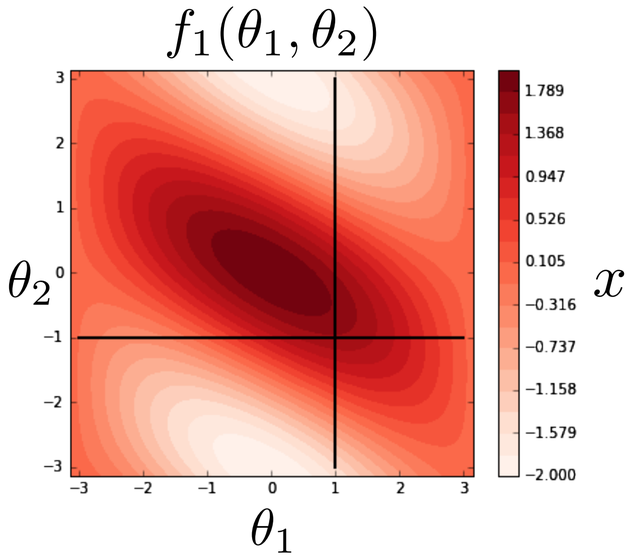


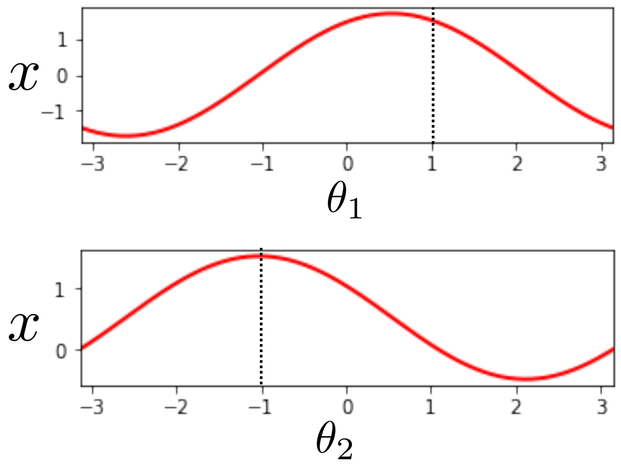
Help please!

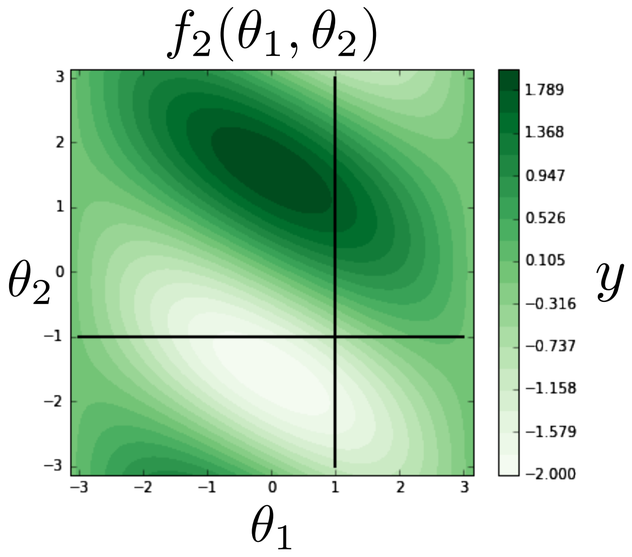
Jacobian

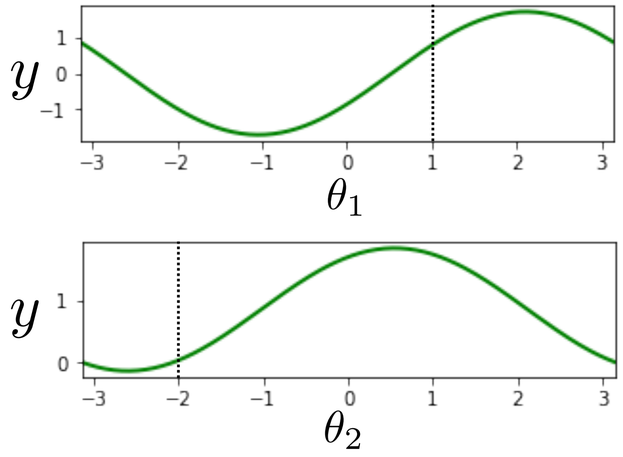
Jacobian

Forward kinematics



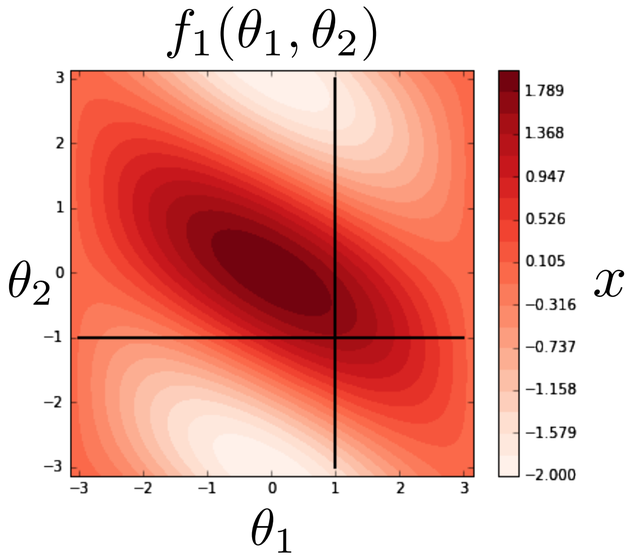


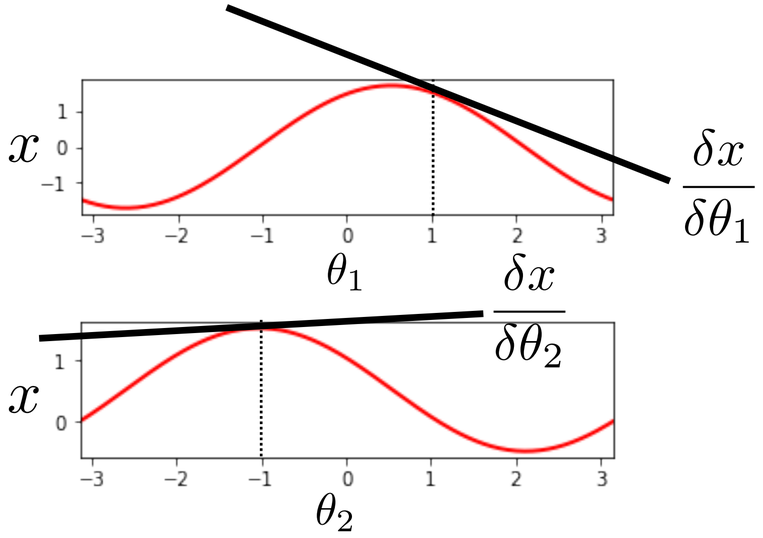


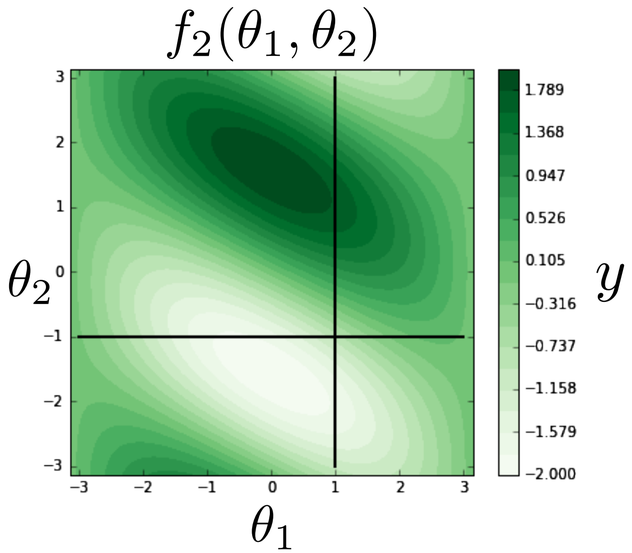


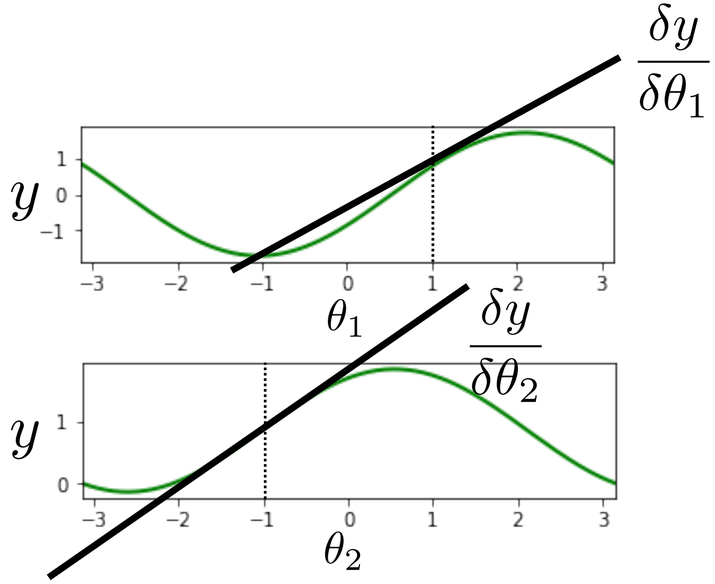
Jacobian

Forward kinematics



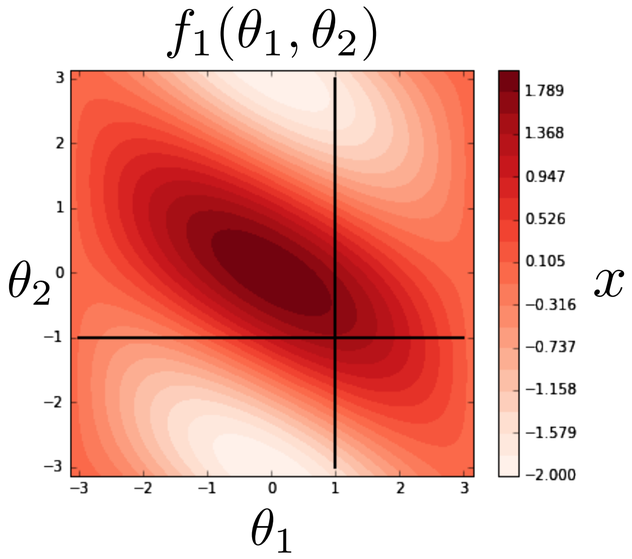


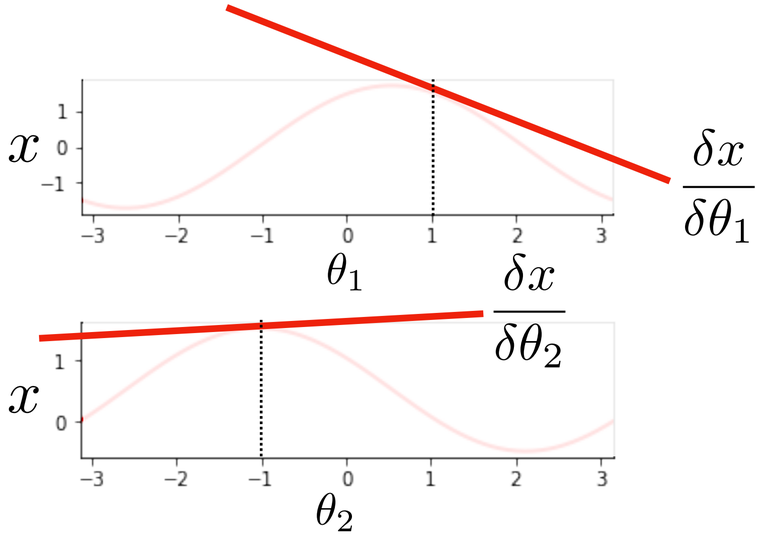


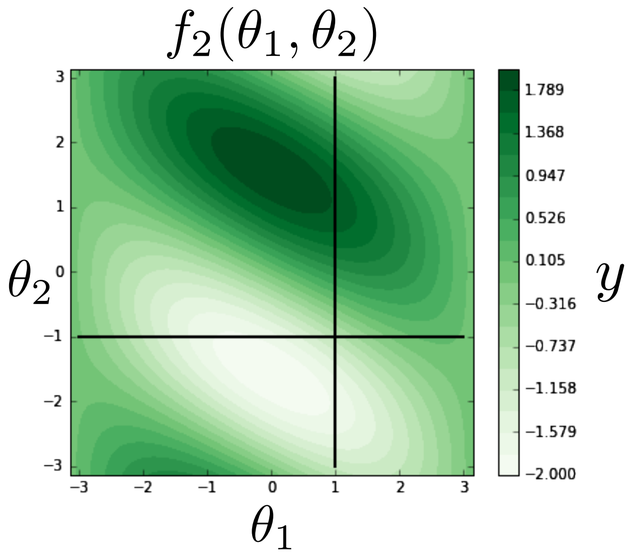


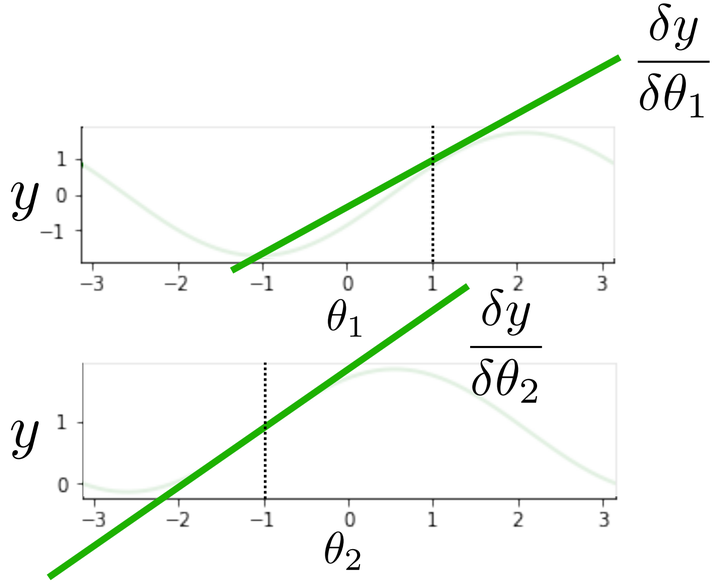
Jacobian

Forward kinematics



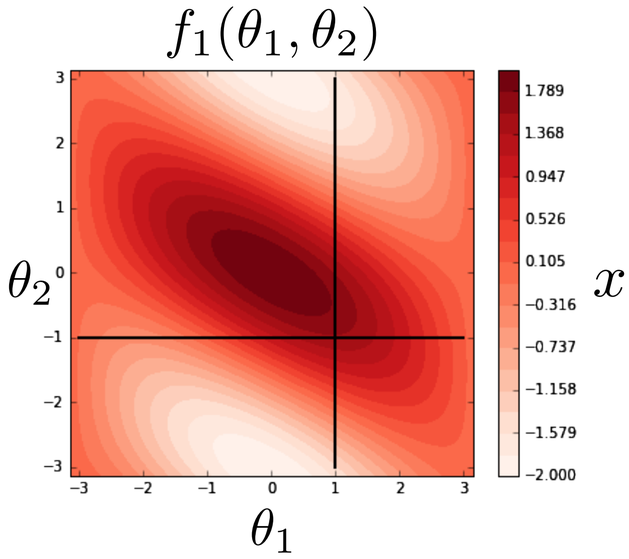


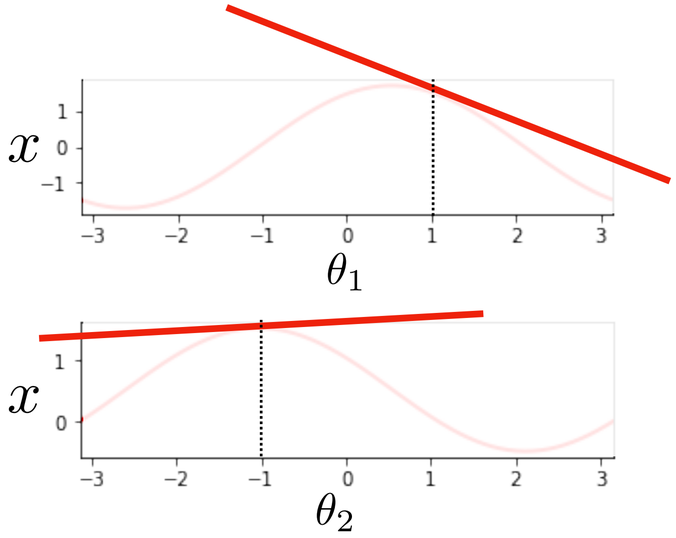




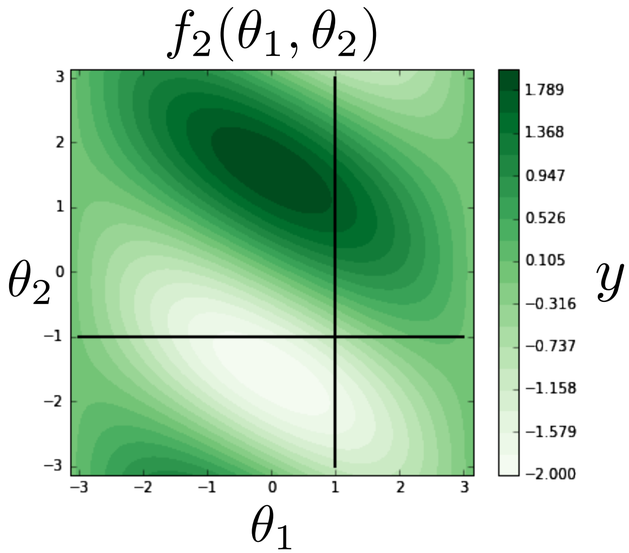
Jacobian

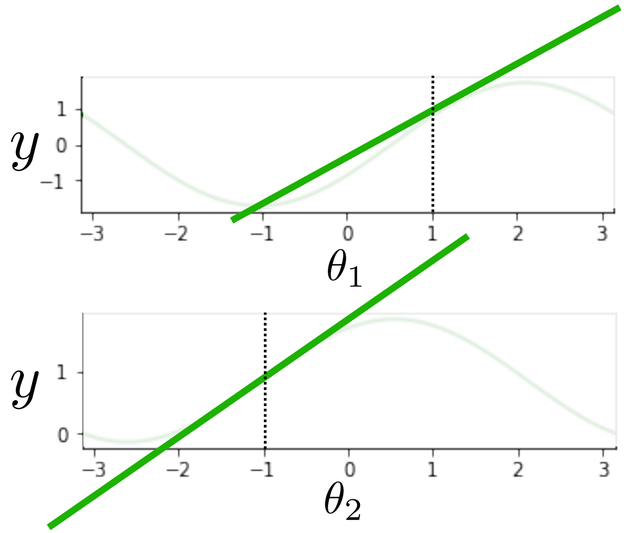
Forward kinematics





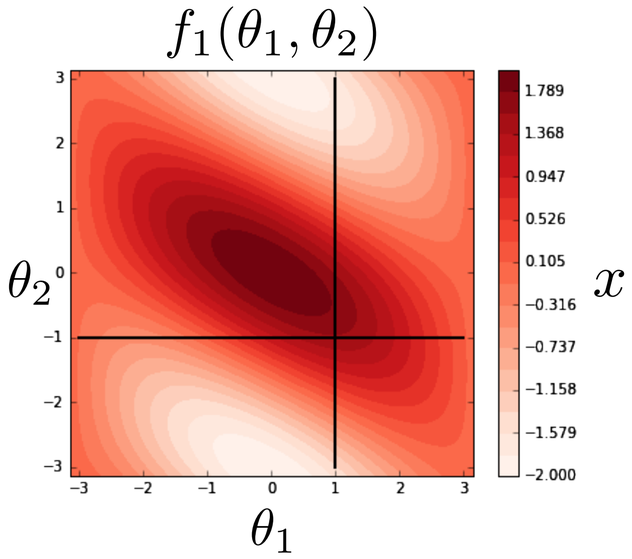
$$J = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix}$$

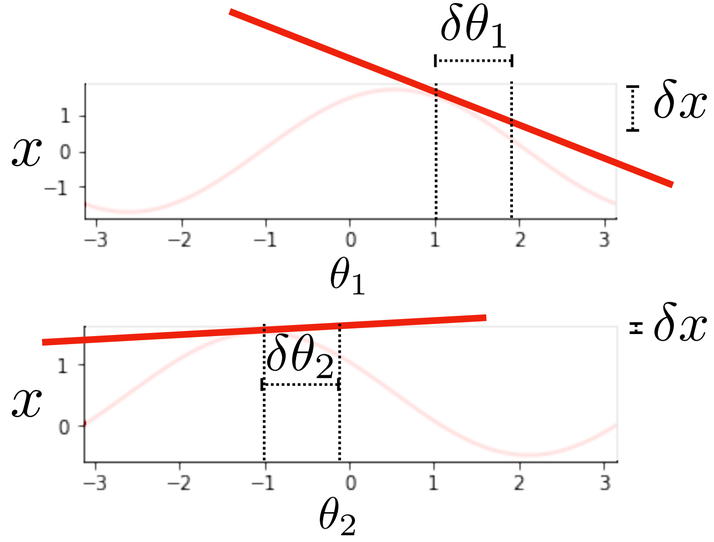




Jacobian

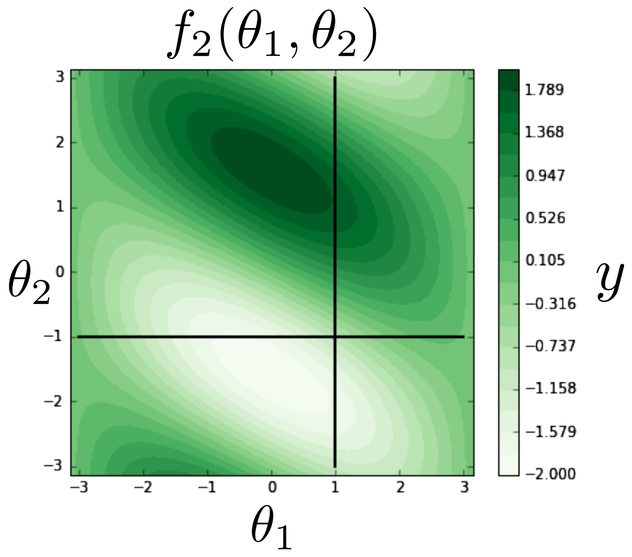
Forward kinematics

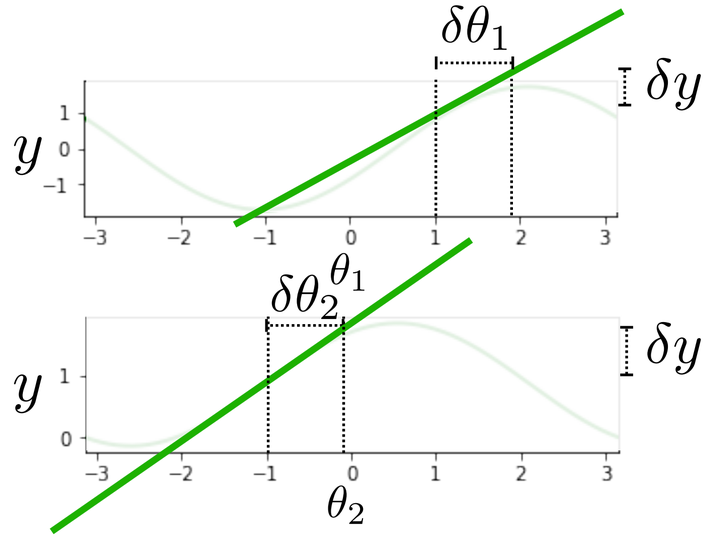




$$J$$

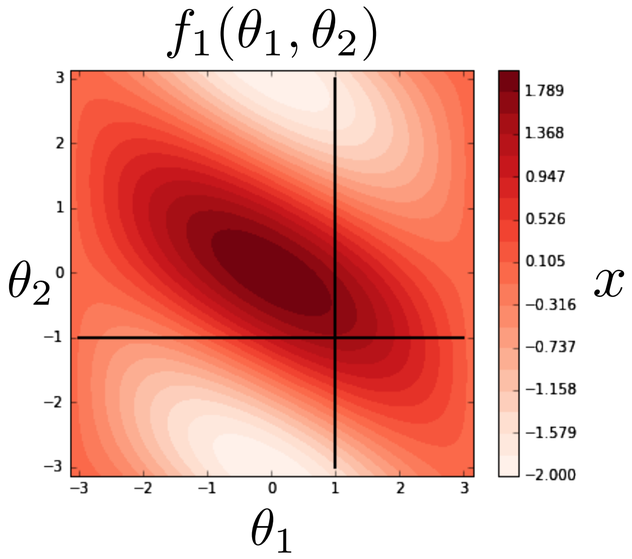
$$\begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix}$$

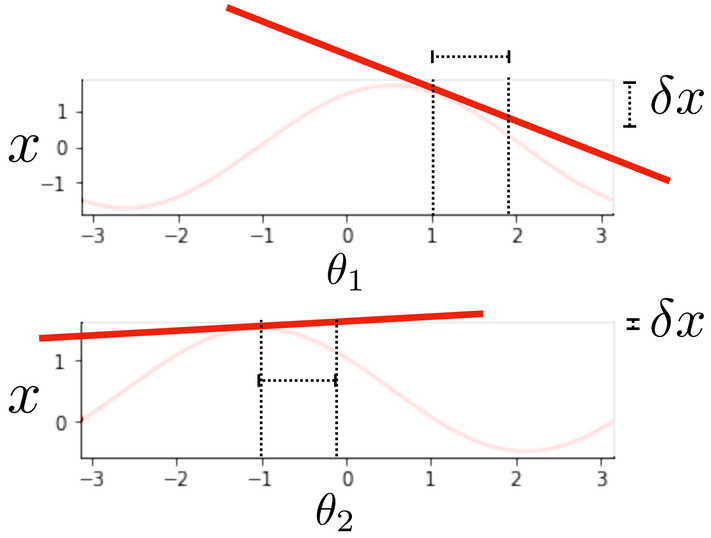


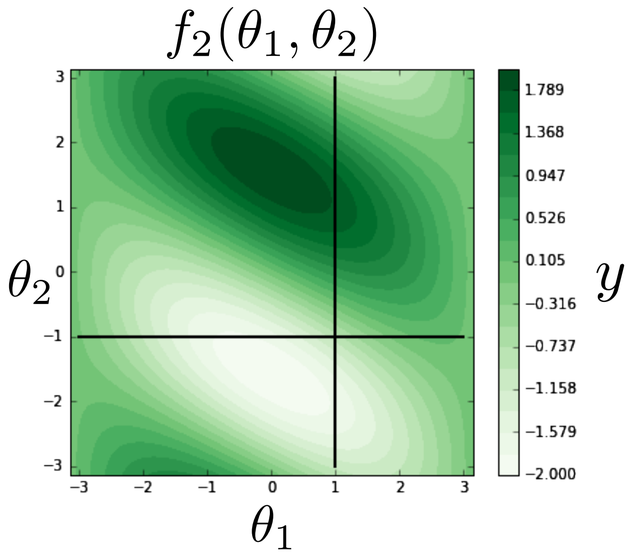


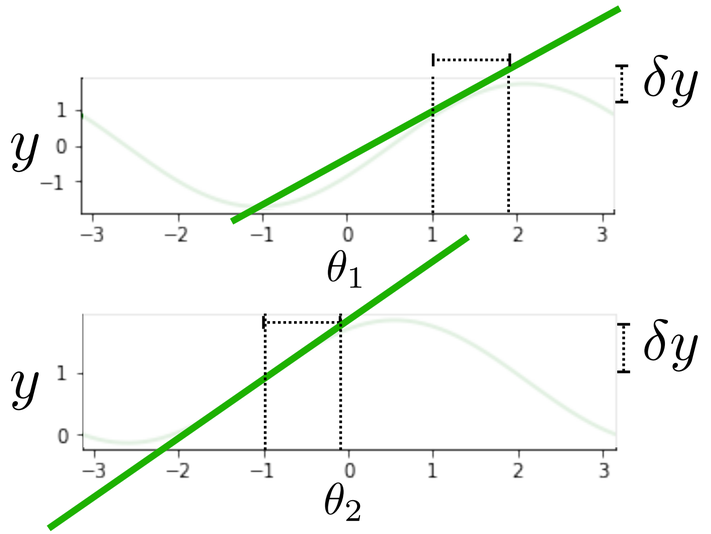
Jacobian

Forward kinematics







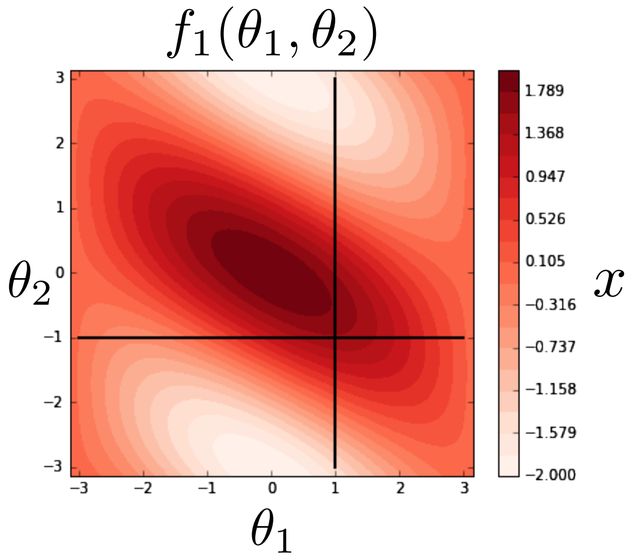


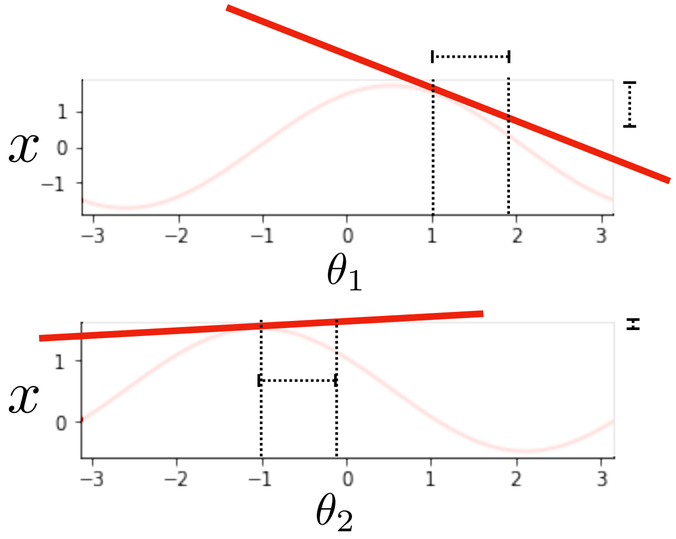
$$\begin{matrix} J & \delta q \end{matrix}$$

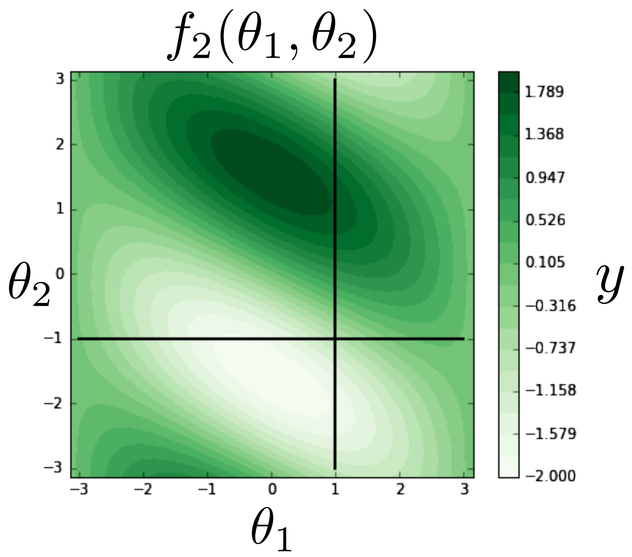
$$\begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix} \begin{bmatrix} \delta \theta_1 \\ \delta \theta_2 \end{bmatrix}$$

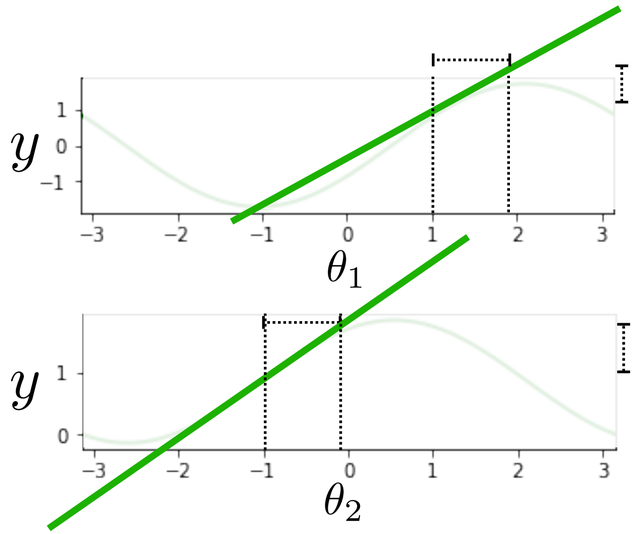
Jacobian

Forward kinematics





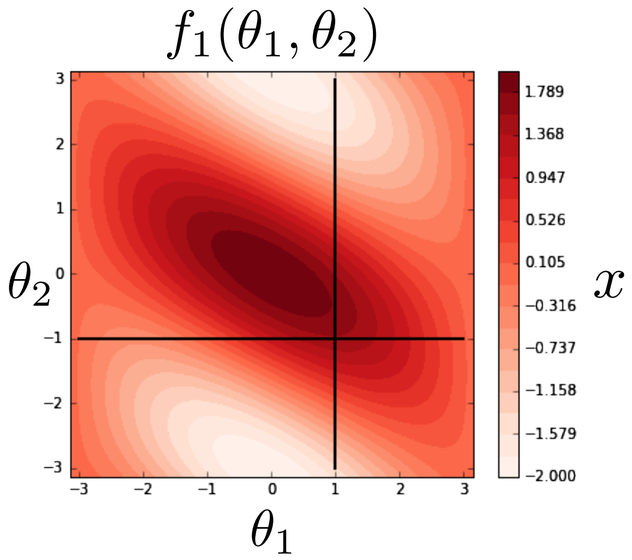


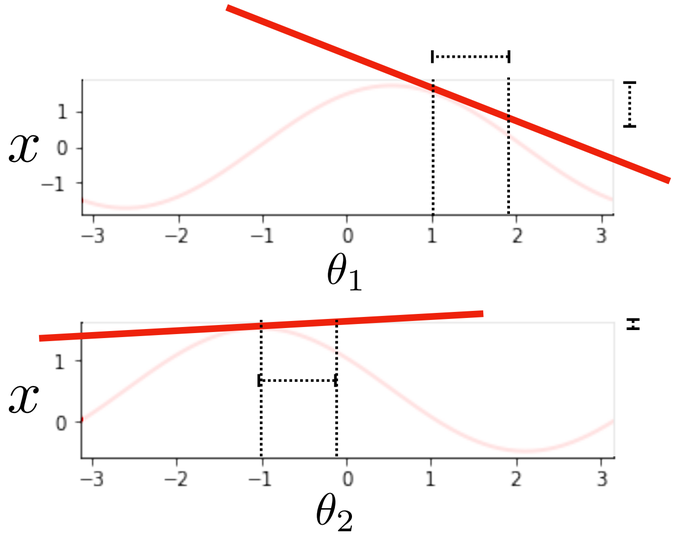


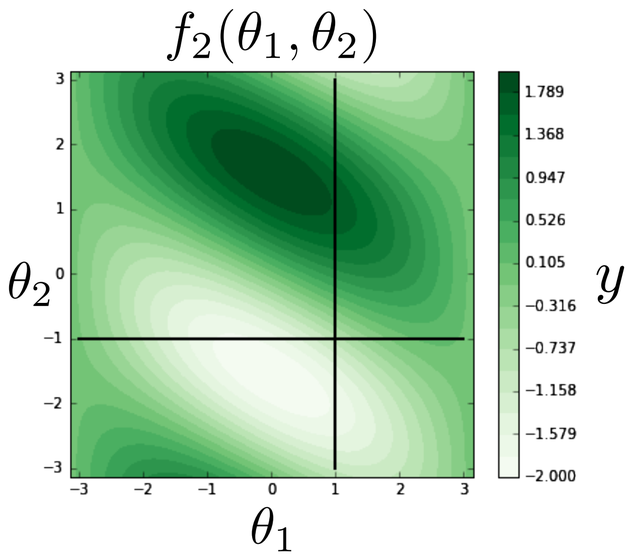
$$\underset{\delta \mathrm{x}}{\begin{bmatrix} \delta x \\ \delta y \end{bmatrix}} = \underset{J}{\begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix}} \underset{\delta q}{\begin{bmatrix} \delta \theta_1 \\ \delta \theta_2 \end{bmatrix}}$$

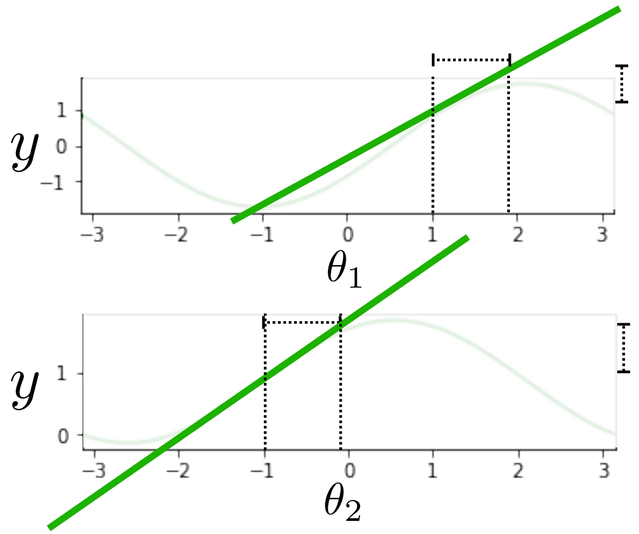
Jacobian

Forward kinematics









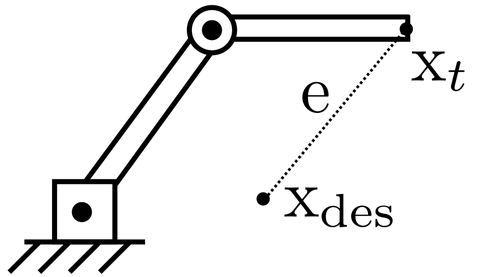
$$\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix} \begin{bmatrix} \delta \theta_1 \\ \delta \theta_2 \end{bmatrix}$$

δx J δq

Easy to invert !!!

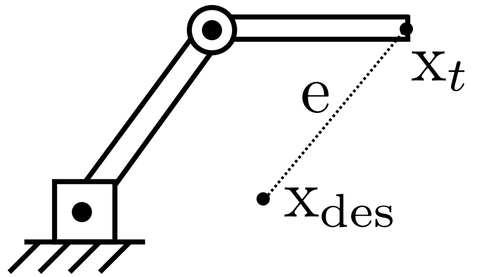
But how to use
the Jacobian
for inverse Kinematics ???

Inverted Jacobian for inverse kinematics



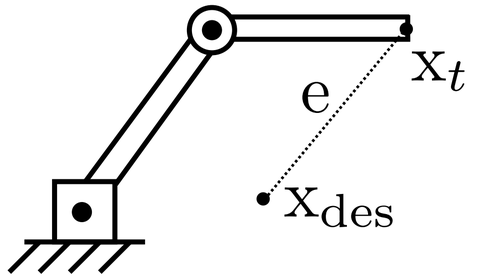
Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



Inverted Jacobian for inverse kinematics

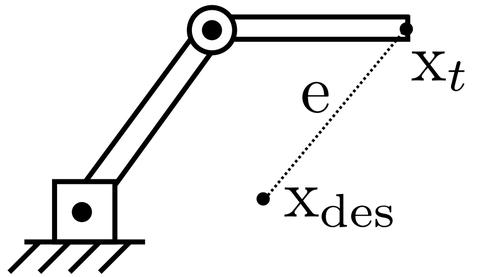
Algorithm for
P-Controller in operational space:



Repeat:

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:

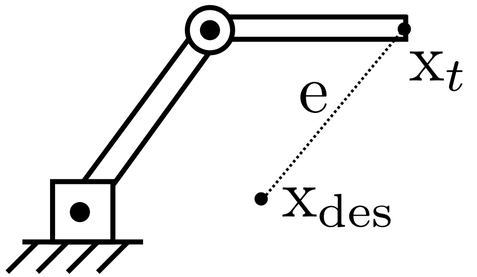


Repeat:

Compute $\mathrm{x}_t = f(\mathrm{q}_t)$ **(using forward model)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



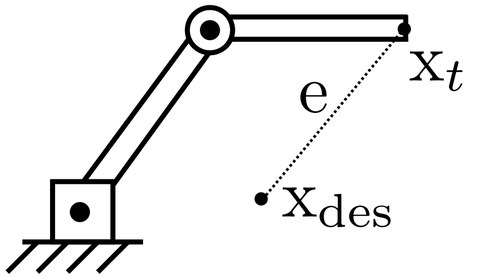
Repeat:

Compute $\mathrm{x}_t = f(\mathrm{q}_t)$ **(using forward model)**

Compute $\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$ **(i.e. the error)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



Repeat:

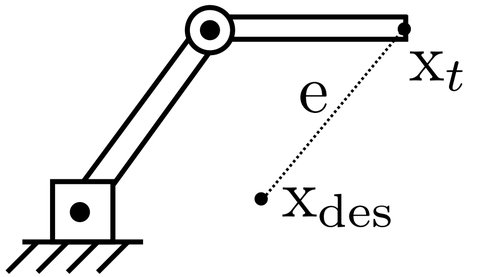
Compute $\mathrm{x}_t = f(\mathrm{q}_t)$ **(using forward model)**

Compute $\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$ **(i.e. the error)**

Set $\delta \mathbf{x} = \gamma \mathbf{e}$ **(small step to reduce the error)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



Repeat:

Compute $\mathrm{x}_t = f(\mathrm{q}_t)$ **(using forward model)**

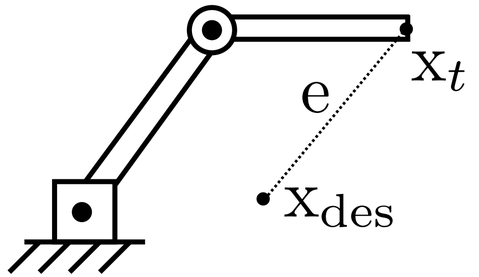
Compute $\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$ **(i.e. the error)**

Set $\delta \mathbf{x} = \gamma \mathbf{e}$ **(small step to reduce the error)**

Compute J **(using** q_t**)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



Repeat:

Compute $\mathrm{x}_t = f(\mathrm{q}_t)$ **(using forward model)**

Compute $\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$ **(i.e. the error)**

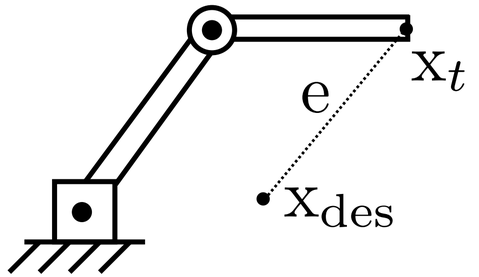
Set $\delta\mathbf{x} = \gamma\mathbf{e}$ **(small step to reduce the error)**

Compute J **(using** q_t**)**

Compute J^{-1} **(much easier than inverting f)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:

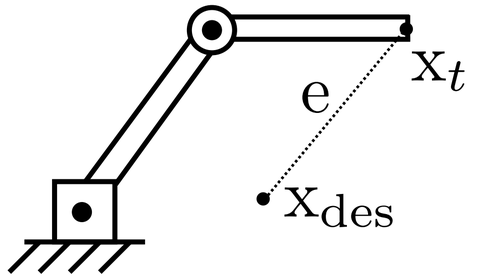


Repeat:

Compute	$\mathrm{x}_t = f(\mathrm{q}_t)$	**(using forward model)**
Compute	$\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$	**(i.e. the error)**
Set	$\delta\mathbf{x} = \gamma\mathbf{e}$	**(small step to reduce the error)**
Compute	J	**(using q_t)**
Compute	J^{-1}	**(much easier than inverting f)**
Compute	$\delta\mathrm{q} = J^{-1}\ \delta_x$	**(this is the inverse kinematics part)**

Inverted Jacobian for inverse kinematics

Algorithm for
P-Controller in operational space:



Repeat:

Compute	$\mathrm{x}_t = f(\mathrm{q}_t)$	**(using forward model)**
Compute	$\mathrm{e} = (\mathrm{x}_{\mathrm{des}} - \mathrm{x}_t)$	**(i.e. the error)**
Set	$\delta\mathbf{x} = \gamma\mathbf{e}$	**(small step to reduce the error)**
Compute	J	**(using q_t)**
Compute	J^{-1}	**(much easier than inverting f)**
Compute	$\delta\mathrm{q} = J^{-1}\ \delta_x$	**(this is the inverse kinematics part)**
Compute	$\mathrm{q}_{t+1} = \mathrm{q}_t + \delta\mathrm{q}$	**(i.e. move robot)**

Let’s make it even simpler

Use Jacobian with forces

Use Jacobian with forces

Work = Force * Distance

Work = Torque * Angle

Use Jacobian with forces

Work = Force * Distance

Work = Torque * Angle

$$\mathbf{F}^T \, \delta \mathbf{x} = \tau^T \delta \mathbf{q}$$

Use Jacobian with forces

Work = Force * Distance

Work = Torque * Angle

$$\mathbf{F}^T \, \delta\mathbf{x} = \tau^T \delta\mathbf{q}$$

using $\delta\mathbf{x} = J(\mathbf{q}) \, \delta\mathbf{q}$

$$\mathbf{F}^T \, J\delta\mathbf{q} = \tau^T \delta\mathbf{q}$$

Use Jacobian with forces

Work = Force * Distance

Work = Torque * Angle

$$\mathbf{F}^T \, \delta\mathbf{x} = \tau^T \delta\mathbf{q}$$

using $\delta\mathbf{x} = J(\mathbf{q}) \, \delta\mathbf{q}$

$$\mathbf{F}^T \, J\delta\mathbf{q} = \tau^T \delta\mathbf{q}$$

divide by $\delta\mathbf{q}$

$$\mathbf{F}^T \, J = \tau^T$$

Use Jacobian with forces

Work = Force * Distance

Work = Torque * Angle

$$\mathbf{F}^T \, \delta\mathbf{x} = \tau^T \delta\mathbf{q}$$

using $\delta\mathbf{x} = J(\mathbf{q}) \, \delta\mathbf{q}$

$$\mathbf{F}^T \, J\delta\mathbf{q} = \tau^T \delta\mathbf{q}$$

divide by $\delta\mathbf{q}$

$$\mathbf{F}^T \, J = \tau^T$$

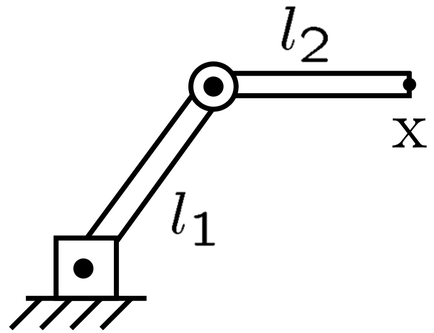
using $(A \; B)^T = B^T \; A^T$

$$\tau = J^T \mathbf{F}$$

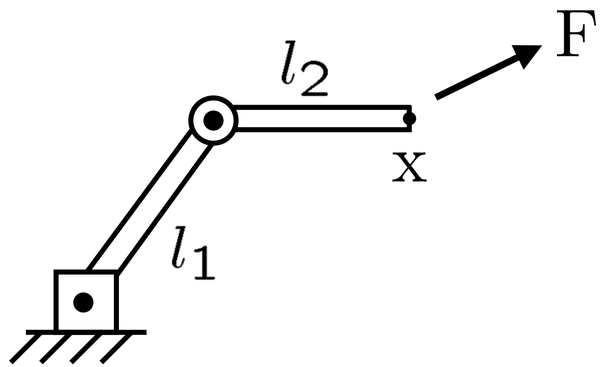
Super simple !!!

Use Jacobian with forces

Use Jacobian with forces

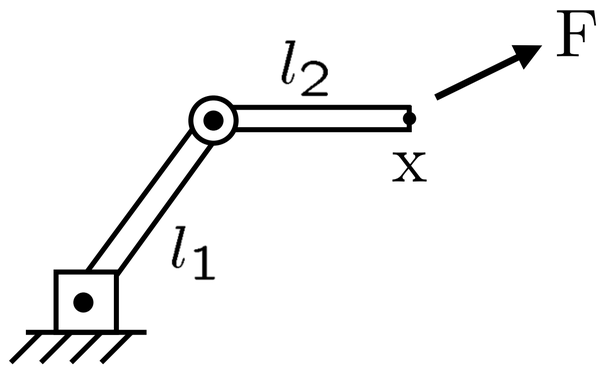


Use Jacobian with forces



Use Jacobian with forces

$$\tau = J^T \mathbf{F}$$



Use Jacobian with forces

$$\tau = J^T \mathbf{F}$$