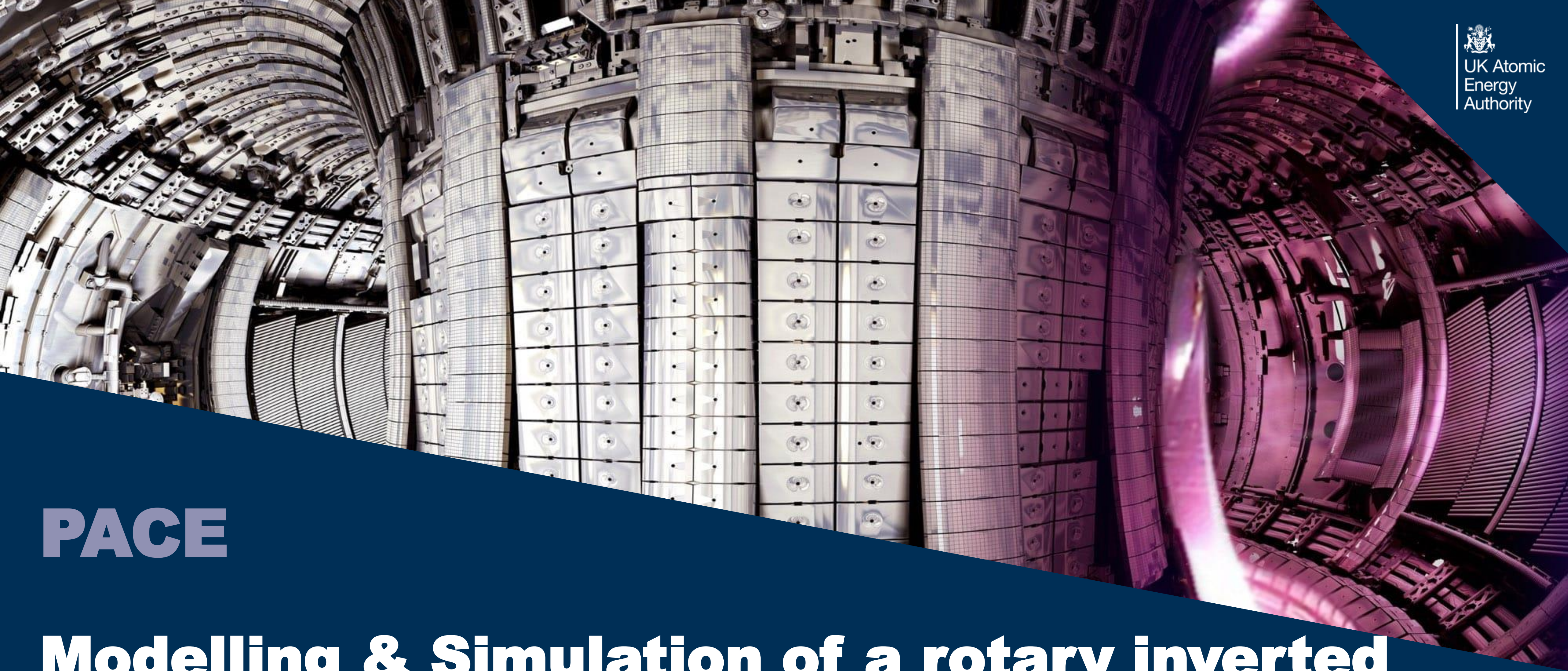


UK Atomic Energy Authority

PACE

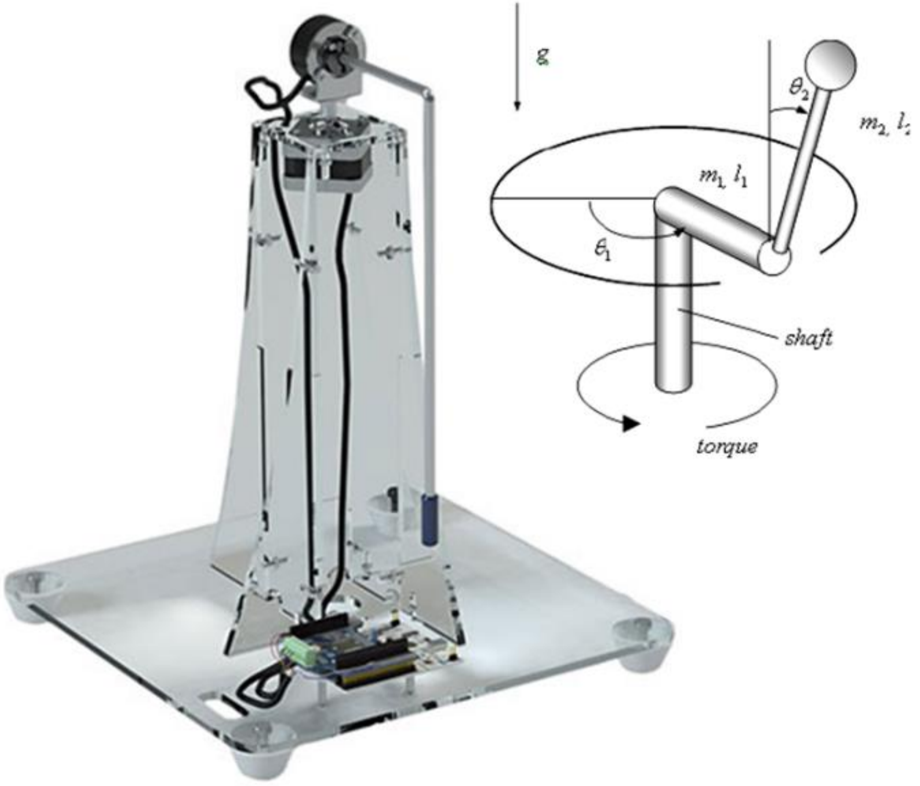
Modelling & Simulation of a rotary inverted pendulum

Sam Herschmann

sam.herschmann@ukaea.uk

Scope

These slides outline the steps required to derive the non-linear equations of motion (aka the dynamic model) of a rotary pendulum with only one actuator.

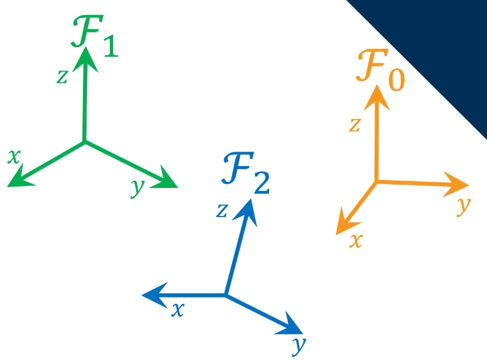


STEVAL-EDUKIT01

<u>Challenging topics involved:</u>

- Vector calculus
- Rigid Body Kinematics & Dynamics
    - Lagrange’s Equations
- Simulation of ODEs

Important Kinematics

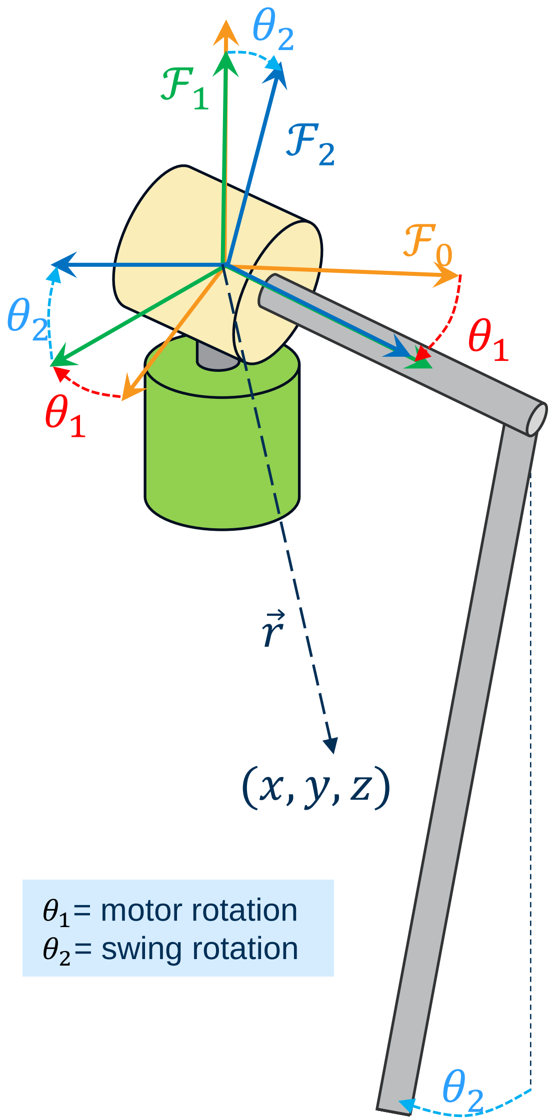


There are 3 frames of reference: $\mathcal{F}_0$, $\mathcal{F}_1$ and $\mathcal{F}_2$.

Inertial (globally fixed) frame

Encoder-fixed frame

Pendulum-fixed frame



θ_1= motor rotation
θ_2= swing rotation

- Let's define some position vector $\vec{r}$. We can express this as a numerical (x, y, z) vector, but these values depend on the frame of reference that we choose to express the vector in.
- Let's define: "$\vec{r}$ expressed in frame $\mathcal{F}_0$" as $\boldsymbol{r}^{\mathcal{F}_0} \in \mathbb{R}^3$.
- For our pendulum system, it can be shown that:

$$\boldsymbol{r}^{\mathcal{F}_1} = \mathbf{R}_{10}(\theta_1)\boldsymbol{r}^{\mathcal{F}_0}$$
$$\boldsymbol{r}^{\mathcal{F}_2} = \mathbf{R}_{21}(\theta_2)\boldsymbol{r}^{\mathcal{F}_1}$$

$$\boldsymbol{r}^{\mathcal{F}_2} = \underbrace{\mathbf{R}_{21}\mathbf{R}_{10}}\boldsymbol{r}^{\mathcal{F}_0}$$

$$\mathbf{R}_{20}(\theta_1, \theta_2) = \begin{bmatrix} \cos\theta_1 \cos\theta_2 & -\sin\theta_1 & \sin\theta_2 \cos\theta_1 \\ \sin\theta_1 \cos\theta_2 & \cos\theta_1 & \sin\theta_1 \sin\theta_2 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix}$$

This is known as a $z \rightarrow y'$ (yaw-pitch) transformation

Rotation matrices R

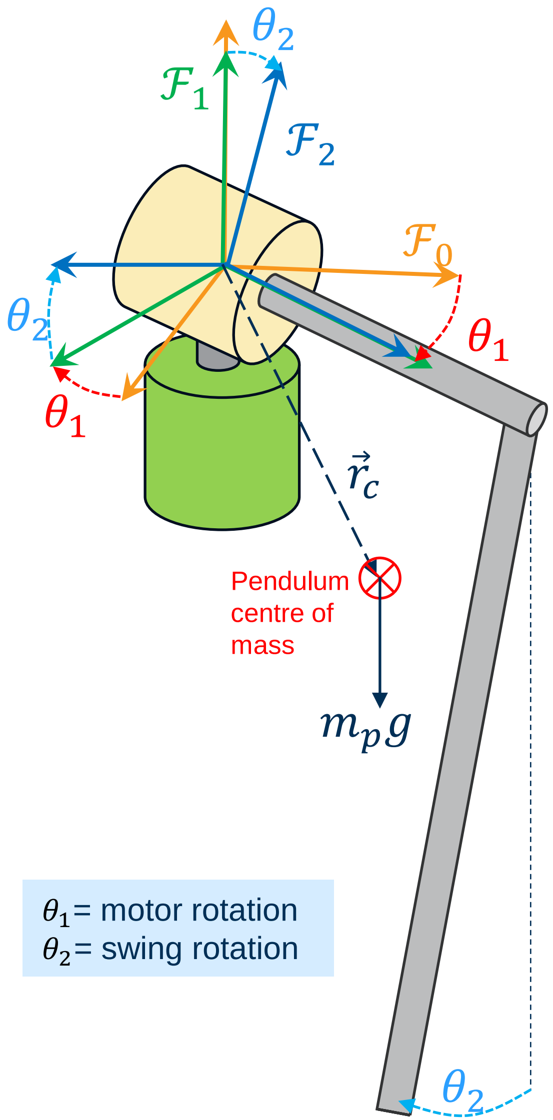
$$\mathbf{R}_{10} = \mathbf{R}_z(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{21} = \mathbf{R}_y(\theta_2) = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix}$$

$\mathbf{R}_{20}$ describes the orientation of the pendulum body with respect to the inertial frame of reference.

UK Atomic Energy Authority

Calculating the energy of the payload



θ_1= motor rotation
θ_2= swing rotation

Gravitational Potential Energy

$$V = m_p g\, \boxed{\boldsymbol{r}_c\big|_z}$$

Vertical position of the centre of mass, expressed in the inertial frame.
$\boldsymbol{r}_c = \boldsymbol{r}_c^{\mathcal{F}_0} = \mathbf{R}_{20}(\theta_1,\theta_2)^\top\, \boldsymbol{r}_c^{\mathcal{F}_2}$

Rotational Kinetic energy

$$T_{rot_p} = \frac{1}{2}\boldsymbol{\omega}_p^\top \boxed{\boldsymbol{I}_p}\boxed{\boldsymbol{\omega}_p}$$

Pendulum inertia tensor, calculated in the next slide

Angular velocity of pendulum, expressed in the inertial frame. This can be shown to be:

$$\boldsymbol{\omega}_p = \boldsymbol{\omega}_p^{\mathcal{F}_0} = \mathbf{R}_z(\theta_1)^\top \begin{bmatrix} 0 \\ \dot{\theta}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

Linear Kinetic energy

$$T_{lin_p} = \frac{1}{2}\boxed{m_p}\dot{\boldsymbol{r}}_c^\top \boxed{\dot{\boldsymbol{r}}_c}$$

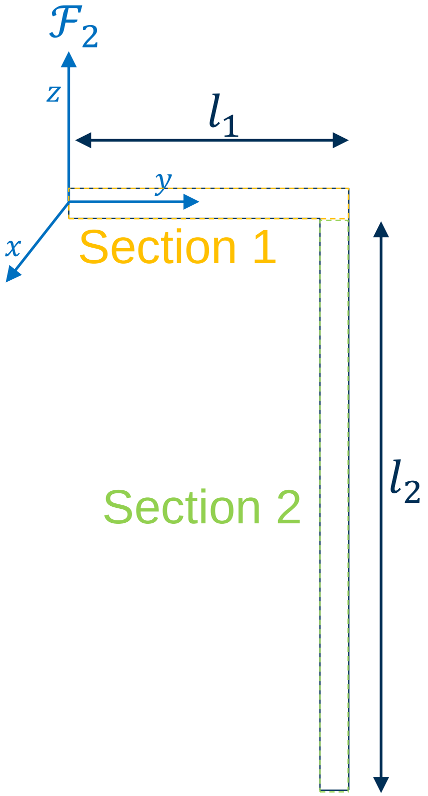
Pendulum Mass

Linear velocity of pendulum centre of mass, expressed in the inertial frame. This can be shown to be:
$\dot{\boldsymbol{r}}_c = \dot{\boldsymbol{r}}_c^{\mathcal{F}_0} = \dot{\mathbf{R}}_{20}(\theta_1,\theta_2)^\top\, \boldsymbol{r}_c^{\mathcal{F}_2}$

We can now define the systems 'Lagrangian': $L = T_{rot_p} + T_{lin_p} - V$

Aside: Approximating the pendulum Inertia Tensor I_p

UK Atomic Energy Authority



$\boldsymbol{I}_p$ can be thought of as the 'rotational mass' of an object. This depends on the axis of rotation, which makes $\boldsymbol{I}_p$ a 3X3 matrix.

$$\boldsymbol{I}_p^{\mathcal{F}_2} = \boldsymbol{I}_1^{\mathcal{F}_2} + \boldsymbol{I}_2^{\mathcal{F}_2}$$

Pendulum inertia tensor resolved in the moving (or 'body') frame $\mathcal{F}_2$

Section 1

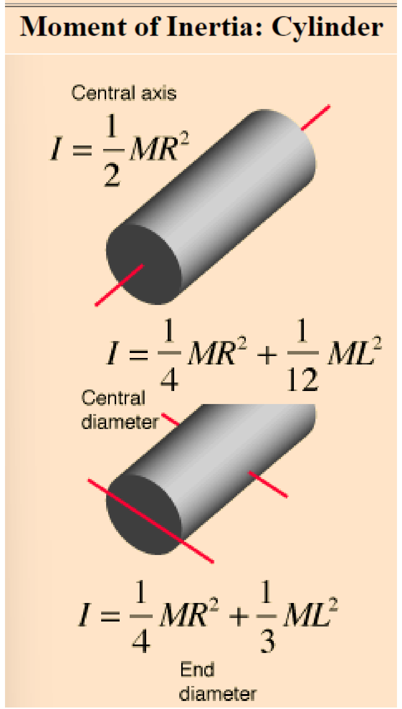
$$\boldsymbol{I}_1^{\mathcal{F}_2} = \begin{bmatrix} \frac{1}{4}m_1r^2 + \frac{1}{3}m_1l_1^2 & 0 & 0 \\ 0 & \frac{1}{2}m_1r^2 & 0 \\ 0 & 0 & \frac{1}{4}m_1r^2 + \frac{1}{3}ml_1^2 \end{bmatrix}$$

Section 2

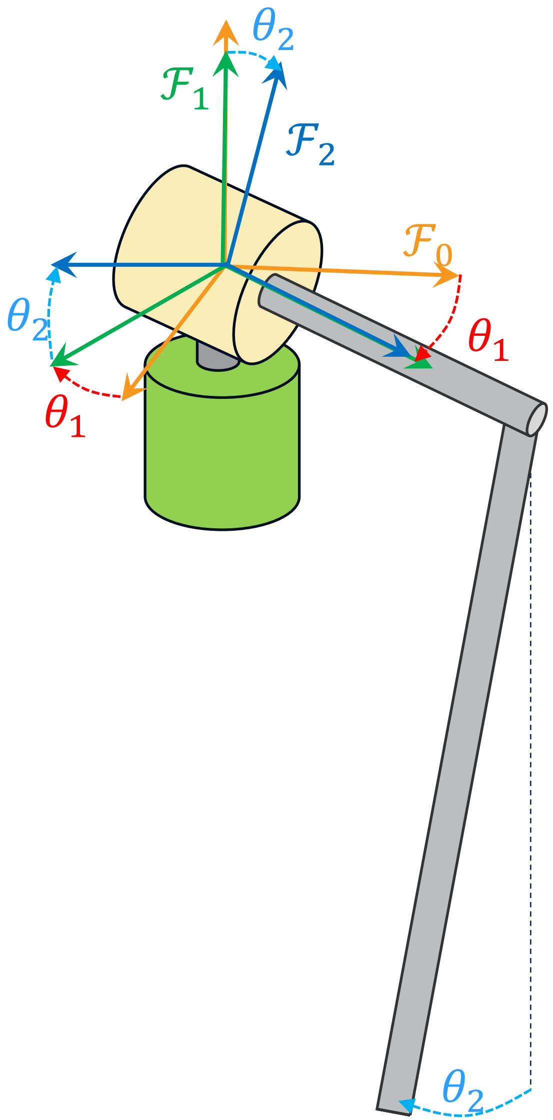
$$\boldsymbol{I}_2^{\mathcal{F}_2} = \begin{bmatrix} \frac{1}{4}m_2r^2 + \frac{1}{12}m_2l_2^2 & 0 & 0 \\ 0 & \frac{1}{4}m_2r^2 + \frac{1}{12}m_2l_2^2 & 0 \\ 0 & 0 & \frac{1}{2}m_2r^2 \end{bmatrix} + \underbrace{m_2\left(\begin{bmatrix} 0 \\ l_1 \\ \frac{-l_2}{2} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 \\ l_1 \\ \frac{-l_2}{2} \end{bmatrix} \boldsymbol{E}_3 - \begin{bmatrix} 0 \\ l_1 \\ \frac{-l_2}{2} \end{bmatrix} \begin{bmatrix} 0 \\ l_1 \\ \frac{-l_2}{2} \end{bmatrix}^{\mathsf{T}} \right)}_{\text{From the parallel axis theorem}}$$

Assumption: solid cylinder

(can improve this using measurements)



Finding the state-space 'Equations of Motion'



Let's define our state coordinates, $\boldsymbol{q} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$.

$\rightarrow$ $\boldsymbol{q}$ completely describes the position of our pendulum

Euler-Lagrange Equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \boxed{\boldsymbol{F}}$$

The 'external force' vector. If we ignore no damping from friction, then $\boldsymbol{F} = \begin{bmatrix} u \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$, where u is the torque applied by the motor.

This equation basically creates the "F=ma" of the system: the equation that describes the behaviour of the pendulum. Also known as a "Dynamic Model".

By substituting L into this and expanding / rearranging, the result ends up looking something like this:

$$\boxed{\boldsymbol{M}(\boldsymbol{q})}\ddot{\boldsymbol{q}} + \boxed{\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})} = \boldsymbol{F}$$

"Mass Matrix"

Vector describing gravity & Coriolis effects

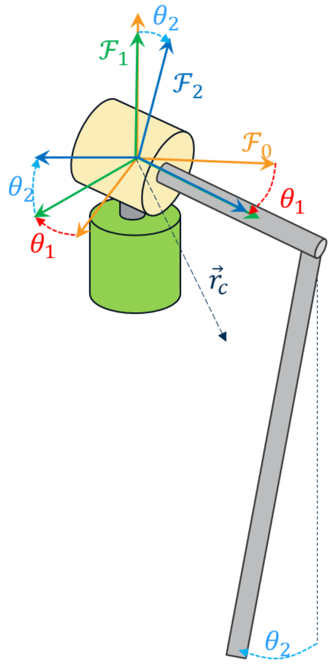
$\equiv$

Often it is nice to express this in the *equivalent* 'first order ODE form', by defining a new set of variables

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{F} - \boldsymbol{N}) \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x}, u)}$$

Simulating the free-response system (u=0)

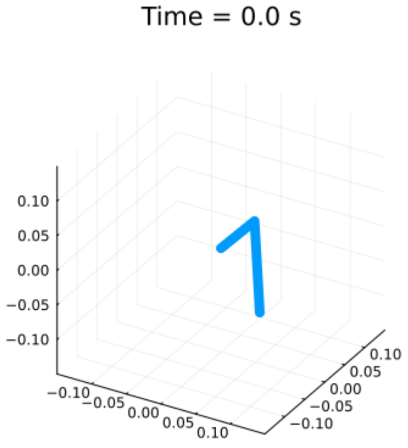


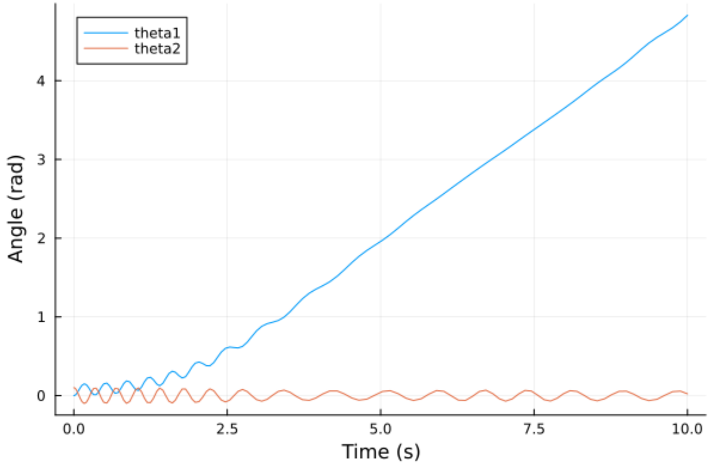
Given a starting state $\boldsymbol{x}(\boldsymbol{t}=\mathbf{0})=\boldsymbol{x}_0$, the model equations $\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})$ can be 'solved' (aka 'simulated') using a variety of techniques (aka 'ODE solvers').

- This can be done in Python, Matlab, C++, Julia and more.
  - I use ODE solvers in Julia's DifferentialEquations.jl toolbox, this is just my personal preference.

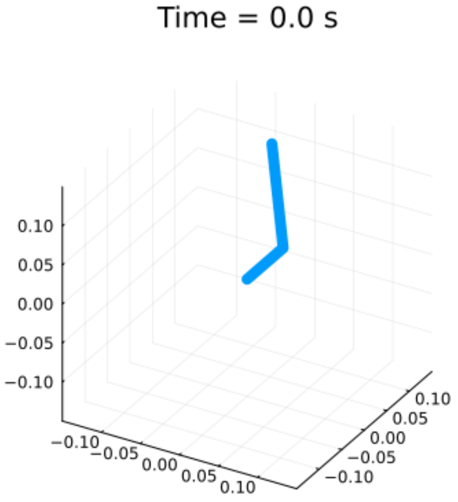
You can then create an animation of the model simulation.

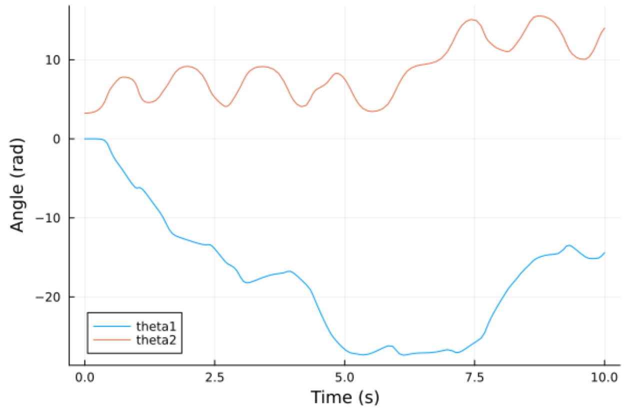
Example 1: $\boldsymbol{x}_0=\begin{bmatrix}\theta_{1_0}\\ \theta_{2_0}\\ \dot{\theta}_{1_0}\\ \dot{\theta}_{2_0}\end{bmatrix}=\begin{bmatrix}0\\ 0.1\\ 0\\ 0\end{bmatrix}$





Example 2: $\boldsymbol{x}_0=\begin{bmatrix}0\\ \pi+0.1\\ 0\\ 0\end{bmatrix}$





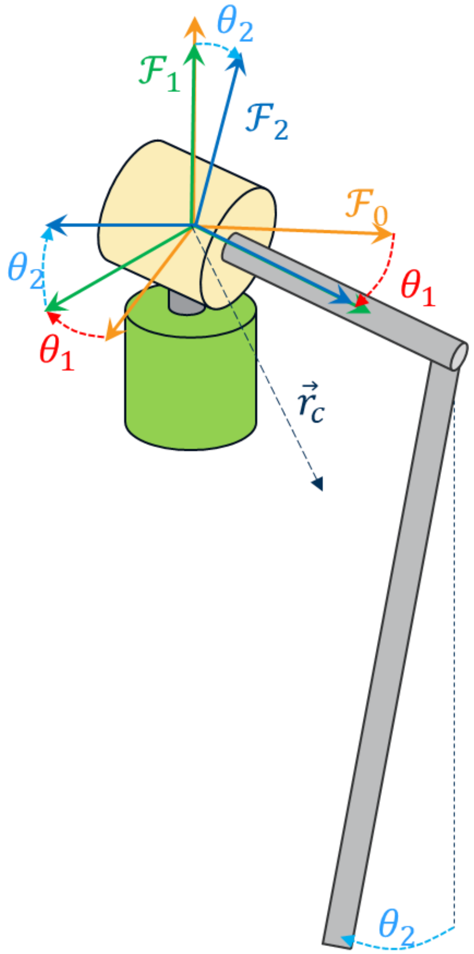
Very chaotic!
(no damping is modelled)

Rotary inverted pendulum dynamic modelling

Adding Damping

In reality, there will be *viscous damping* at the joints.

- This is a friction torque proportional to the joint velocity·



A damping constant of proportionality for joint i

$$\tau_{fr_i} \approx -\boxed{d_{fr_i}}\dot{\theta}_i$$

In vector form:

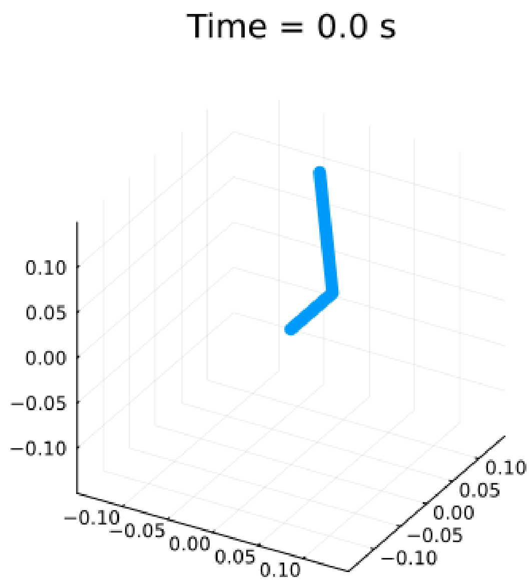
$$\boldsymbol{\tau}_{fr} \approx -\boxed{\boldsymbol{D}}\dot{\boldsymbol{\theta}}$$

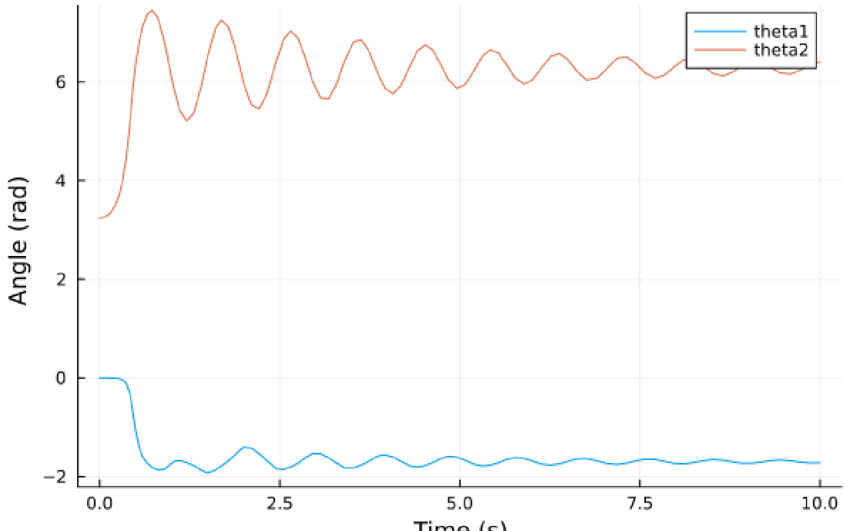
A diagonal Damping matrix

The model now becomes:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{D}\dot{\boldsymbol{\theta}} = \boldsymbol{F}$$

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{F} - \boldsymbol{N} + \boldsymbol{D}\dot{\boldsymbol{\theta}}) \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}$$





Next steps

- **Improve model parameters & verify**
- **Model Linearisation**
- **Control**
  - LQR
  - Swing-up trajectory?