ECE MBD: TU4 p1

Self-balancing bot: Model in-the-Loop

The main objective of this laboratory session is to make use and improve the knowledge gained along the MBD course for modeling a more complex system: an inverted pendulum (self-balancing bot). One of the first tasks is the derivation of the systems equations representing the dynamic behavior of the self-balancing bot. The model shall be carried out first in its non-linear form and afterwards will be linearized by taking into account physical operation constraints. The linear model will be converted to its transfer function representation by means of Laplace transform. Finally, a controller shall be designed in order to keep the system in balance. Together with the following laboratory session (TU4 p2) the whole Model Based Design process will be covered and the virtual environment will be replaced by its real counterpart. Consider also the appendix of this document.

Laboratory work

Model development

In principle the self-balancing bot is an inverted pendulum. The model of such a system can be obtained by starting at the free body diagram of the system. Let's assume that the system has the following layout:

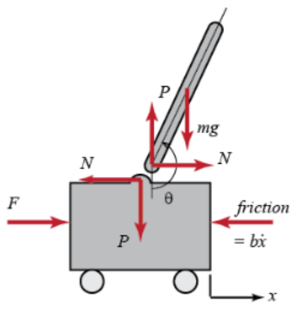


Figure 1: System layout

Where:

x: cart's position	b: coefficient of friction for cart
$\dot{x}$: cart's velocity	l: length to pendulum center of mass
$\ddot{x}$: cart's acceleration	J: moment of inertia of the pendulum
θ: pendulum's position (angle)	F: external force applied (by motors)
$\dot{\theta}$: angular velocity	N: interaction force between cart and pendulum in x direction
$\ddot{\theta}$: angular acceleration	P: interaction force between cart and pendulum in y direction
m: mass of pendulum	g: gravitational constant
M: mass of cart	

Tasks:

Modell Development

1. **Derive** the system equations based on Newton's laws. The pendulum and cart have one degree of freedom. Use the Figure 1 as base! (You can integrate handwritten derivations to the report if readable)
   - θ: pendulum's position (angle)
   - x: cart's position

   An angle of zero shall indicate the upright position of the self-balancing bot (perfectly balanced), or its aligned with the y coordinate axes

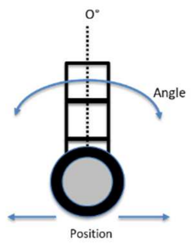


Figure 2: Zero angle position

2. **Implement** the derived differential equation of the system in MATLAB/Simulink (use standard blocks) as done for the system modelling session. For a proper handling of the system parameters consider to use an m-file

3. The **parameters** of the system can be derived from the hardware setup. Following parameters are required:

• Mass of pendulum	m = 0.1756 kg	(0.877*0.2)
• Mass of chart	M = 0.7016 kg	(0.877*0.8)
• Gravitational constant	g = 9.81 m/s^2	
• Friction coefficient	b = 0.1 kg/s	
• Length to pendulum center	l = 0.11 m	
• Width of bot	W = 0.145 m	
• Moment of inertia	J = 0.001 kgm^2	

System analysis and control

4. Analyze the open loop response (pendulum's angle and self-balancing bot position) as follows:
   - Apply zero force at the input and observe the response.
     - What is the initial pendulum position x?
     - What is the initial angle θ?
     - Is the system stable?

     Report your observations.
   - Modify the model for having an initial condition of θ = 5°
     - Where does this initial condition need to be applied?
     - Is the system still stable?

     IMPORTANT! Take care of selecting an adequate solver and its settings.
   - Interpret and report your findings.
5. Apply a proportional (P) control in a feedback loop and search for a suitable Kp
   - Use as a reference an angle of zero and start with Kp = 1 and observe the system response
   - Increase gradually (different experiments) the value of Kp and analyse the results. Generate a table to compare the different experiments in the form:
     - Selected Kp;
     - Oscillation period(sec);
     - stable response Y/N;

     How long does the system stay stable?
   - Report and interpret your observations

Linearization

6. As our approaches for modelling and control are restricted to linear time invariant systems it is required to **linearize** the system equations.
   - What is meant by the linearization process?
   - Explain why it is possible to linearize the equations?
   - Describe the restrictions, and show the contrast between the linear versus non-linear version
7. Derive by yourself the **transfer functions** for: (handwritten derivations are allowed for the report if readable)
   - pendulum's position: θ/F(s)
   - and self-balancing bot position: x(s)/F(s)
   - compare the correctness by checking against the previous implementation
   - Analyze the allocation of poles and zeros of the continuous transfer functions. Explain your findings.

     Hint: You can use the MATLAB's commands *pole* or *pzmap.*
8. **Implement** in the same Simulink model, the linear system together with the non-linear one and compare their responses:
   - Use transfer function blocks in MATLAB/Simulink to create the linear system
   - Apply to both systems (linear and non-linear) a little deviation in the pendulums angle during 0.1sec., e.g. 5°
   - What are your observations?

PID Control

<u>**Functional requirements**</u>

FR1. The control function for the self-balancing bot requires a Kalman filter followed by a PID controller.

- The Kalman filter provides a more plausible actual vertical position (given in degrees) using the information coming from the plant (The filter will be provided with the template for p2).
- Next, the PID controller computes the necessary torque that the motors shall provide in order to keep the 0° vertical position.

FR2. The control function shall operate only in the range between -40° and 40°. In case that the actual position exceeds the operating range then the traction motors shall stop immediately.

<u>**Non-functional requirements**</u>

NF1. The entire control function shall be implemented as a triggered sub-system model in MATLAB/Simulink with a **sampling time of 0.01** sec.

NF2. The Kalman filter script (provided as an annex) shall be included in a MATLAB function within the control function. It is recommendable to test the Kalman filter in a standalone model as shown as follows:

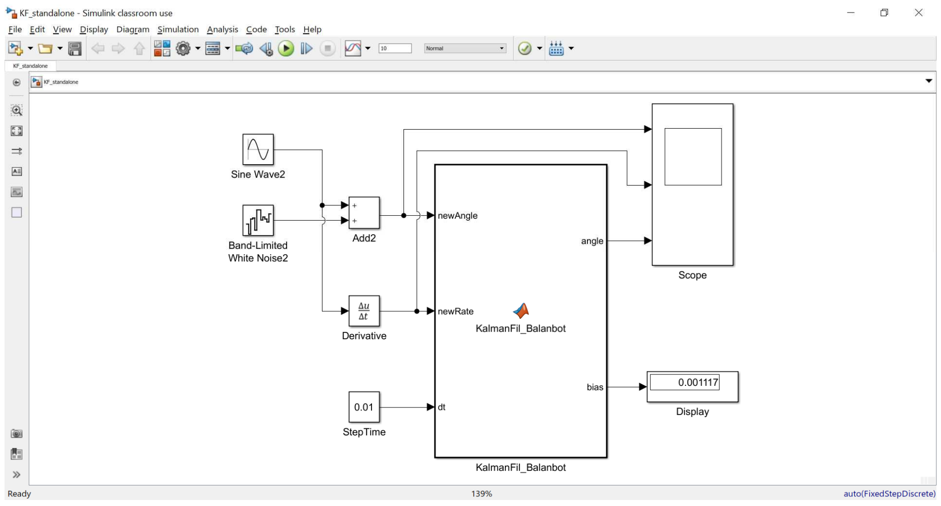


Figure 3: Kalman Filter test

NF3. The PID controller shall be implemented by means of Simulink blocks, i.e. don't use the predefined PID block; Build the controller according to:

a. The proportional, integral and derivative gains (Kp, Ki and Kd; respectively) shall be adjustable during run

b. Kp shall be multiplied by angle (output of Kalman filter)

c. Ki shall be multiplied by the integral of angle (output of Kalman filter)

d. Kd shall be multiplied directly by newRate (input of the Kalman filter)

Note: In the hardware implementation this is a reliable signal coming from the gyroscope.

e. The output of the controller (sum of the three previous parts) shall be saturated to -255 and 255

Hint 2: In order to **convert the control output into torque** (force) for the plant, a **factor of 0.005** is recommended. This factor was empirically found!

9. Select a value for Kp capable to hold the system under control by a reasonable time (let's consider 20 sec.) Utilize this value as Ku and its corresponding oscillation period as Tu and tune a PID controller according to "Ziegler Nichols method"
10. Apply the PID controller to both systems, linear and non-linear, with the same tuning of parameters and compare the results between them. Report your findings
11. Change the reference angle around the vertical position and analyse the response. Is it possible to control different angle with both of the models? How is the position of the self-balancing bot influenced?

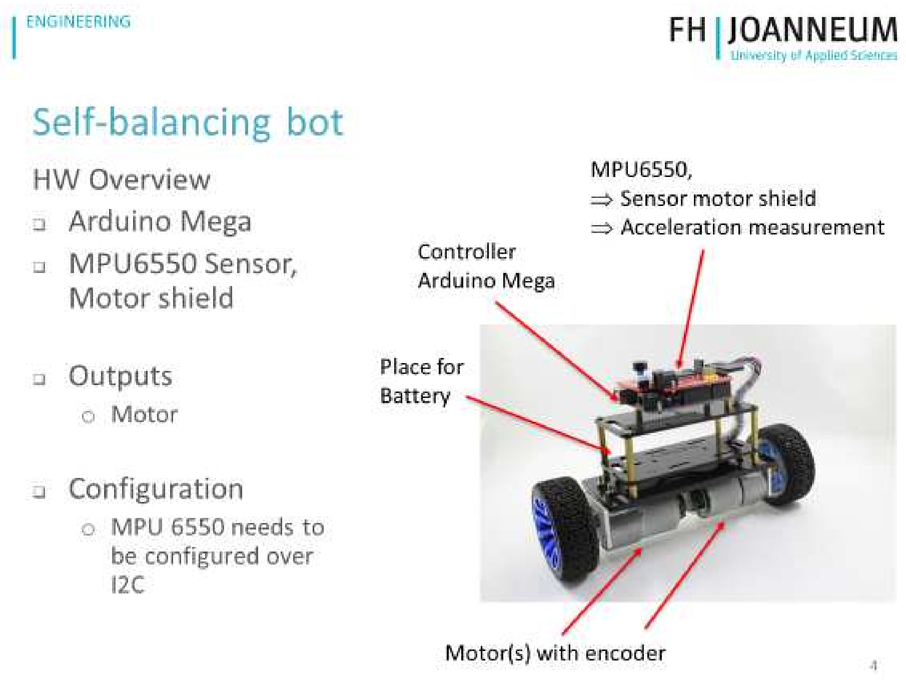
12. Try out the PID control from the Simulink library and use the Auto-Tune tool separately for each case (linear/none linear).
    - Are the suggested parameters the same or similar?
    - Are they similar to the ones found experimentally?
    - How is the system response?
    - Report and interpret your observations

Discretization linear model

13. Discretize with the help of MATLAB the linear continuous model (T=0.001 sec)
    - use the three different discretization methods (ZOH, Impulse, Tustin).
    - Analyse the allocation of poles and zeros of the discrete transfer functions
    - If the original Simulink model (none transfer function) shall be discretised, at which places the discretization should be applied?
14. Compare the results between the continuous plant (and solver) and the discrete plant (and solver). The control function in either case shall be discrete.

From the exercises, it is expected that you report your development process of your solution: design and testing.

Appendix: Model development



ENGINEERING

FH | JOANNEUM
University of Applied Sciences

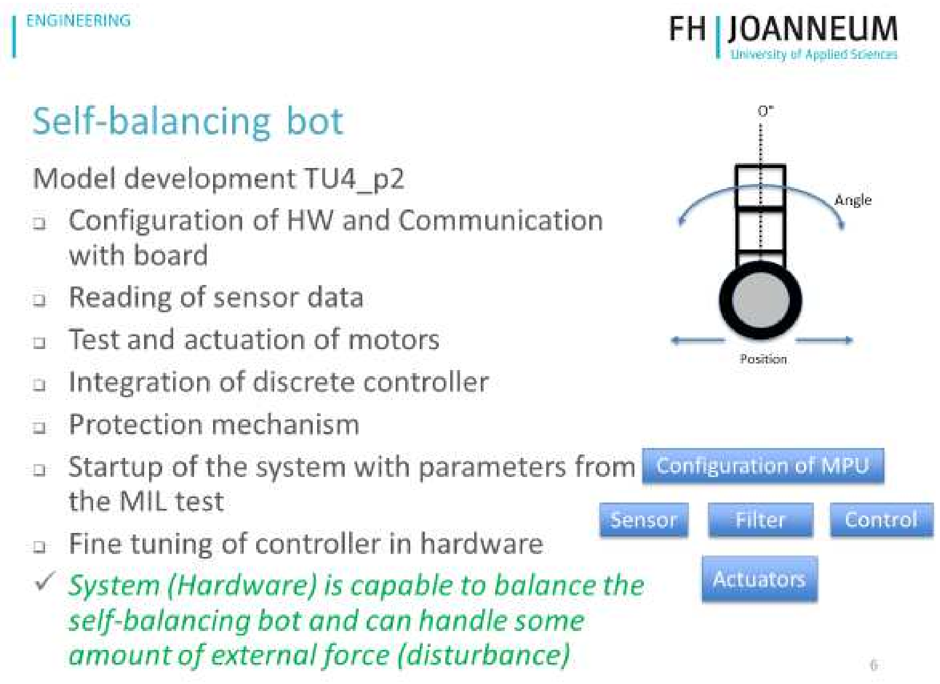
Self-balancing bot

HW Overview

- Arduino Mega
- MPU6550 Sensor, Motor shield
- Outputs
  - Motor
- Configuration
  - MPU 6550 needs to be configured over I2C

MPU6550,
⇒ Sensor motor shield
⇒ Acceleration measurement
Controller
Arduino Mega
Place for
Battery
Motor(s) with encoder

4



ENGINEERING

FH | JOANNEUM
University of Applied Sciences

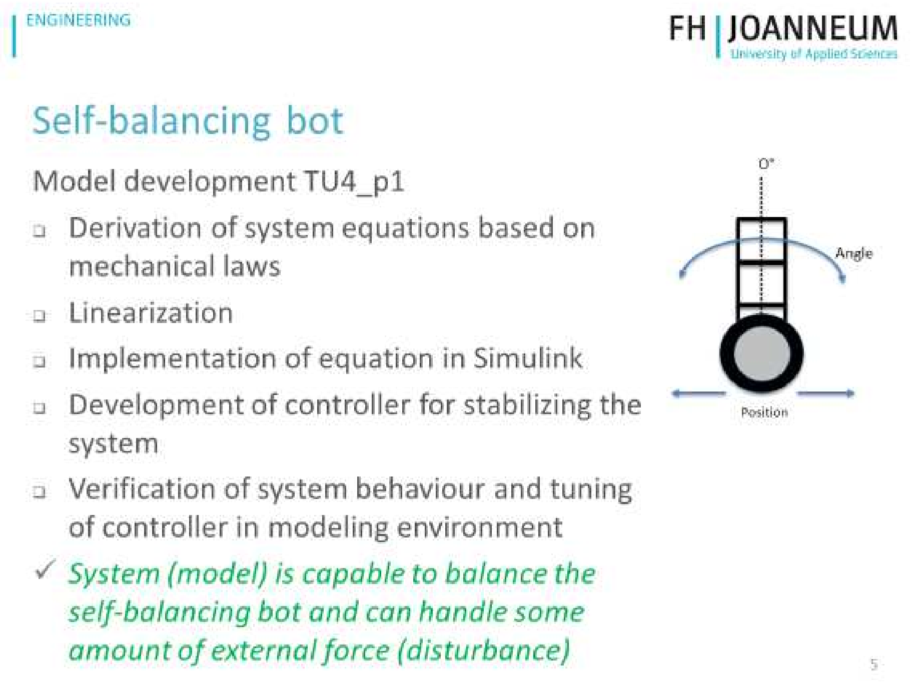
Self-balancing bot

Model development TU4_p2

- Configuration of HW and Communication with board
- Reading of sensor data
- Test and actuation of motors
- Integration of discrete controller
- Protection mechanism
- Startup of the system with parameters from the MIL test
- Fine tuning of controller in hardware

✓ *System (Hardware) is capable to balance the self-balancing bot and can handle some amount of external force (disturbance)*

6

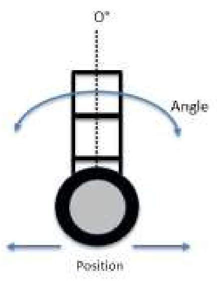


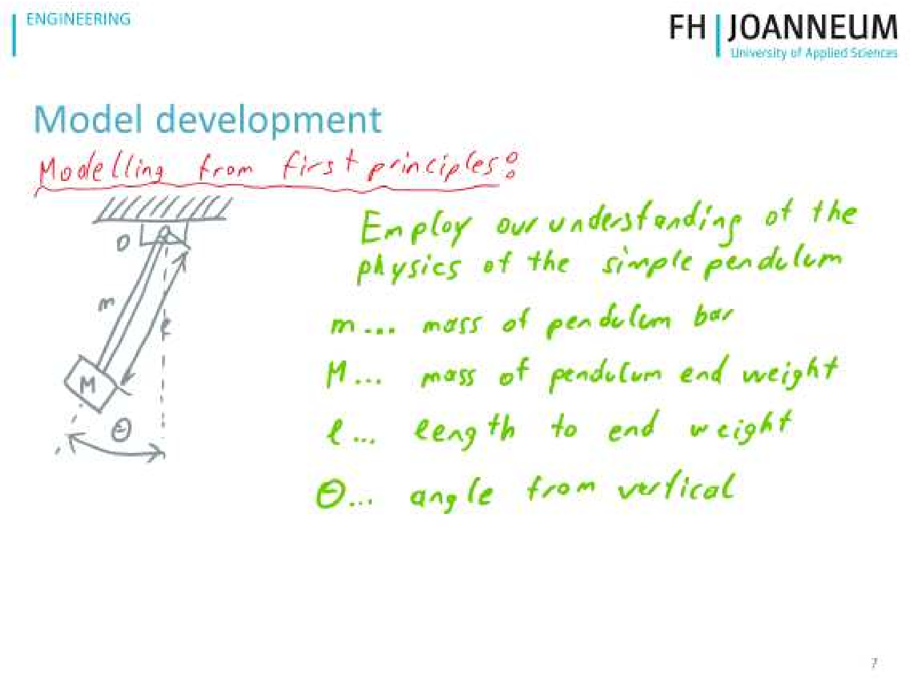
Self-balancing bot

Model development TU4_p1

- Derivation of system equations based on mechanical laws
- Linearization
- Implementation of equation in Simulink
- Development of controller for stabilizing the system
- Verification of system behaviour and tuning of controller in modeling environment

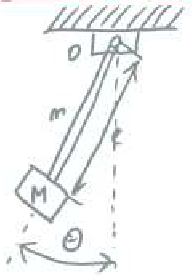
✓ *System (model) is capable to balance the self-balancing bot and can handle some amount of external force (disturbance)*





Model development

Modelling from first principles:



Employ our understanding of the physics of the simple pendulum

m ... mass of pendulum bar

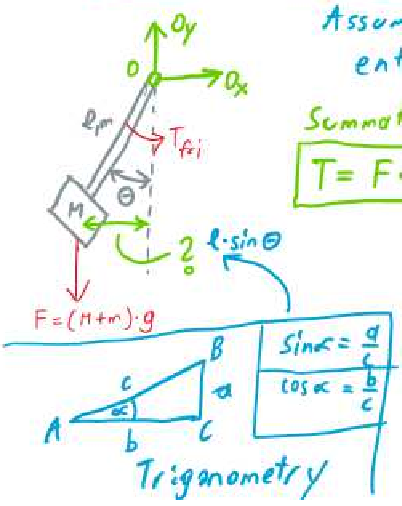
M ... mass of pendulum end weight

ℓ ... length to end weight

Θ ... angle from vertical

Model development

Free - body diagram



Assumption: mass of bar is neglipible entire mass it at the end !

Summation of moments about the point O

$\boxed{T = F \cdot r}$ Torque

$$\Sigma T = (M+m) g \ell \cdot \sin\Theta - T_{fri} = \underline{I \cdot \ddot{\Theta}}$$

$$T = I \cdot \alpha = I \cdot \ddot{\Theta}$$

Check:

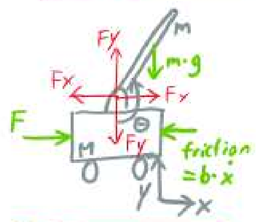
$kg \cdot \frac{m}{s^2} \cdot m - Nm$ ✓

$$T = \frac{kg \cdot m^2}{s^2} = 1J = 1 Nm$$

$$I = kg \cdot m^2 \quad \ddot{\Theta} = \frac{1}{s^2}$$

Model development

Inverted pendulum



Pendulum and cart have one degree of free dom x, Θ

↳ For these freedoms the ODE's shall be developed !

1. Equation for the cart:

$$\Sigma F = F - F_x - F_{fr} = F - F_x - b \cdot \dot{x}$$

$$\ddot{x} = a = \frac{1}{M} \cdot (F - F_x - b \cdot \dot{x}) \quad (1)$$

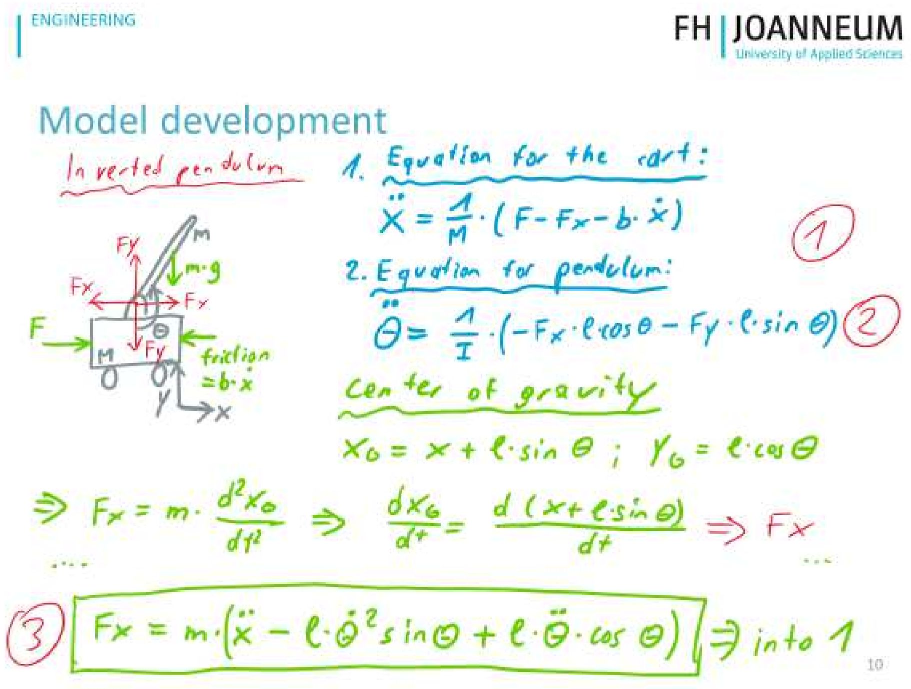
$\boxed{F = m \cdot a = m \cdot \frac{dv}{dt}}$

2. Equation for pendulum:

$T = F \cdot r$ ✓ $\quad T = I \cdot \alpha$

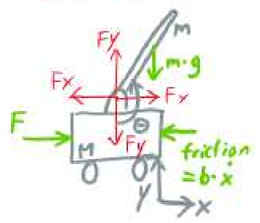
Trigonometry ?

$$\ddot{\Theta} = \alpha = \frac{1}{I} \Sigma T = \frac{1}{I} \cdot (-F_x \cdot \ell \cos\Theta - F_y \cdot \ell \cdot \sin\Theta) \quad (2)$$



Model development

Inverted pendulum



1. Equation for the cart:

$$\ddot{x} = \frac{1}{M}\cdot\left(F - F_x - b\cdot\dot{x}\right) \quad (1)$$

2. Equation for pendulum:

$$\ddot{\Theta} = \frac{1}{I}\cdot\left(-F_x\cdot l\cdot\cos\Theta - F_y\cdot l\cdot\sin\Theta\right) \quad (2)$$

Center of gravity

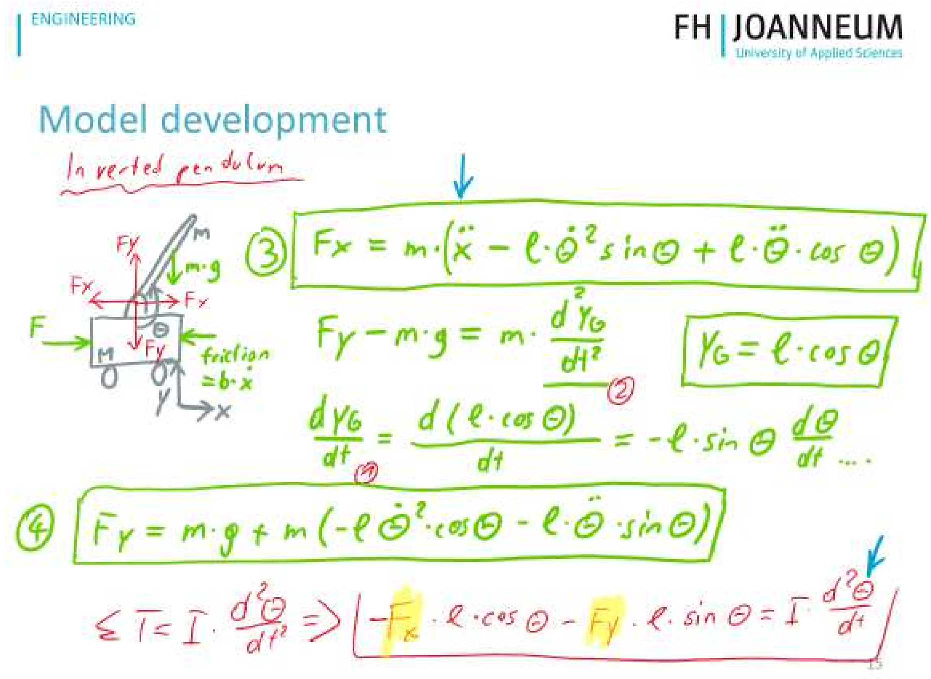
$$x_G = x + l\cdot\sin\Theta \;;\quad y_G = l\cdot\cos\Theta$$

$$\Rightarrow F_x = m\cdot\frac{d^2x_G}{dt^2} \Rightarrow \frac{dx_G}{dt} = \frac{d(x + l\cdot\sin\Theta)}{dt} \Rightarrow F_x$$

... ...

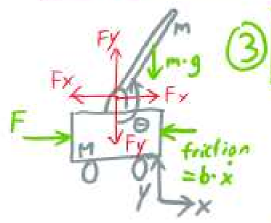
$$(3)\ \boxed{F_x = m\cdot\left(\ddot{x} - l\cdot\dot{\Theta}^2\sin\Theta + l\cdot\ddot{\Theta}\cdot\cos\Theta\right)} \Rightarrow \text{into } 1$$

10



Model development

Inverted pendulum



$$(3)\ \boxed{F_x = m\cdot\left(\ddot{x} - l\cdot\dot{\Theta}^2\sin\Theta + l\cdot\ddot{\Theta}\cdot\cos\Theta\right)}$$

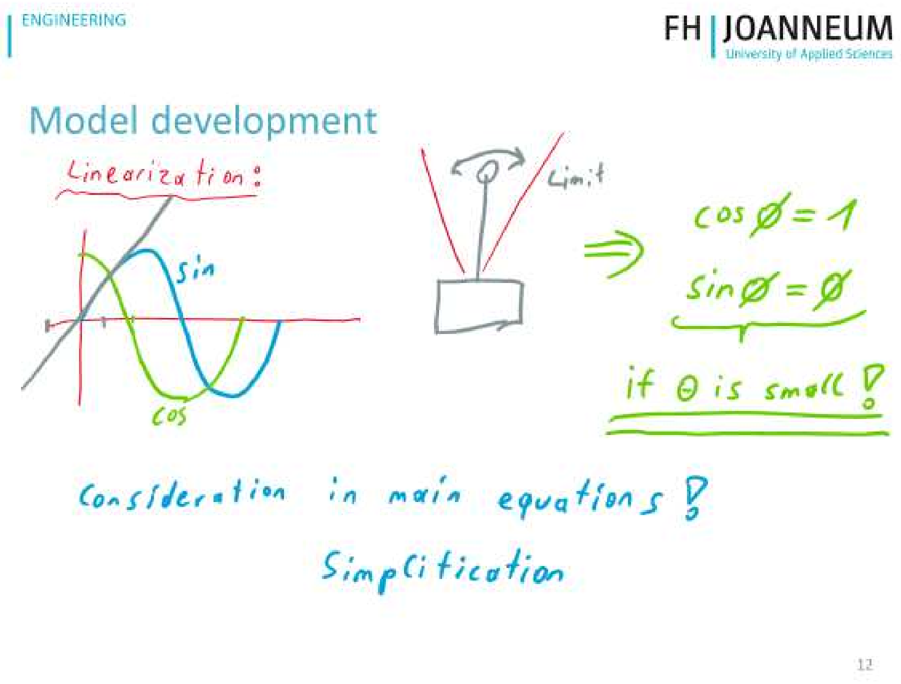
$$F_y - m\cdot g = m\cdot\underbrace{\frac{d^2y_G}{dt^2}}_{(2)} \qquad \boxed{y_G = l\cdot\cos\Theta}$$

$$\underbrace{\frac{dy_G}{dt}}_{(1)} = \frac{d(l\cdot\cos\Theta)}{dt} = -l\cdot\sin\Theta\,\frac{d\Theta}{dt} \ldots$$

$$(4)\ \boxed{F_y = m\cdot g + m\left(-l\,\dot{\Theta}^2\cdot\cos\Theta - l\cdot\ddot{\Theta}\cdot\sin\Theta\right)}$$

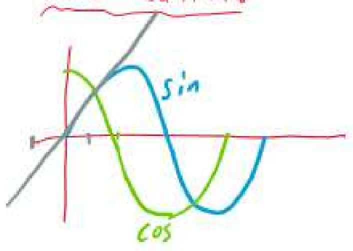
$$\sum T = I\cdot\frac{d^2\Theta}{dt^2} \Rightarrow \boxed{-F_x\cdot l\cdot\cos\Theta - F_y\cdot l\cdot\sin\Theta = I\cdot\frac{d^2\Theta}{dt}}$$

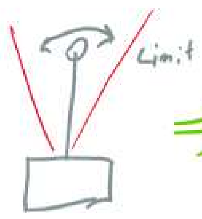
13



Model development

Linearization:





$$\Rightarrow \underbrace{\begin{matrix}\cos\phi = 1\\ \sin\phi = \phi\end{matrix}}_{\text{if } \theta \text{ is small!}}$$

Consideration in main equations!

Simplification