Modeling, Simulation and Control of Legged Robots Using MATLAB/Simulink and MuJoCo Physic Engine

Prof. Dr. Hakan Temeltaş
Istanbul Technical University
temeltash@itu.edu.tr

Physics Engines

- a software tool used to model and simulate the physical properties and interactions various environments.
- mimic real-world physical principles (such as kinematics, dynamics, collision responses) to predict, analyze, and optimize how robots and algorithms will perform in a virtual environment.

- There are a several physics engine for robotics. Some of these are:
  - Mujoco
  - Gazebo
  - Webots
  - CoppeliaSim (V-REP)

- **MuJoCo(Multi-Joint dynamics with Contact)** is the one of the popular physics engine

Advantages of MuJoCo:

- **Speed**: MuJoCo operates quite efficiently and quickly across various physical simulations, making it a suitable option for large-scale simulations and machine learning experiments
- **Soft and Hard Contact Modeling**: This engine can effectively simulate various kinds of collision and contact scenarios, which allows for the creation of realistic interaction scenarios for robots
- **User-Friendly XML Syntax**: MuJoCo allows users to define robotic systems and environments in an XML format, providing the ability to easily create and modify systems
- **API Support**: offers integration possibilities with Python and C++ APIs, making it accessible and applicable for a broad user base.

⇒ this platform proves highly beneficial for simulating the locomotion of legged robots.

Advantages of MuJoCo

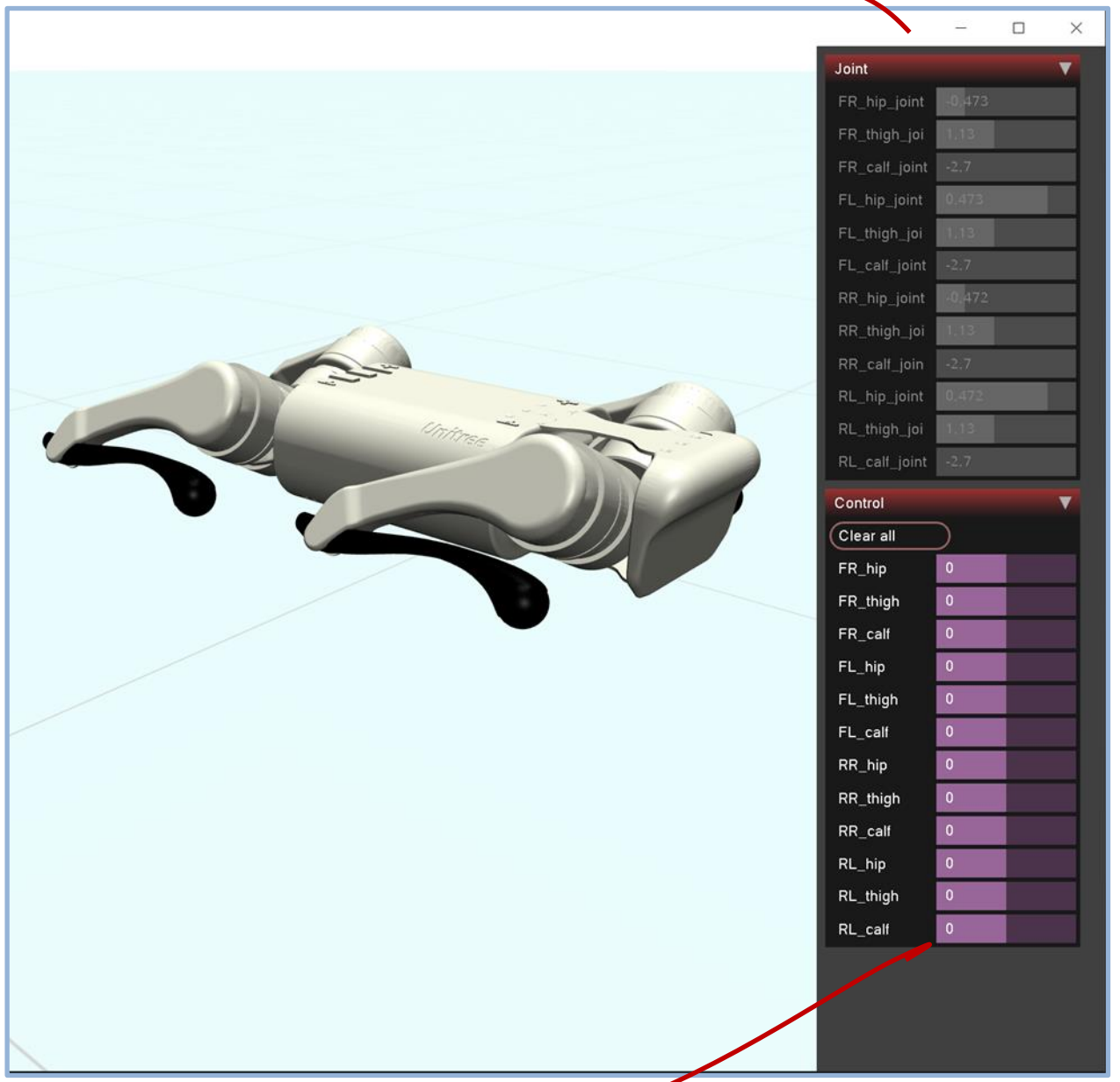
System Under Simulation

Inputs:

- Force/Torque
- Joint positons
- Joint Velocities

Outputs:

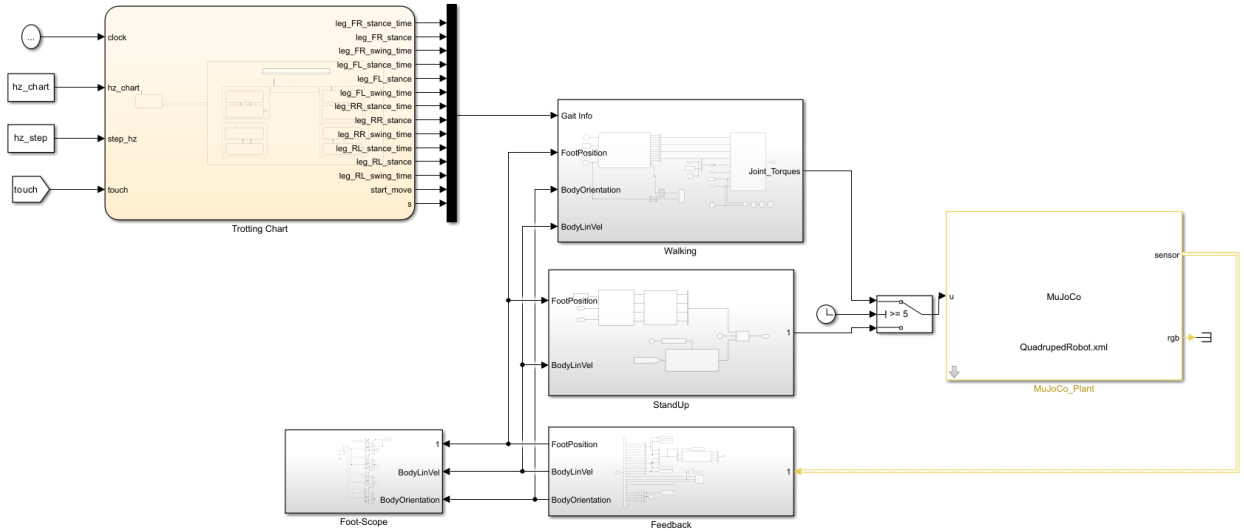
- Joint Position Sensor
- Joint Velocity Sensor
- Accelerometer
- Gyro
- Magnetometer
- Touch Sensor
- Force Sensor
- Torque Sensor

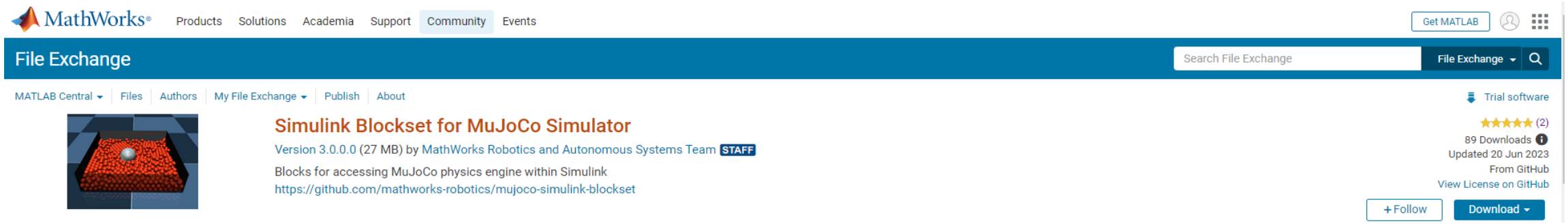


Benefits / added value of using MATLAB\Simulink

With MuJoCo Simulink API,

- The ease of designing control algorithms provided by Simulink.
- Simulation advantages of MuJoCo.





Modeling of Legged Robots (Quadrupeds)

- The equations of motion:

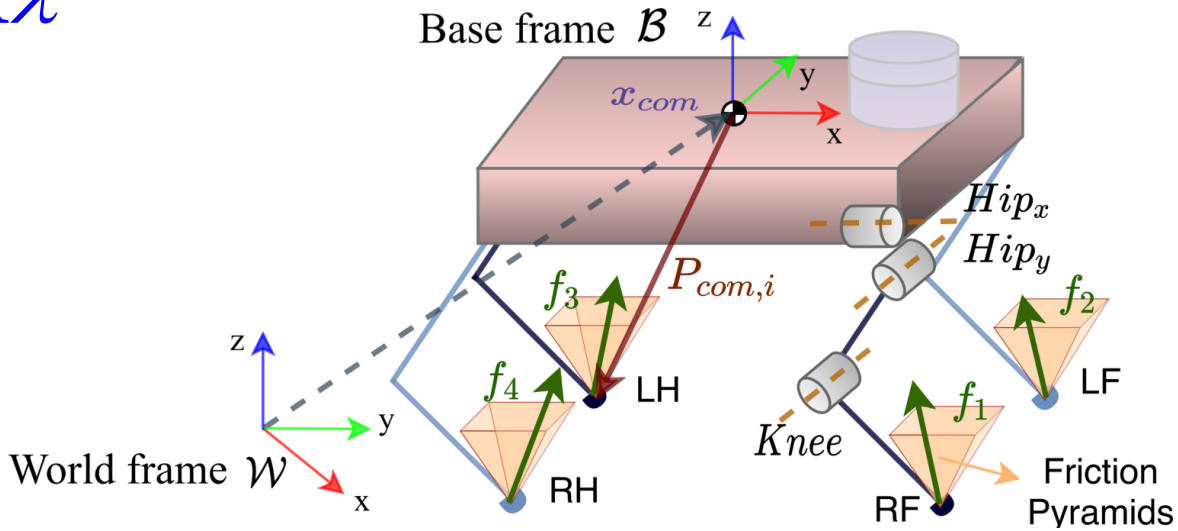
$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q},\mathbf{u}) = \mathbf{S}^T\tau + \mathbf{J}_s^T\lambda$$

- project the equations of motion to the contact constrained and unconstrained spaces

$$\mathbf{Q}_c^T[\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q},\mathbf{u})] = \mathbf{Q}_c^T\mathbf{S}^T\tau + \mathbf{R}\lambda$$

$$\mathbf{Q}_u^T[\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q},\mathbf{u})] = \mathbf{Q}_u^T\mathbf{S}^T\tau$$

- Mujoco meets this need in an optimal way using its Soft and Hard Contact Modelling.



Virtual Model Control of Quadruped Robot

- Virtual Model Control (**VMC**): the application of springs and dampers,
- Computation of swing leg forces: mechanical elements are integrated at the end of each leg.

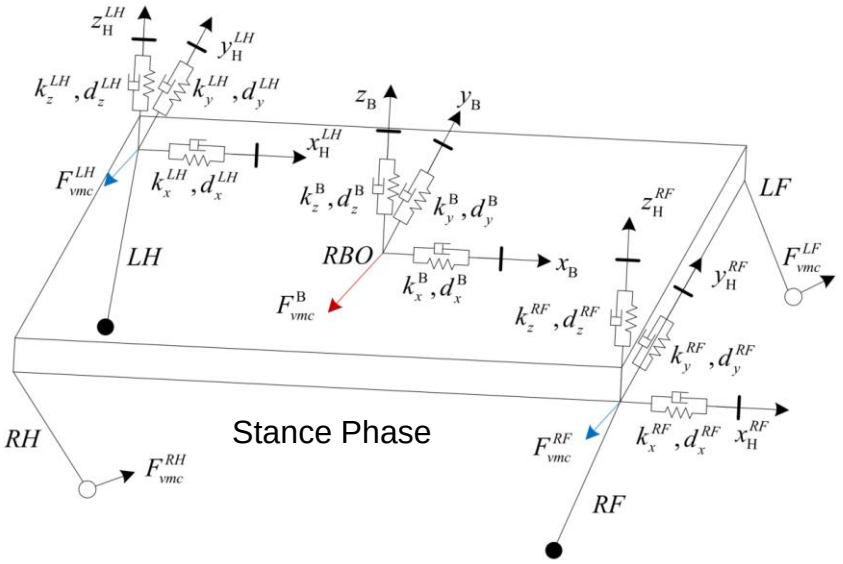
$$F_{vmc}^{sw} = \begin{bmatrix} F_x^{sw} \\ F_y^{sw} \\ F_z^{sw} \end{bmatrix} = \begin{bmatrix} k_x^{sw} & 0 & 0 \\ 0 & k_y^{sw} & 0 \\ 0 & 0 & k_z^{sw} \end{bmatrix} \cdot \begin{bmatrix} x_d^{sw} - x^{sw} \\ y_d^{sw} - y^{sw} \\ z_d^{sw} - z^{sw} \end{bmatrix} + \begin{bmatrix} d_x^{sw} & 0 & 0 \\ 0 & d_y^{sw} & 0 \\ 0 & 0 & d_z^{sw} \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_d^{sw} - \dot{x}^{sw} \\ \dot{y}_d^{sw} - \dot{y}^{sw} \\ \dot{z}_d^{sw} - \dot{z}^{sw} \end{bmatrix}$$

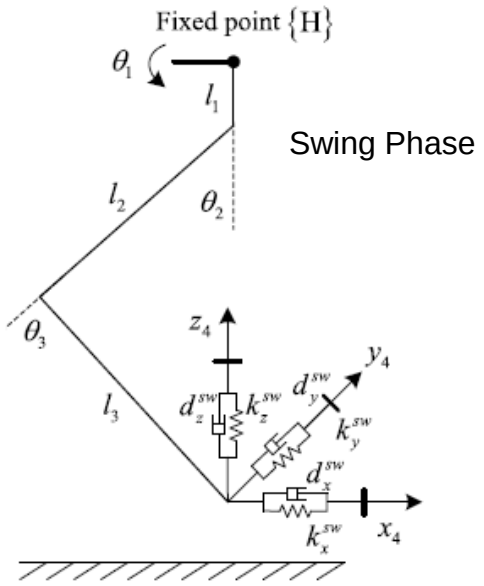
$$\tau^{sw} = J^T F_{vmc}^{sw}$$

- Calculating stance leg forces: mechanical elements are positioned at the center of the trunk.

$$F_{vmc}^{sp} = \begin{bmatrix} F_{ix}^{sp} \\ F_{iy}^{sp} \\ F_{iz}^{sp} \end{bmatrix} = \begin{bmatrix} k_x^{sp} & 0 & 0 \\ 0 & k_y^{sp} & 0 \\ 0 & 0 & k_z^{sp} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ z_d - z \end{bmatrix} + \begin{bmatrix} d_x^{sp} & 0 & 0 \\ 0 & d_y^{sp} & 0 \\ 0 & 0 & d_z^{sp} \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_d - \dot{x} \\ \dot{y}_d - \dot{y} \\ \dot{z}_d - z \end{bmatrix} + \mu \cdot \begin{bmatrix} k_\gamma(\gamma_d - \gamma) + d_\gamma(\dot{\gamma}_d - \dot{\gamma}) \\ 0 \\ 0 \end{bmatrix}$$

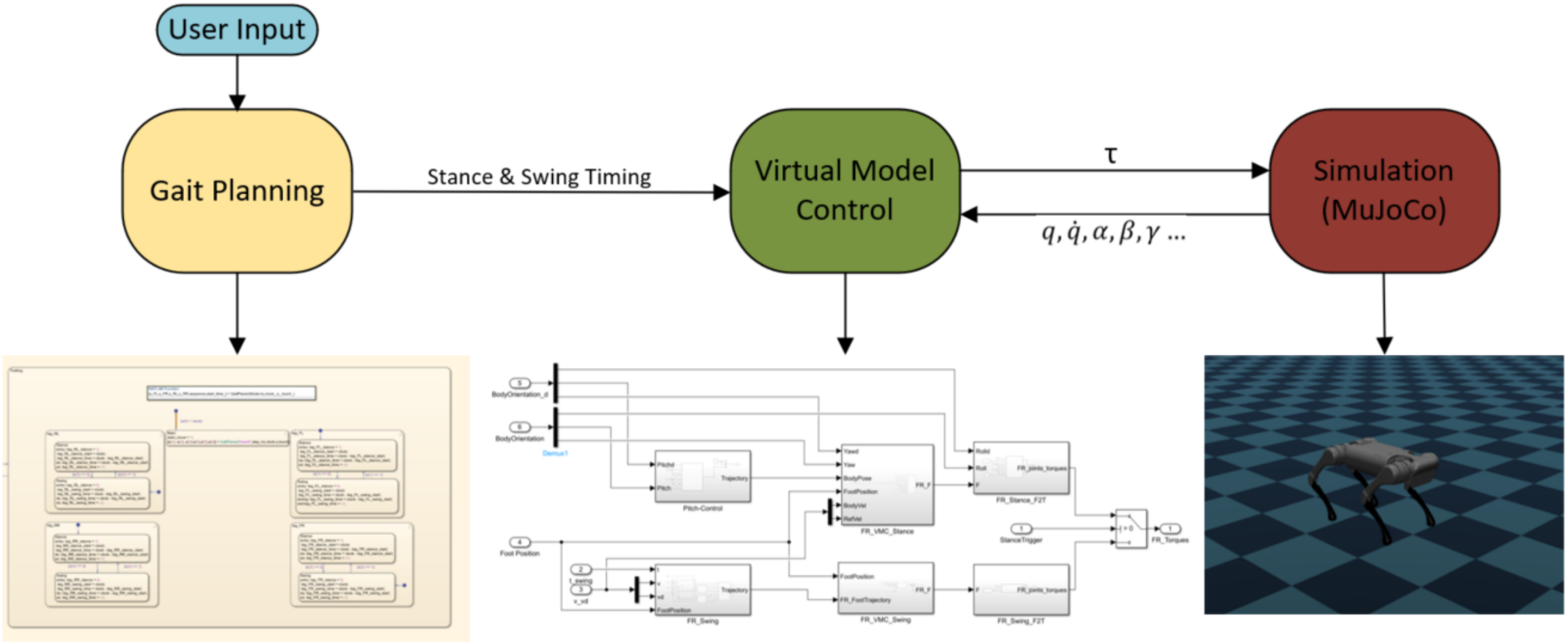
$$\tau^{sp} = -J_i^T F_i^{sp} - \begin{bmatrix} k_\alpha(\alpha_d - \alpha) + d_\alpha(\dot{\alpha}_d - \dot{\alpha}) \\ 0 \\ 0 \end{bmatrix}$$





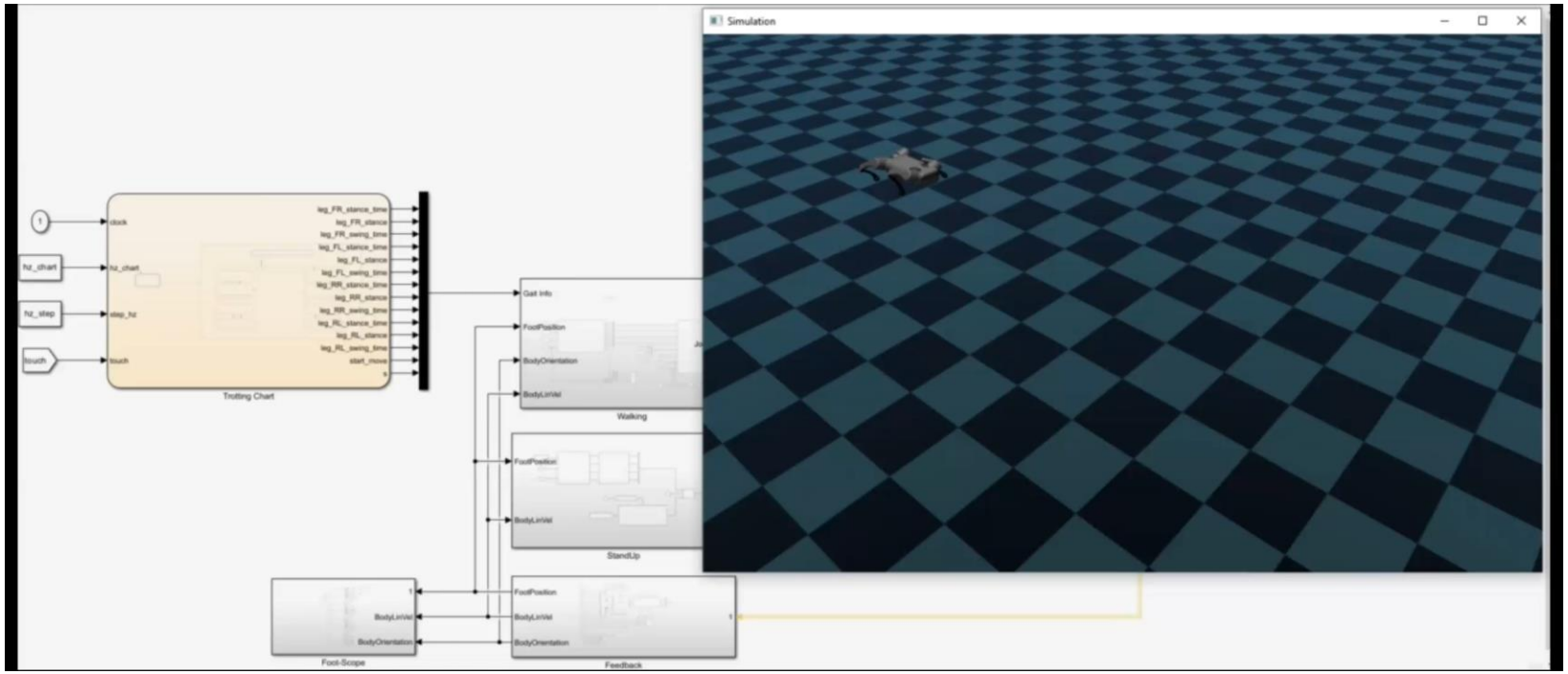
As a result of VMC: MATLAB / SIMULINK optimize the system's control mechanisms and enhance dynamic stability

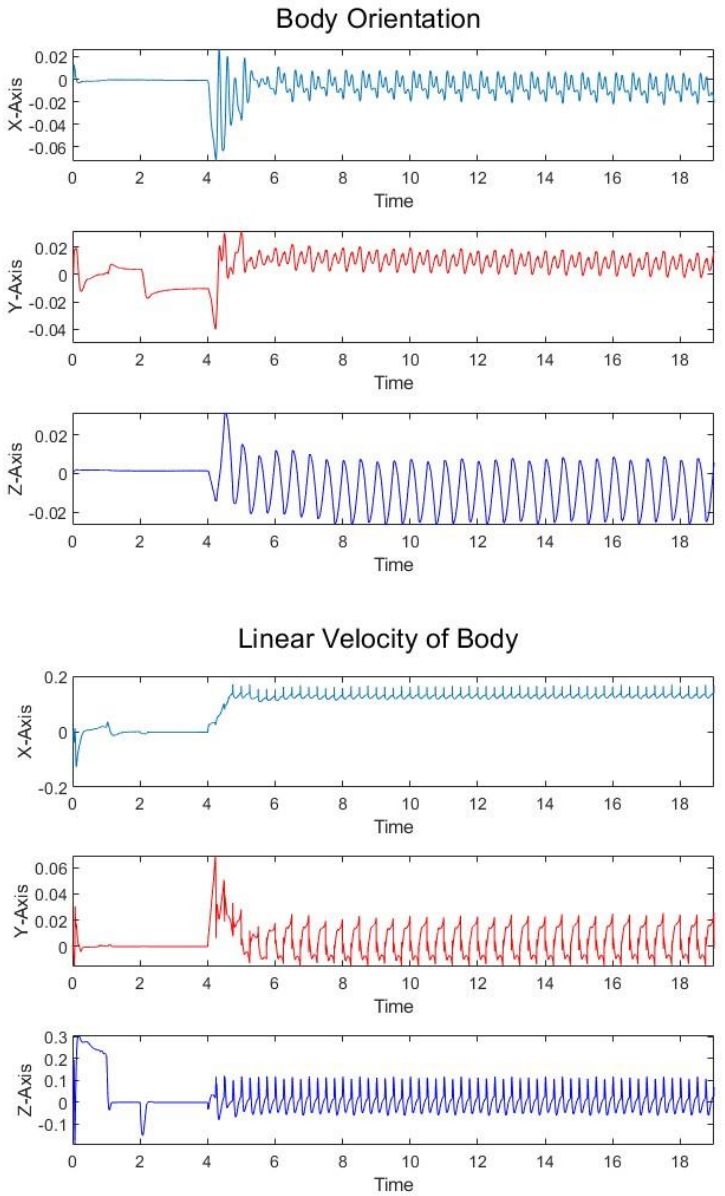
Work Flow Chart



Co-Simulation with Matlab\Simulink-MuJoCo

- Thanks to Matlab\Simulink and MuJoCo Co-Simulation feature ⇒ Robot Motion Planning and Control algorithms can be designed & tested practicaly.

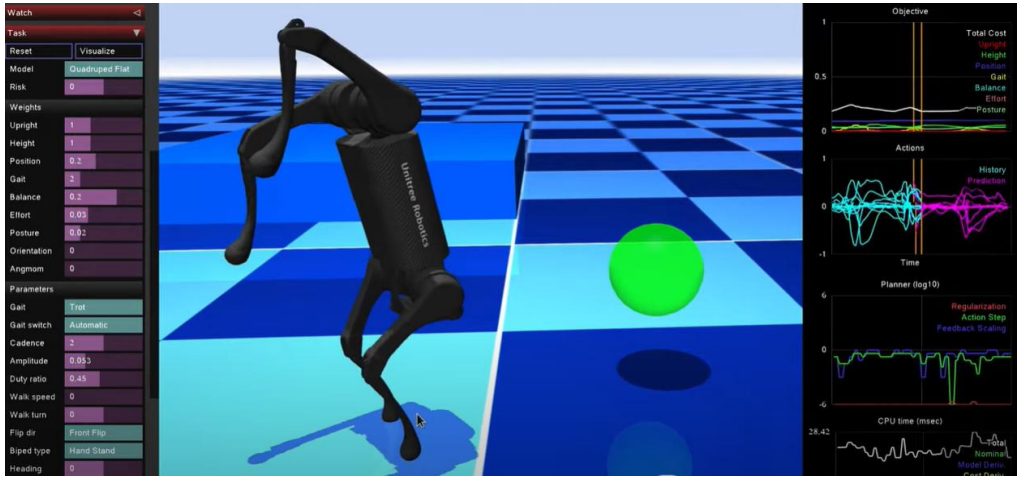




Future Plans

Advanced Motion Planning & Control

- **MPC-NMPC** (MATLAB Toolboxes) for Agile Locomotion of Quaqrupeds Robot
- **AI-Based Locomotion** (requires MATLAB Machine & Deep LearningToolboxes)
- **Autonomous Navigation** (requires MATLAB Navigation, ROS, Point Cloud toolboxes for)
- **Real Time Implementation and HIL simulations** (requires MATLAB & SIMULINK Embedded Coders)
- Extensions for **HUMANOID robots** studies.



Thank you

Q&A – 5min

Prof. Dr. Hakan Temeltaş
Istanbul Technical University
temeltash@itu.edu.tr