CONTRIBUTIONS TO THE MODELING AND SIMULATION OF A BIPED WALKING ROBOT

Daniel Mircea Popescu[1], Liviu Marian Ungureanu[2] and Cătălin Gheorghe Amza[3]
[1]National University of Science and Technology POLITEHNICA Bucharest, 313 Splaiul Independenței, Bucharest, Romania,
ORCID No. 0009-0008-0333-415X, danielmircea.popescu@upb.ro
[2]National University of Science and Technology POLITEHNICA Bucharest, 313 Splaiul Independenței, Bucharest, Romania,
[3]National University of Science and Technology, Faculty of Industrial Engineering and Robotics,
Corresponding author, ORCID No. 0000-0002-6621-5034, catalin.amza@upb.ro

This study presents the modeling and simulation of a biped walking robot using MathWorks® simulation tools. The project focuses on developing a parametric and dynamic model that accurately represents the mechanical structure, joint kinematics, and actuator behavior of a bipedal robot. The methodology involves 3D modeling of the robot's components, simulation of foot–ground contact forces, and comparative analysis of different actuation strategies—motion-based, torque-driven, and motor-based systems. By integrating mechanical, electrical, and control subsystems, the study evaluates trade-offs between model fidelity and simulation speed. Results indicate that while motor-actuated models provide higher physical realism, they significantly increase computational complexity. The work also highlights the importance of appropriate damping, force modeling, and control architecture (PID) for ensuring stable gait dynamics. Future developments include hardware implementation, enhanced actuator modeling, and machine learning approaches for optimizing walking performance.
KEYWORDS: bipedal robot, walking robot, mechanical structure, foot joints

1. INTRODUCTION

Bipedal robots attempt to implement human locomotion and have been a challenge for robotics research for decades. Bipedal walking represents a top accomplishment of biological engineering, combining the stability that must be provided by the walking surface, the dynamics conferred by the way in which the movement is performed, the energy efficiency of the motors in the joints, and the adaptability to irregular walking surfaces. The development of stable and efficient bipedal robots is paved with multiple challenges, due to their inherent complex nature, which attempts to replicate human joints. To overcome this complexity, a systematic combination between mechanical design, dynamic modelling, and computer simulation has become indispensable.

The process begins with the design phase, which establishes the physical form and kinematic structure of the proposed bipedal robot. The choices made during the design phase such as the link lengths, mass distribution, joint configuration, and selection of the type of motor that performs the actuation are profoundly influencing the inherent dynamic capabilities of a walking robot. A well-designed morphology can simplify control by promoting passive stability, as in the case of passive dynamic robots descending a slope without active control. In contrast, poor design can create inherent instabilities that are impossible to overcome even with the most sophisticated software-based control algorithms.

One can use high-fidelity modelling and simulation that are for many years facilitated by tools such as Mathworks Matlab's Symbolic Math Toolbox.

Using the Symbolic Math Toolbox in Mathworks Matlab, the port-Hamiltonian equations [25] of the system, which naturally represent storage, dissipation, and exchange of energy, were derived by the authors of [1]. The simulation demonstrated superior performance in analyzing the passivity and energy coupling of the system, providing a structured framework for modeling complex interconnected systems.

In [2], using the System Identification Toolbox in Mathworks Matlab, Patel and Jones created data-driven models of a bipedal robot from experimental data, going beyond idealized modeling based on fundamental principles. By applying input torques and measuring joint angles, they have identified linear ARX and nonlinear Hammerstein-Wiener models [24]. These models were used for controller design, demonstrating improved performance of the system.

Bridging robotics and biomechanics, a sophisticated neuromuscular model of human gait was developed in Mathworks Matlab [3]. The proposed model includes musculotendinous actuators, reflex loops, and a central pattern generator. Simulations were used to investigate neural control strategies.

Modeling of electromechanical actuators for a bipedal robot was performed using Simscape Electrical and Simscape Multibody in [4]. The proposed simulation revealed critical effects of motor saturation, bandwidth limitations, and electrical losses on walking performance, demonstrating the importance of modeling the interaction between the electrical and mechanical domains.

Another type of modeling addressed the problem of inaccurate robot model parameters using the Global Optimization Toolbox in Mathworks Matlab as in [5]. Genetic algorithms were used to identify key dynamic parameters by minimizing the errors between simulated and experimental data, followed by a sensitivity analysis to guide the mechanical design.

To model walking on difficult surfaces, Mathworks Matlab/Simulink (robot dynamics) and a Finite Element Method software were used in [6]. The interaction forces were changed in real time, allowing for a highly realistic simulation.

The design and simulation of a Model Predictive Controller (MPC) for a biped robot using the Model Predictive Control Toolbox in Mathworks Matlab/Simulink [7-12] was detailed from a dynamic point of view, achieving real-time performance and demonstrating the feasibility of advanced, computationally intensive controllers for dynamic gait.

Trajectory optimization of bipedal robots can also be studied using Matlab [13-15]. The toolkit allows users to define robot parameters, constraints, and cost functions to solve optimal gait problems, proving effective for rapid gait prototyping and serving as a benchmarking tool for researchers.

Control and simulation of locomotion using the Computer Vision Toolbox in Mathworks Matlab for obstacle detection and terrain classification [15-20]. Parameters such as accuracy, calculation speed of the stability of walking robots as well as continuous monitoring provide practical guidance for selecting the types of actuators that can be integrated into the locomotion structure.

As a conclusion, Mathworks Matlab offers the suitable tool for modeling and simulating the locomotion of a biped walking robot.

The objective of this work is to determine whether improvements and corrections of errors in the existing model are possible, leading to a more natural gait in the simulation of locomotion.

2. MODELING AND SIMULATION OF A BIPED WALKING ROBOT

In this article, Mathworks Matlab was used to model and simulate a bipedal walking robot with 4 movement steps [21].

Locomotion is ensured by a spherical joint at the hip, a cylindrical joint at the knee, and two other cylindrical joints at the ankle. The joint is the connection between two limbs (ankle, knee) or the torso and and a limb (hip) [24], as shown in the picture 1. The joints are known also as kinematic pairs in the theory of mechanisms.

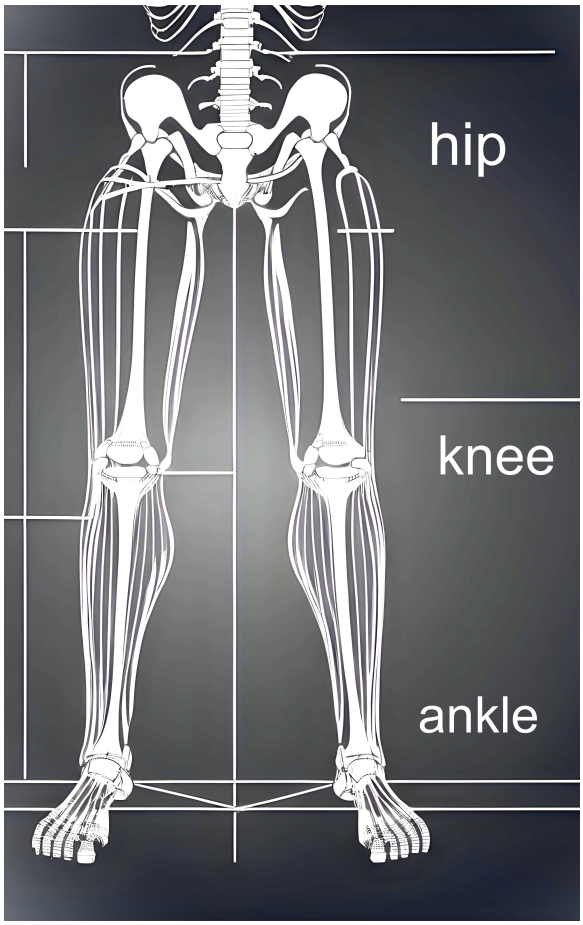


Figure 1. Leg joints

The starting point is a biped walking robot model developed by using the MathWorks® software. Each of the modules and libraries included in this software plays its own role in achieving this scope:

- *Simscape Multibody:* modeling the mechanics of a rigid body of a walking robot.
- *Simulink®:* modeling closed-loop torque controllers.
- *Simscape Electrical™:* defining the dynamic parameters of the actuators
- *Mathworks Matlab®:* writing scripts and saving working parameters

The main steps are:

- 3D modeling of the robot mechanisms

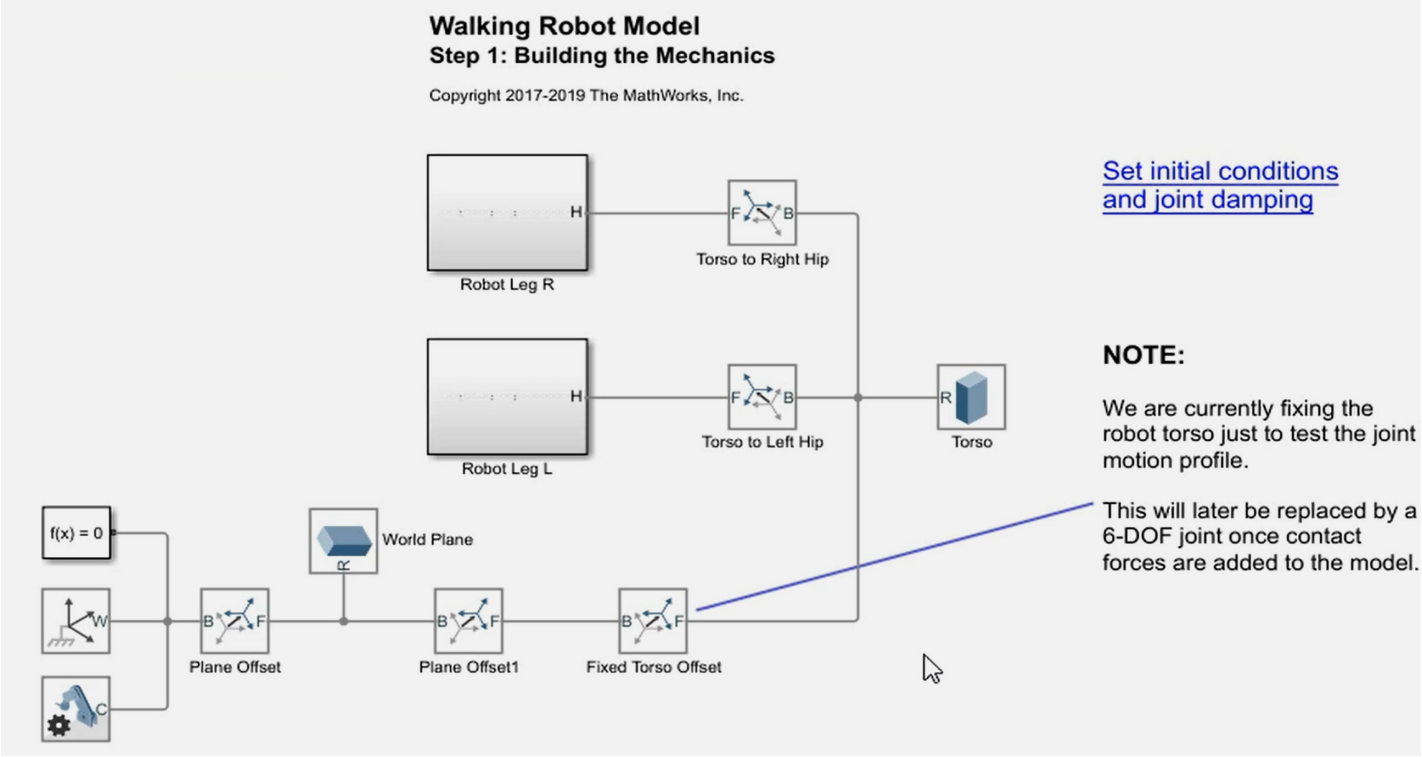


Figure 2. Building the mechanics

- adding motion actuators
- adding contact forces

2.1 3D modeling of the robot mechanisms

According to the literature, 3D modelling of the robot mechanism can be performed in two ways:

a) Using Simscape Multibody®

The advantage in this case is that the kinematic elements can be parameterized, making it easy to modify them later in order to optimize the mechanism.

b) Models can be imported from CAD or URDF files using the *smimport* command. CAD files serve many general design purposes, whereas URDF (Unified Robot Description Format) files are XML-based and tailored for robot modeling.

The first approach was selected because, unlike imported elements, components modeled with Simscape Multibody® can be easily parameterized. Their parameters are stored in MathWorks Matlab files, making it straightforward to modify their values.

In Figure 2, one can notice the structure made up of blocks that on the left side have the input blocks (from top to bottom):

- Solver Configuration
- World Frame
- Mechanism Configuration

Also take note of the blocks that describe the solid components, such as the two legs: robot Leg L and Robot Leg R.

In turn, each of these components is made up of blocks, as depicted in Figure 3, that describe the component elements as well as their relative movements:

a) Solid components: sole, calf of the foot (lower part), thigh of the foot (upper part)

b) Rigid transformations

c) Joints: two-component ankle joint (Ankle Roll Joint and Ankle Pitch Joint); knee joint (Knee Joint); hip joint with three components (Hip Pitch Joint, Hip Roll Joint and Hip Yaw Joint)

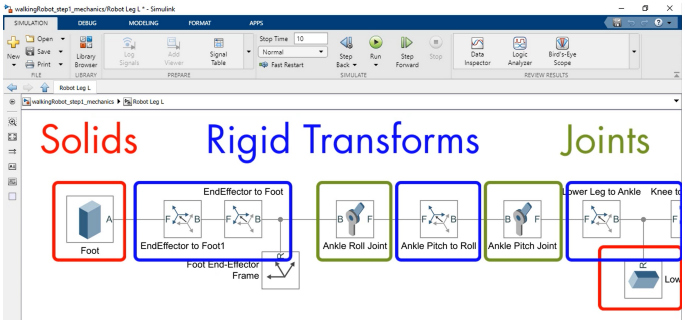


Figure 3. Leg block detail

System Description and Modeling

Each block within the simulation includes the description and parameterization of the corresponding component.

For example, the **Foot** component block illustrated in Figure 4 defines the sole's dimensions, which are modeled as a parallelepiped.

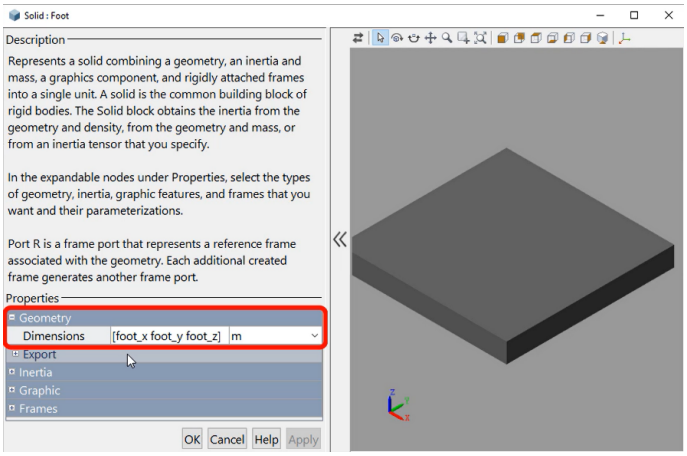


Figure 4. Leg block detail

Regarding the joints (kinematic pairs), the Simscape Multibody library provides a wide range of predefined elements as shown in Figure 5.

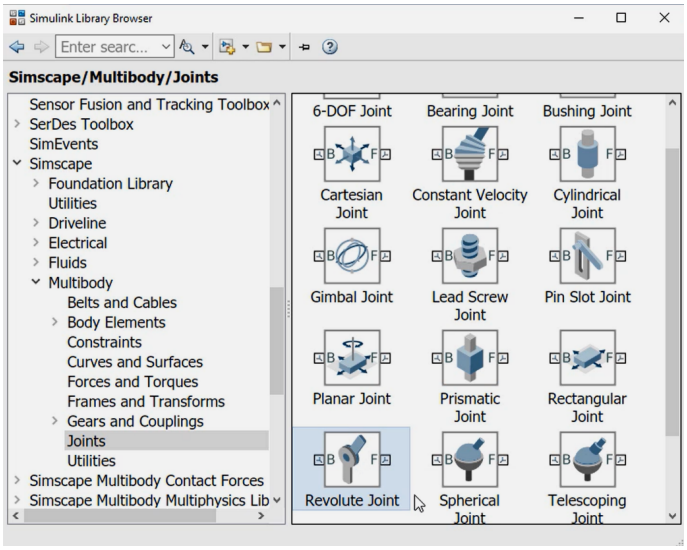


Figure 5. Predefined joints library

After selecting an element from the library, its operational parameters are defined accordingly.

In the Simulink graphical interface, the system is represented as interconnected blocks describing physical components, interactions, and motion dynamics.

All parameters defined within the graphical interface are stored in a Mathworks Matlab® file associated with the project, as can be seen in Figure 6.

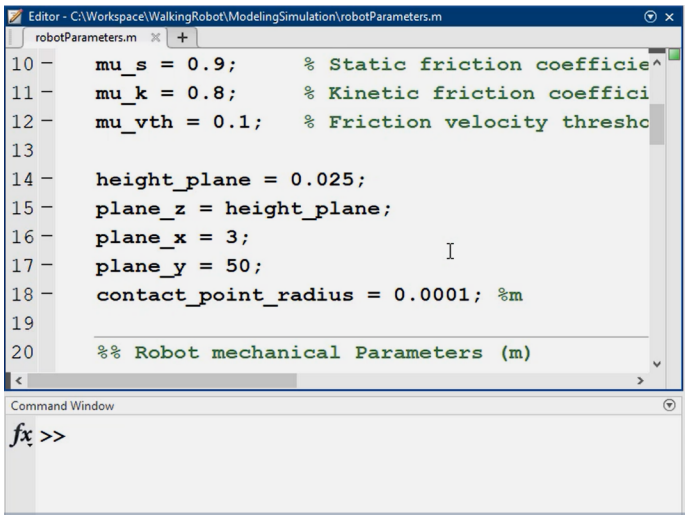


Figure 6. Parameters saved in Mathworks Matlab

The working variables are saved in a binary file with the .mat extension, which can be accessed and visualized as required.

For instance, the six joint angles of the right leg can be plotted to observe motion profiles. The Figure 7 illustrates the variation of the six joint angles during the three steps motion.

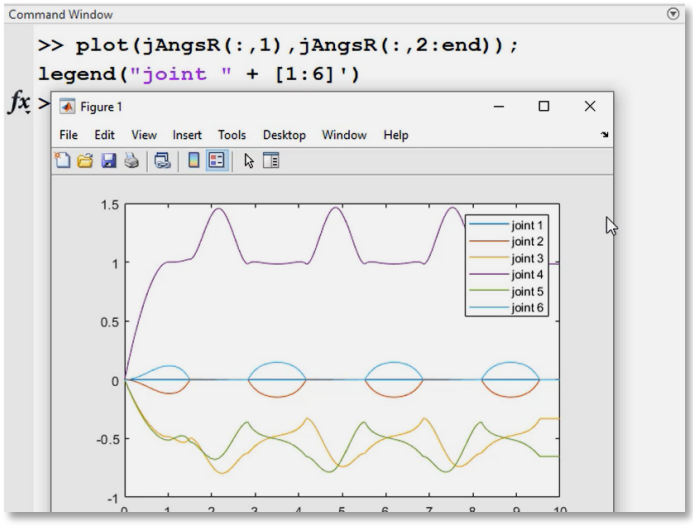


Figure 7. Ploting the angles of the right leg joints

The next stage involves setting the initial conditions and joint damping.

Special attention must be given to the damping parameters, as excessively low values lead to unstable or chaotic motion of the mechanism.

The Figure 8 presents the adjustment of the damping parameters with effect on reducing the bouncing movement.

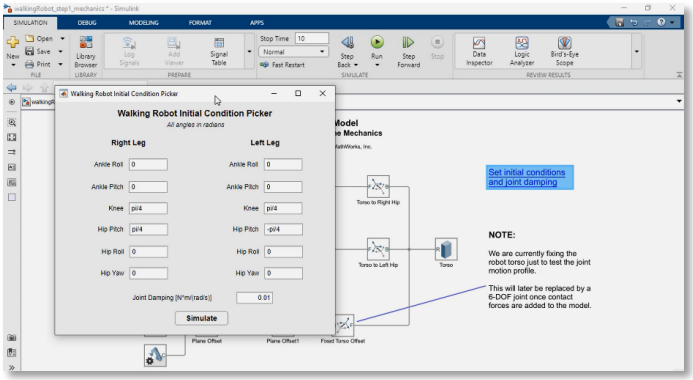


Figure 8. Adjustment of the damping parameters

2.2 Motion Actuation

As previously mentioned, six joints are defined for each leg, corresponding to the six angular variables described above. Figure 9 shows the Walking Robot subsystem block with input and output signals.

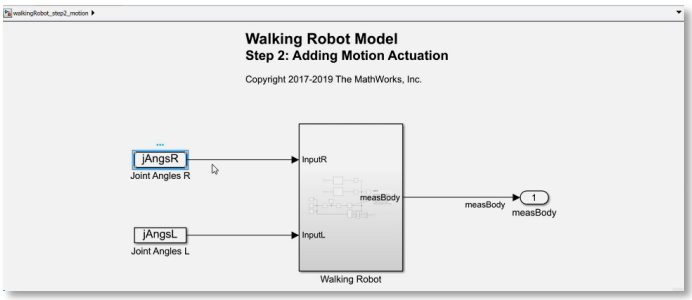


Figure 9. Adding motion actuation

Consequently, the respective subsystem block includes six input signals, as depicted in Figure 10.

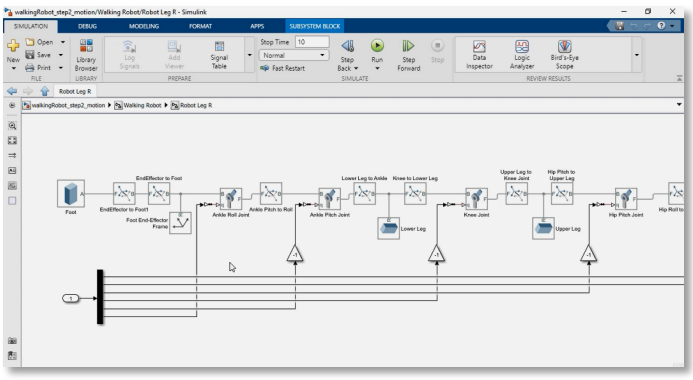


Figure 10. Input signals for each leg

In addition to the internal joint motions, the robot also performs a global motion relative to the walking surface. This is possible only if a frictional force exists between the sole and the ground.

Friction is modeled using four spheres placed at the corners of each sole, with the friction force at each contact point proportional to the local normal force.

For each contact point, the parameters of the spatial contact force are defined as follows:

a) Normal Force:

- Contact stiffness
- Contact damping

b) Friction Force:

- Static friction coefficient
- Dynamic friction coefficient
- Critical velocity

c) Detection Parameters

The selection of these parameters depends on both the robot and the environment. For example, contact stiffness is influenced by the robot's weight, whereas friction coefficients depend on the materials of the sole and the walking surface. The four spheres placed at the corners of the foot are used to model the foot contact, as shown in Figure 11.

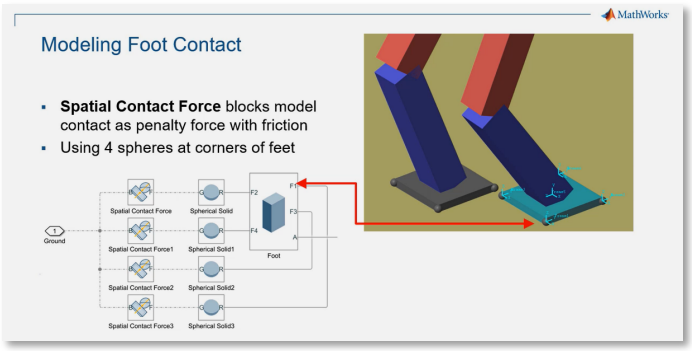


Figure 11. Modeling foot contact

2.3 Adding Contact Forces

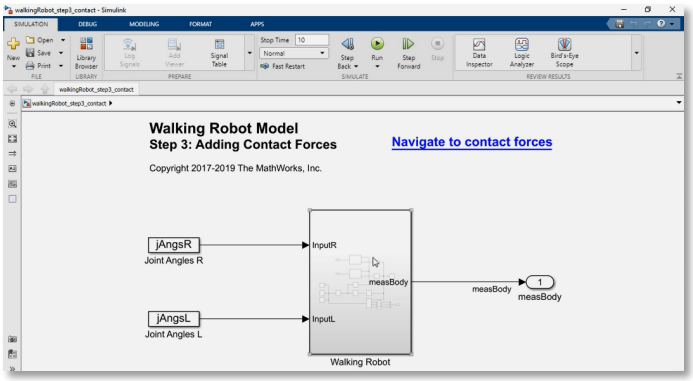


Figure 12. Adding contact forces

The contact forces are the normal forces to the soil and the friction forces, four forces for each foot. The step is seen in Figure 12.

In this stage, the human body, initially modeled as a Fixed Torso Offset in Step 1, is redefined as a 6-DOF Joint to enable motion during walking, as depicted in Fgure 13.

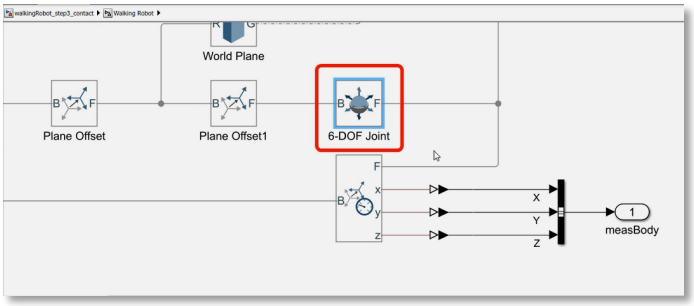


Figure 13. Torso redefined as 6-DOF Joint

6-DOF = Six Degrees of Freedom

Types of Actuation

Up to this point, the model has employed motion-based actuation.

An alternative approach is to define force or torque-based actuation. The both types are shown in Figure 14.

Motion Actuation:

- Requires the automatic computation of the corresponding joint torque.
- Requires a defined motion profile and its derivatives (velocity and acceleration).
- Infinite acceleration values are not permitted.

Force/Torque Actuation:

- Requires a defined force or torque profile.
- Positioning controllers must be implemented to ensure trajectory tracking.

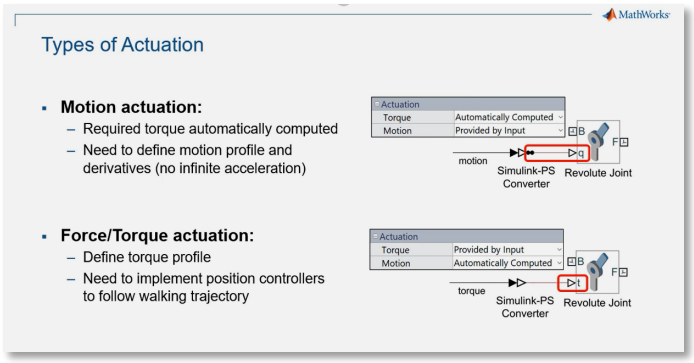


Figure 14. Types of actuation

Each approach has specific input configurations in Simulink. The complete walking robot model is depicted in Figure 15.

The selection of the actuation type influences subsequent calculations and control strategies.

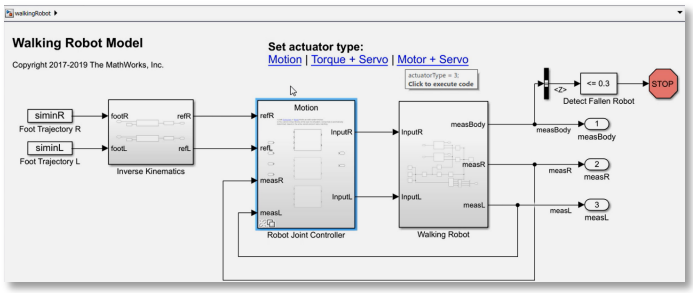


Figure 15. Walking robot model

a) Motion Actuation Mode

When the Motion option is selected, clicking on the Revolute Joint opens a property window where motion is defined as the input, and torque is computed automatically. Figure 16 shows how this actuation mode works with motion input and torque output.

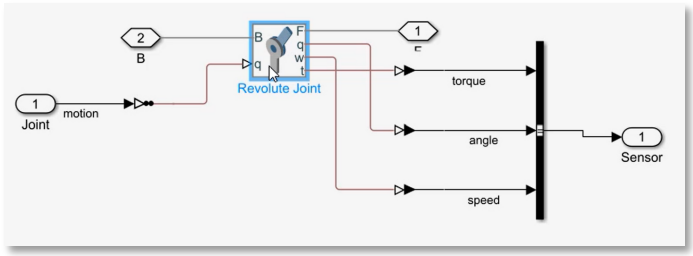


Figure 16. Motion actuation

In this configuration, motion is imposed as an input parameter, and controllers are unnecessary since no corrective feedback is required, as shown in Figure 16.

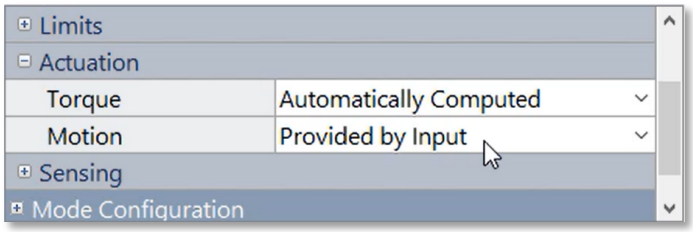


Figure 17. Motion is the input parameter

b) Torque + Servo Mode

Selecting the Torque + Servo option introduces PID controllers for motion regulation of each joint, as presented in Figure 18.

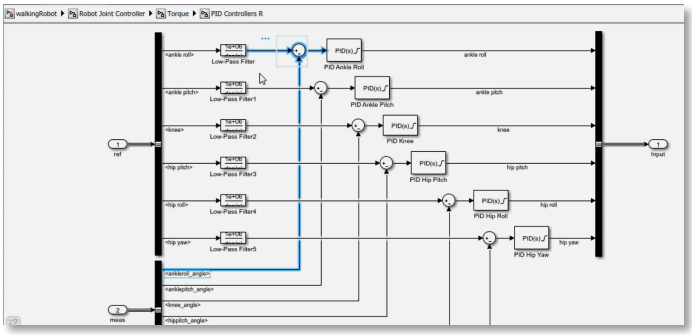


Figure 18. Torque + servo actuation

For example, in the right ankle joint (Ankle Pitch Joint), the configuration is similar to the motion-actuated case, but the computation direction changes, as torque becomes the input variable. The updated Properties windows is shown in Figure 19.

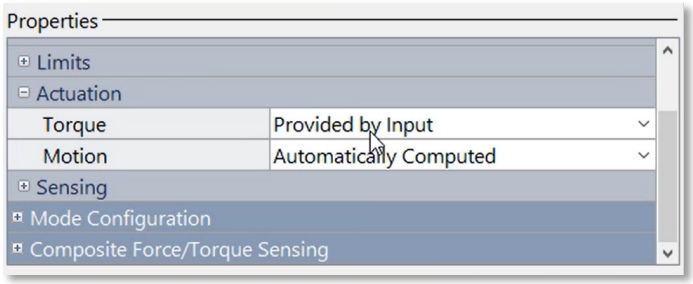


Figure 19. Torque is the input parameter

d) Motor + Servo Mode

In this case, Simscape® integrates components from the electronics libraries, modeling the Motor subsystem as a mechatronic system, as seen in Figure 20.

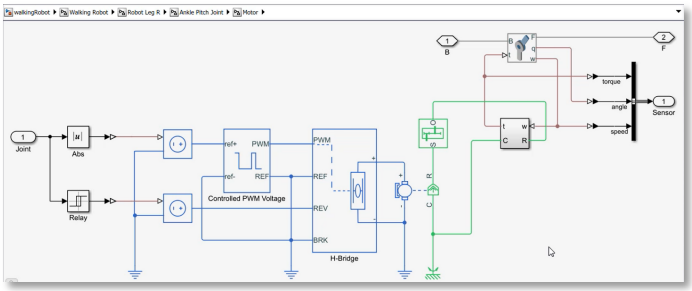


Figure 20. Motor + Servo actuation

The DC motor is powered through an H-bridge, enabling bidirectional rotation.

The H-bridge receives a Pulse Width Modulated (PWM) control signal, which allows precise adjustment of motor speed.

Both electrical and mechanical parameters of the DC motor can be tuned through the property window.

Comparison of Actuation Types

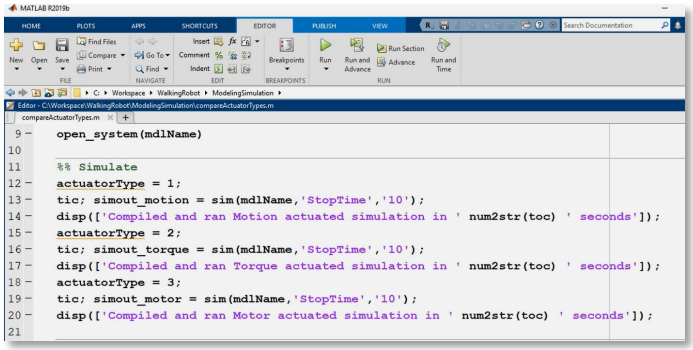
Each actuation approach presents a trade-off between model fidelity and simulation speed.

Variants of actuator models can be developed and compared by running simulations and collecting data using the sim function. The actuators types are coded as per Figure 21.

Name (read-only)	Variant control expression
Motion	actuatorType == 1
Motor	actuatorType == 3
Torque	actuatorType == 2

Figure 21. Motor + Servo actuation

The Mathworks Matlab® script presented in Figure 22 was executed to determine the simulation times for each actuation type:



```
open_system(mdlName)

%% Simulate
actuatorType = 1;
tic; simout_motion = sim(mdlName,'StopTime','10');
disp(['Compiled and ran Motion actuated simulation in ' num2str(toc) ' seconds']);
actuatorType = 2;
tic; simout_torque = sim(mdlName,'StopTime','10');
disp(['Compiled and ran Torque actuated simulation in ' num2str(toc) ' seconds']);
actuatorType = 3;
tic; simout_motor = sim(mdlName,'StopTime','10');
disp(['Compiled and ran Motor actuated simulation in ' num2str(toc) ' seconds']);
```

Figure 22. Mathworks Matlab script for simulation times

Compiled and ran Motion-actuated simulation in 5.4984 seconds

Compiled and ran Torque-actuated simulation in 19.5301 seconds

Compiled and ran Motor-actuated simulation in 43.4575 seconds

Differences among the three configurations are also reflected in the values of certain output variables.

As expected, the simulation time strongly depends on the actuation type as follows:

- The motion actuation simulation requires the shortest time because it is based on direct kinematics.
- The motor + servo actuation requires the longest time for simulation because the PID closed loop need exyended computational resources compared to the other two actuation types.

The simulation time is a key factor influencing the choice of actuation type. Ultimately, the decision maker must balance the higher model fidelity provided by motor and servo actuation against the faster simulation speed offered by motion actuation. The output values displayed during simulation, see Figure 23, offer additional arguments for choosing one of the actuation types.

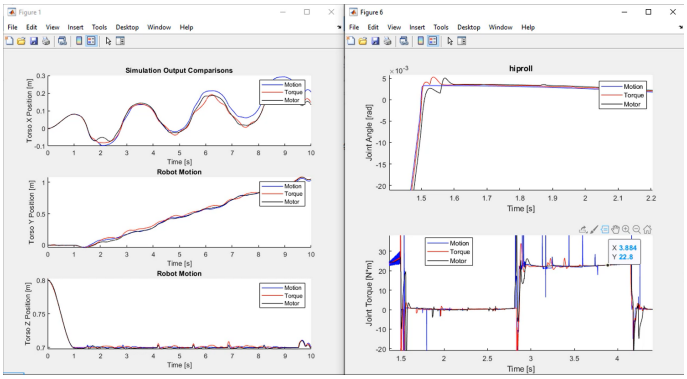


Figure 23. Simulation output values

3. CONCLUSIONS

The simulation was originally developed by MathWorks® as an educational activity that includes several aspects that can be refined to enhance accuracy and realism. During modeling and simulation, the following limitations were identified and addressed:

Foot geometry: The initial square feet caused an unnatural gait. Changing them to rectangular shapes produced a more stable and realistic walking motion—first tested on the right foot, then applied to the left.

Joint damping: Initially low damping values led to oscillatory, unrealistic movements. Increasing the damping coefficients—both at the joints and at the foot–ground interface by about an order of magnitude—yielded motion more consistent with human walking dynamics.

Contact point positioning: The original mid-foot contact points generated unrealistic interactions. Moving them to the base surface ensured realistic surface, line, or point contact depending on motion phase, improving physical accuracy.

Non-linearity iterations: Raising this value from 2 to 4 made the motion smoother.

Link interference: Persistent geometric interferences between certain links could not be resolved within Simscape Multibody. These can only be eliminated by modeling the parts in third-party CAD software for greater geometric precision.

After these improvements, the simulated bipedal robot exhibited a more natural gait, closely approximating human locomotion. Future work will involve developing components in SolidWorks® and importing them into Simscape Multibody to achieve a higher-fidelity walking mechanism.

4. REFERENCES

1. Chen, X., & Zhang, Y. (2025). Modeling and Simulation of a Bipedal Robot with Series Elastic Actuators: A Port-Hamiltonian Approach in MATLAB. *Mechanism and Machine Theory*, 154, 104-120.
2. Wills, A., Schön, T. B., Ljung, L. & Ninness, B. (2013). Identification of Hammerstein–Wiener Models. *Automatica*, 49(1), 70-81.
3. Patel, R., & Jones, D. (2024). Data-Driven System Identification of a Bipedal Robot using MATLAB's System Identification Toolbox. *Control Engineering Practice*, 132, 105-118. This study used MATLAB's System
4. Anderson, B., & Taylor, C. (2022). Simulation of Human-Like Bipedal Walking using a MATLAB-Based Neuromuscular Model. *Journal of Biomechanics*, 134, 110-125.
5. Singh, A., et al. (2024). Co-Simulation of Electromechanical Actuators for a High-Performance Bipedal Robot using Simscape. *Mechatronics*, 88, 102-115.
6. Harris, L., & Clark, K. (2024). Parameter Identification and Sensitivity Analysis of a Bipedal Robot Model using Global Optimization in MATLAB. *Engineering Applications of Artificial Intelligence*, 126, 107-119.
7. Wright, S., & Adams, J. (2024). Simulation of Bipedal Locomotion on Deformable Terrain using a Co-Simulation between MATLAB and a Discrete Element Method. *Granular Matter*, 26(2), 1-15.
8. Nguyen, T., et al. (2023). Model Predictive Control for Bipedal Locomotion: A MATLAB/Simulink Framework for Real-Time Simulation. *IEEE Access*, 11, 45672-45685..
9. Kumar, S., & Park, H. (2023). Real-Time Gait Phase Estimation and Adaptive Control for a Powered Ankle-Foot Prosthesis using MATLAB/Simulink. *IEEE/ASME Transactions on Mechatronics*, 28(1), 210-222.
10. Wang, J., et al. (2023). Hybrid Zero Dynamics Control for a 3D Bipedal Robot: Simulation and Implementation using Simscape Multibody.

International Journal of Humanoid Robotics, 20(04), 225-245.
11. Baker, N., & Hill, G. (2023). Robust H-Infinity Control for a Bipedal Robot using MATLAB's Robust Control Toolbox. *International Journal of Control, Automation and Systems*, 21(3), 889-901.
12. Martinez, C., & Davis, P. (2023). Fault-Tolerant Control of a Bipedal Robot under Actuator Failure: A Simulation Study in MATLAB/Simulink. *Annual Reviews in Control*, 55, 321-335.
13. Perez, M., & Scott, T. (2023). Energy-Based Control and Simulation of Passive Dynamic Walkers in MATLAB. *Nonlinear Dynamics*, 111(2), 1457-1475.
14. Garcia, M., et al. (2022). A MATLAB-Based Optimization Toolbox for Generating Dynamic Bipedal Walking Gaits. *Robotics and Autonomous Systems*, 148, 103-115.
15. Li, W., et al. (2024). A Deep Reinforcement Learning Framework for Robust Bipedal Locomotion on Irregular Terrain using MATLAB and Simulink. *IEEE Transactions on Robotics*, 40(2), 501-517.
16. Evans, R., & Green, M. (2025). Generative Adversarial Networks for Bipedal Gait Synthesis: Training and Evaluation in a MATLAB Environment. *Nature Machine Intelligence*, 7(1), 45-58.
17. Rodriguez, F., & King, E. (2022). From Simulation to Reality: A MATLAB Workflow for Validating Bipedal Robot Designs. *Journal of Field Robotics*, 39(5), 678-695.
18. Thompson, J., & White, S. (2022). A MATLAB-Based Framework for Multi-Robot Bipedal Locomotion and Collaboration. *Robotics and Computer-Integrated Manufacturing*, 74, 102-118.
19. Kim, H., & Brown, D. (2025). Leveraging MATLAB's Computer Vision Toolbox for Vision-Based Bipedal Locomotion over Obstacles. *IEEE International Conference on Robotics and Automation (ICRA)*, 1-8.
20. Morris, A., & Turner, B. (2025). A Digital Twin for a Bipedal Service Robot: Real-Time Simulation and Monitoring with MATLAB. *Advanced Engineering Informatics*, 55, 101-115.
21. Lee, K., & Park, S. (2022). A Comparative Study of ODE Solvers in MATLAB for Simulating Bipedal Robot Dynamics. *Journal of Computational and Nonlinear Dynamics*, 17(3), 031-045.
22. Mathworks® - technical documentation
23. Cantrell AJ, Imonugo O, Varacallo MA. (2023) Anatomy, Bony Pelvis and Lower Limb: Leg Bones. *StatPearls [Internet]. Treasure Island (FL)*
24. Wills, A., Schön, T. B., Ljung, L. & Ninness, B. (2013). Identification of Hammerstein–Wiener Models. *Automatica*, 49(1), 70-81.
25. A. van der Schaft and D. Jeltsema (2014). Port-Hamiltonian Systems Theory: An Introductory Overview. Foundations and Trends® *in Systems and Control*, vol. 1, no. 2-3, 173-37