Dynamic Walking Challenge: Go the Distance!

Note: This example is compatible with OpenSim version 4.1.

Overview

In this exercise you will use the OpenSim software to design and simulate a dynamic walker. As a starting point, you will be given a Passive Dynamic Walker Model and an arena with obstacles. The goal of the exercise is to maximize the distance the walker can travel on increasingly challenging terrain by adjusting the model's parameters and adding new model components. You will use the OpenSim graphical user interface (GUI) and Matlab scripting commands to add model components, adjust component properties, visualize dynamic simulations, and make plots of your simulation results.

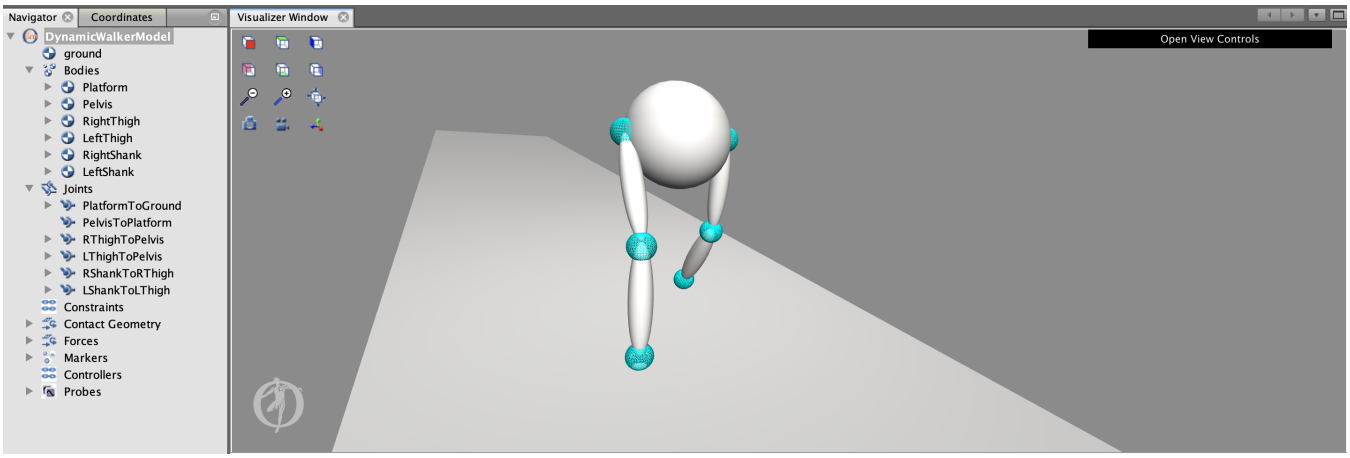


Table of Contents

I. Getting Started

Below are some useful (and necessary) resources that you should read before you begin and keep handy while you build your model;

- Read the Introduction to the OpenSim API section
- Setup your Matlab environment using instructions from the Scripting with Matlab page.
- Bookmark the Common Scripting Commands page for helpful tips and code snippets
- Read the Guide to Using Doxygen pages and bookmark the OpenSim Doxygen documentation
- Bookmark the API Guide on Doxygen.
- Setup a working folder where you will save and test your model.

All scripts and model files for this exercise are included in the OpenSim 4.1 distribution and are found in your OpenSim resources directory; ***<Resources-Dir>/Code/Matlab/Dynamic_Walker_Challenge/.***

II. Explore the OpenSim GUI and Model

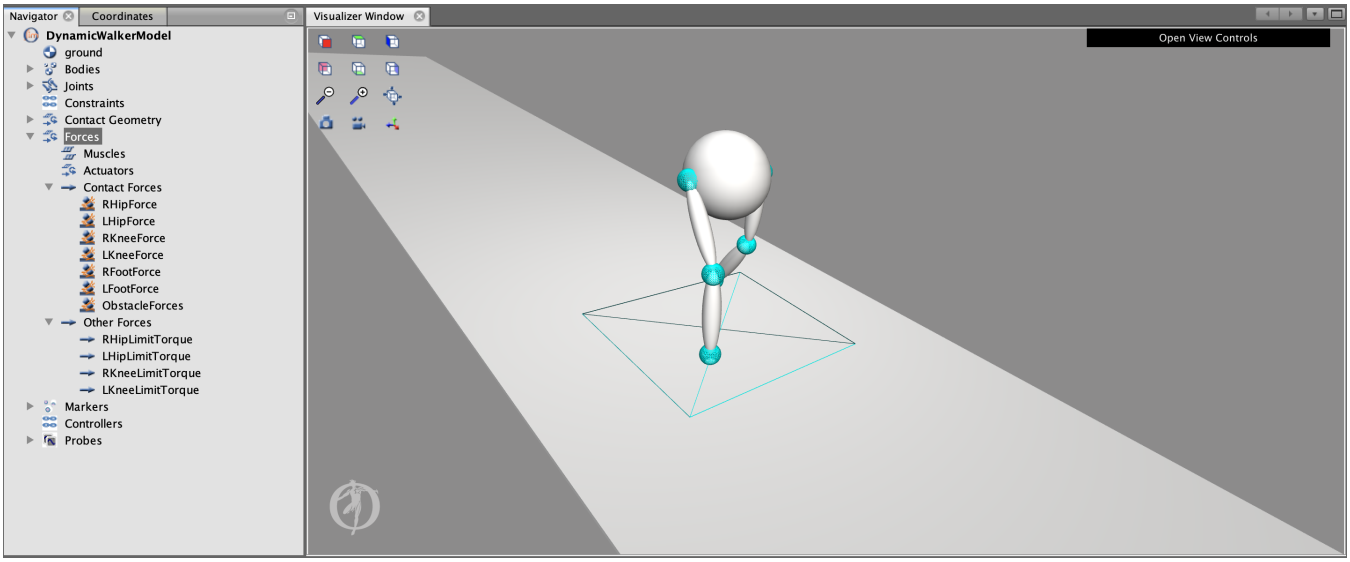
In this section, you will familiarize yourself with the OpenSim GUI and explore the editable properties of the walker and environment. You will use the GUI to examine the bodies, joints, contact geometry, and forces that make up the model. In the following sections, you will learn how to add bodies and joints and adjust the model properties to create your own robust walker.

A. Launch the OpenSim Program

Launch the OpenSim application. You will see multiple information panels and an empty main Visualizer Window. You can learn more about the Graphical User Interface (GUI) in the OpenSim User's Guide.

B. Explore the model components

- In the OpenSim GUI, select **File>Open Model...** to open the ***WalkerModelTerrain.osim.***
- The **View** panel is used to control the model's visual display. You can move the view camera around the model using a combination of mouse and trackpad gestures.
- In the **Navigator** panel on the left side of the screen, use the **+ icon** to expand the list of model components. Explore the individual components in the model by clicking on them in the Navigator Panel.
- As you click on each component, scroll through the component's properties in the **Property Editor** panel underneath the Navigator panel.
- Observe each of the following:
    - **Bodies:** A platform, pelvis, thigh, and shank bodies with mass, inertia, and visual geometry.
    - **Joints:** A Planar joint connecting the model with the Platform and Pin joints between all the segments' pin joints.
    - **Contact Geometry:** Contact spheres and a contact half space (plane), which are used to generate contact forces.
    - **Contact forces**: Forces that are computed and applied due to the interaction between contact geometries.
    - **Other Forces:** Coordinate limit forces, which enforce the desired range of motion of the joints. Springs and other forces.



- The **Coordinates** panel lists the current values of the model's generalized coordinates.
    - The values of the coordinates (in meters or degrees) can be changed with the sliders or by entering a value in the left text box.
    - The initial coordinate speeds (in meters/second or radians/second) are used as the initial conditions forward simulations. These can be set in the speed text box on the right hand side of each row.
    - The value of a coordinate can be locked by toggling the lock icon.
    - To reset the model to the default pose, select **Poses>Default**. You can also uses the Poses menu to change the default pose and add custom poses.

III. Simulate and Visualize the Walker

In this section, you will run a dynamic simulation of the model, visualize the resulting motion, plot the output, and save a movie of the motion using the OpenSim GUI.

A. Run the Forward Tool

1. Before you run the forward tool, go to the **Coordinates** tab in the left panel and choose a set of initial coordinate values and coordinate velocities for your simulation.
2. Select **Tools>Forward Dynamics** from the top menu in the GUI
3. In the Time Pane, set the **Time Range to Process** from 0 to 2 seconds
4. In the Output Pane, append a **directory** called FWD to the currently listed directory. You can leave all the other settings at their default value.
5. To quickly set up future runs of the forward tool, save your settings to a file (e.g., Setup_Forward.xml) by clicking the **Save...** button
6. Click the **Run** button to begin the simulation.
7. **Close** the tool.

B. Visualize the motion

After running the forward tool, three storage files (.sto) are written to the output directory specified with the time history of the control signals and the model states. The model state histories, which for this model contain the coordinate values and velocities, are automatically loaded into the program (**Results**) and can be accessed in the **Navigator** panel under **Motions**.

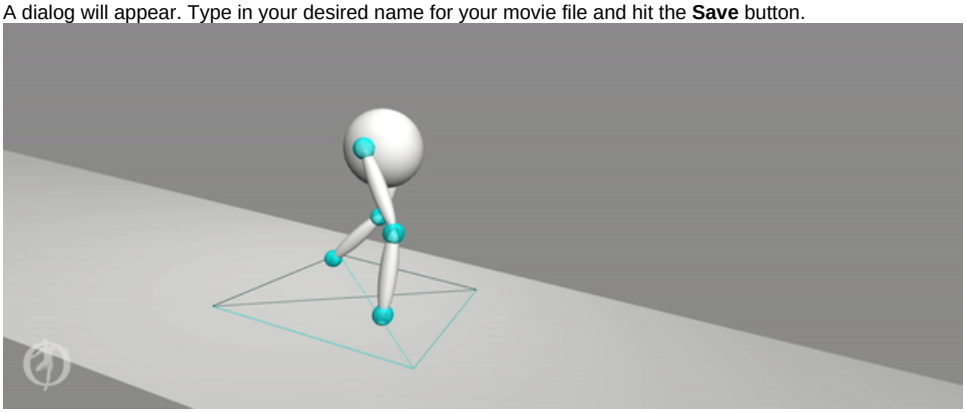
1. If you wish to run several motions with different starting conditions, make sure to **Rename** the motion from the default name **Results** by right clicking on the item in the **Navigator** (e.g. Default_Results).

At the top middle section of the GUI, the blue movie controls can be used to playback and visualize the results. On the left hand side, the playback speed can be adjusted with the arrow icons or by entering a value in the textbox. In the center, you can use the control icons to play a movie forward or backwards, either continuously, or frame-by-frame.

1. Hit the blue play button to visualize the results from your forward simulation.

C. Create a movie in the GUI

1. In the **Navigator** panel under **Motions**, double click a motion to make it the current motion (**bold name**).
2. On the left side of the Visualizer Window, click the video camera icon.
3. Set the time and speed of your motion using the view controls.
4. Play the resulting motion (Note: you can pause the movie before completion if you want a specific interval). You can pause and rotate the model and play again to get a movie with two views of the walker.
5. Click the video camera icon again. This will end the capture and turn the camera icon blue again.
6. A dialog will appear. Type in your desired name for your movie file and hit the **Save** button.



D. Plot in GUI

The GUI has a Plotter tool for plotting results. We'll plot the coordinate velocities for the right leg of the model.

1. Select **Tools>Plot** from the top menu in the GUI
2. In the bottom panel, select **Y-Quantity...** and select the **Results(deg.)** file which was created by the forward tool.
3. In the **Select Motion Quantity** window, select the generalized coordinate velocities for the right hip (RHip_rz_u) and the right knee (RKnee_rz_u) and hit OK.
4. In the bottom panel, select **X-Quantity...** and select time.
5. In the bottom right hit **Add** to plot the data.

More information regarding the plotting tool is available by hitting the help button in the GUI or by going to the help page here.

E. Iteratively modify the model to maximize walking distance

The following is a partial list of the common parameters you can change. Try a few different changes to see if you can improve the walker. But don't worry too much of your walker doesn't go very far. In the sections that follow, we will discuss how to generate a more stable walker.

1. Modify the mass and inertia parameters of the bodies.
    a. The mass and inertia properties of the body can be accessed through the Navigator panel. Open the bodies group by clicking on the **+** icon and selecting a body from the list. In the properties window you can select the mass, the mass_center location, and the six unique elements of the inertia matrix for the body.
    b. The mass_center is the location of the mass center from the body origin measured in the body reference frame and the inertia values are the inertia of the body about the body center of mass measured in the body reference frame.
2. Modify the segment length in the joints.
    a. Access the joint properties in the **Joints** set in the **Navigator** panel. The length of the segment is set by the adjacent joints. To change the length of the right thigh segment, edit the translation property of the appropriate RightThigh_offset frame (located under Bodies > RightThigh in the Navigator). Read more about OpenSim joints on the OpenSim Models page or in the OpenSim Doxygen.
3. Modify the initial starting conditions by entering the initial coordinate values and the initial coordinate speeds in the **Coordinate** panel.

Notes:

- In the above configuration of the **Forward Tool**, the initial conditions of the system will be taken from the listed values in the **Coordinates** panel.
- Changing the joint locations to length the segments will not change the visual object in the GUI. To change the visual object, open the **Body** set, click on the body, and click on **...** for the **Displayer** element in the **Properties** panel.
- Use the keyboard shortcuts (Ctrl+Z for undo, Ctrl+Y for redo) or the orange and blue arced arrows in the Top Left section of the GUI to manage your model changes.

IV. Extending the Model with the Matlab Scripting interface

The base model has trouble maintaining knee extension during stance. In this section, you will use the Matlab scripting interface to extend the base model in two ways. First, you will add a force component to the model that will always act to extend the knee. Second, you will add a more substantial foot to the model. Before getting started, make sure your working directory in Matlab is the *UserFunctions* directory

A. Add a Magnet Force around the Knee Joint in Matlab

With the base dynamic walker model, the knee often flexes during the stance phase. One way to maintain knee extension during the stance phase is to add a magnet force around the knee which will contribute to extension of the knee.

OpenSim provides a library of actuators, from simple springs up to efficient muscle models. Here, we add a magnet force with a general purpose force called an ExpressionBasedPointToPointForce. The ExpressionBasedPointToPointForce calculates the relative distance (d) between two points and its time derivative (ddot) and allows the user to specify the mathematical expression for the force using the variables **d** and **ddot**. The function for the ExpressionBasedPointToPointForce can be written as a string such as +,-,*,/ and common functions such as exp, pow, sqrt, sin, cos, and tan. Importantly, the expression may not contain whitespace.

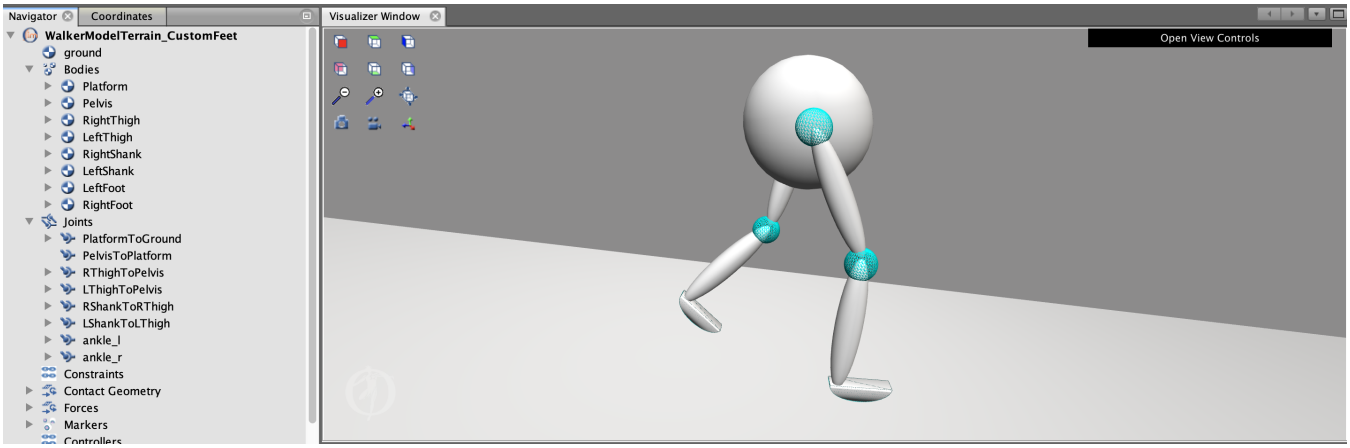
The magnet forces will be calculated as $f = \frac{c}{d^2}$ where the constant, c, is 0.01 Nm^2

AddExpressionPointToPointForceMagnets.m is a partially completed script that adds these magnet like forces to the right and left knee. The script adds a ExpressionBasedPointToPointForce for the right knee, open the script in Matlab and complete the TODO sections using the already completed code and Doxygen documentation as a guide to add an ExpressionBasedPointToPointForce to the left knee. Once you have run the completed script, open the new model in the GUI and observe the affect of the ExpressionBasedPointToPointForce on the right and left knee motion.

B. Add a custom foot mesh

Another way to add an extension moment during stance is to add a more substantial foot with a large radius of curvature. During stance, the contact forces create a moment on the lower leg which acts to extend the knee. By moving the contact point further ahead on the foot, the moment on the foot created by the ground contact forces can be used to inhibit knee flexion in stance. This section demonstrates how to add a custom foot object using the Matlab.

AddCustomFeet.m adds ContactMesh components for the feet from an included mesh file and creates an ElasticFoundationForce component to represent the contact between the foot and the ground. The mesh file (.obj) is of a simple cylindrical foot design, created in the open source program Blender.



OpenSim comes with a large library of geometry files for visualizing elements of the human skeletal system, as well as simple generic shapes such as, spheres, cylinders, and boxes. If the base library is not sufficient for your model, it is easy to add additional geometry files. OpenSim can use Geometry types with the following file extensions: .vtp, .stl and .obj. By default, OpenSim looks for Geometry files in the Model's local directory and in the OpenSim Geometry folder. You can add additional locations to look for Geometry..

V. Design and Simulate your own Walker Model

The UserFunctions directory includes example scripts to help you iteratively build your own walker model and perform forward simulations. These scripts are designed for convenience and to demonstrate key parts of the functionality of the Matlab scripting interface. You can access the documentation for each function by using Matlab's internal help interface (e.g., help RunForwardTool). You can use tab completion or methodsview (e.g. methodsview (Millard2012EquilibriumMuscle) to access available methods for OpenSim Classes.

The below scripts can be used to augment your model by adding different types of force components. Running each script will generate a new model with the corresponding component attached. Each script edits the default, unedited, model by default. You can progressively add additional components to your model by changing the input name of the model in the script. Ideally, you should mix and match any number of components to generate a model of your liking.

Script Name	Description	Related Functions/Classes
AddClutchedPathSpring.m	Adds a clutched path spring to the model. The clutch is set based on length.	ClutchedPathSpring

AddCustomFoot.m	Adds a custom mesh object to the base model.	WeldJoint, ContactMesh, ElasticFoundationForce
AddExpressionPointToPointForceMagnets.m	Adds a magnet force between the thigh and shank to add a knee extension torque.	ExpressionBasedPointToPointForce
AddPathSpring.m	Adds a path spring to the model.	PathSpring
AddSpringGeneralizedForce.m	Adds a spring generalized force to the model.	SpringGeneralizedForce
CreateWalkingModelAndEnvironment.m	Takes a basic walking model and adds obstacles.	ContactSphere, ContactHalfSpace, HuntCrossleyForce

As part of the design process, you will want to iteratively perform forward simulations with your new model to see how additional components, as well altering initial model coordinate values and speeds, improves the perform of the walking model. Use the **DesignMainStarter.m** script to quickly and iteratively perform multiple simulations of your model. This script allows you to change the initial coordinate values and speeds of your model, decide if you want to view the simulation using the SimTK visualizer, and plot results from the simulation. The SimTK visualizer is a handy tool for being able to observe simulation results without having to leave your development environment (Matlab or Python).

Script Name	Description	Notes
DesignMainStarter.m	The starting point for design iteration. The script loads a model, allows you to change the initial coordinate values and speeds, and perform a forward dynamic simulation using Matlab.	Change the path to appropriate model you wish to simulate. Initially, the model file is set to the default *osimModel = Model('../Model/WalkerModelTerrain.osim')*. To use the SimTK Visualizer to view the simulation, set *visualize = true*. Once you have the model walking, change endTime to set the simulation time length. The coordinate values and speeds are all initially set to the default value of the model, change these values to alter the initial pose of the model. Results of the simulation are written to **ResultsFWD/simulation_states.sto**
PlotOpenSimData.m	Generates plots from the simulations results of **DesignMainStarter**. Reads state values from **simulation_states.sto**, and plots some variables of interest.	Plots are only performed for Pelvis X-translation, right Hip rotation, and right knee rotation. Edit the file to plot other states of interest. You can view what is plotable by opening the contents **ResultsFWD/simulation_states.sto** in a text editor (or Excel).

Acknowledgments

The original exercise was created by Daniel A. Jacobs. Ajay Seth, Chris Dembia, Jen Hicks, James Dunne, Tom Uchida contributed to the scripts library used in this example.

References:

1. Millard, M., Uchida, T., Seth, A., Delp, S.L. (2013) Flexing computational muscle: modeling and simulation of musculotendon dynamics. ASME Journal of Biomechanical Engineering, 135(2):021005.