Univ Lyon, Université Claude Bernard Lyon 1, Univ Gustave Eiffel, IFSTTAR, LBMC UMR_T9406

EquiSim
User's Manual

Toufik BENTALEB

March 15, 2021

Abstract

This manual is intended to help users to run and to know how and where they can perform any modification in the EquiSim software tool. This software tool was developed to allow human balance prevention and recovery to be performed using MATLAB and using linear model predictive control (LMPC) algorithm. A "Quick Start" approach to using this software will be presented, along with a detailed section containing full explanations and examples for using this tool.

Contents

1 Introduction

The aim of this document is to help users to use the MATLAB software tool for human/robot balance prevention and recovery. It is intended to provide all necessary information in a simple way. A "Quick Start" approach to using this tool will be presented, along with a detailed. The EquiSim user's manual contains a thorough description of all files in the software tool.

2 System Requirements

In order to get the code to execute properly, the following programs are required:

1. The program has been developed for use in the MATLAB environment and it is running perfectly on `MATLAB 2017b` (`MATLAB 9.03.0`).

2. To solve the quadratic programming problem using `quadprog` function the `OPTIMIZATION TOOLBOX` is required.

3 How files are organised

The folder `EquiSim_folder` contains all files necessary of the software tool, were divided into many categories:

- The main Matlab script file `main_EquiSim.m` to run the program.

- The folder `core_test` contains `classdef`, `function`, and `script` folders because the software was built using Object-Oriented (OO) techniques.

- The folder `core_MPC` contains `classdef`, `function`, and `script` folders.

- The folder `core_physical_model` contains `classdef`, `function`, and `script` folders.

- The folder `results` contains results graphs, for the test performed, in format of `*.JPG`, `*.EPS` and `*.PDF`.

4 Control scheme

The feedback loop used to simulate the balance recovery is shown in the Fig. 1. When you run the main function `main_EquiSim`, the software starts to add all subfolders to matlab path and prepare storage files after that, it runs `classdef_create_experiment` which is shown in the scheme below by ***`Desired final states`***. After creating the human model (using ***`script_constant.m`***), and the experiment steps (using `classdef_create_experiment`), the inputs of a MPC iteration are created. The ***`Mechanical Model`*** is the physical model of the human, in the code we use `classdef_physical_model` to run it. To add the disturbance to the physical model just uncomment `physical_model_storage.add_storage_sensors` and comment `physical_model_storage.add_storage`. The same for the delay in the model (the ***`Sensors`*** in the scheme) you need just give the neural time delay (`neural_time_delay`) and uncomment `sensor_dynamics.sensor_dynamics_iteration`.

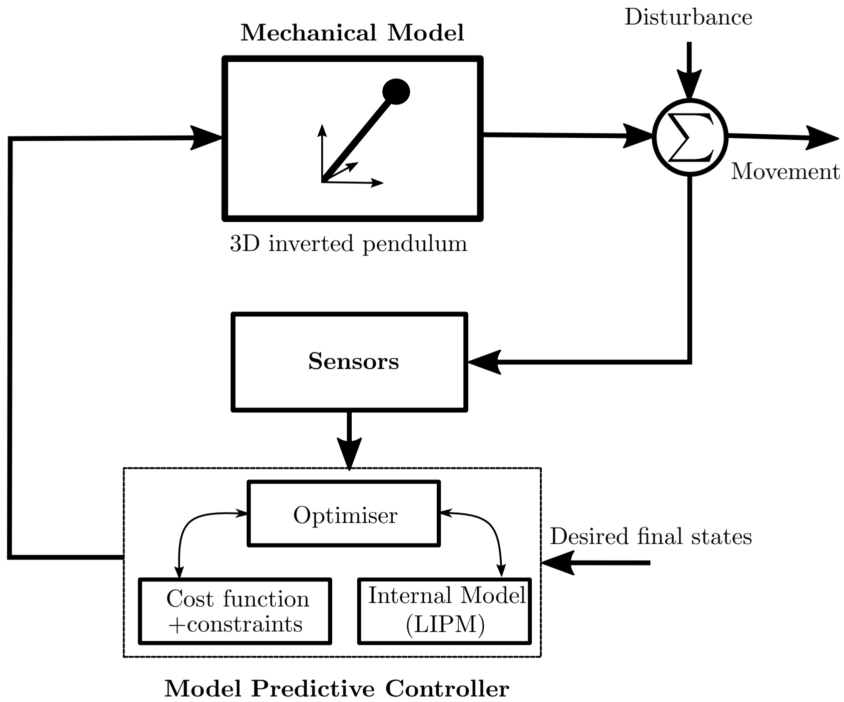

Figure 1: Flow diagram of the human balance recovery

5 Getting started with software

This section will take you through one example in order to get you started with the MATLAB software tool for human/robot balance prevention and recovery.

5.1 The main function

The m-script `main_EquiSim.m` is the only file visible in the folder `EquiSim_folder` to run the software, and you can call it from the command line. Shown below (Fig. 2) is a part of the the main function and their parameters are described in the Table 1 and Table 2.

```
clear all, close all, clc,
path(pathdef); %clear the pathdef of include library

%addpath script/
addpath core_physical_model/function/ core_physical_model/classdef/ ...
    core_physical_model/script/
addpath core_test/function/ core_test/classdef/
addpath core_MPC/classdef/ core_MPC/classdef/linear_trajectories/ ...
    core_MPC/function/ core_MPC/script/

robot_type='human';
phase_duration_type='phase_duration_01';
walking_type=4;
cop_ref_type='ankle_center';
polyhedron_position='waist_center';
kinematic_limit='hexagonTranslation';
COM_form='comPolynomial';

run('core_test_all_axis/script/script_constant.m')
run('core_test_all_axis/script/script_init_storage_qp_result.m')
run('core_physical_model/script/script_init_storage_physical_model.m')
run('core_physical_model/script/script_init_storage_sensor_dynamics.m')
.
.
.
```

Figure 2: File saved as main_EquiSim.m

File/Variable/String	Description
`robot_type`	Name of a m-file script that contains the physical properties of a human or robot bodies: `robot_type="human"` → human model. `robot_type="hrp4"` → hrp4 robot model. `robot_type="hrp2"` → hrp2 robot model. `robot_type="human"` was chosen as a case study. The `"human"` string is used to call the m-file script `human.m`. The default Center of Mass (CoM) height `h_com`. COM height limits to the floor with respect to h^{com} are h^{com}_{max} and h^{com}_{min}. Initial standing state default with release angle θ, $\omega_0 = \sqrt{(g/(h^{com} \cdot \cos(\theta)))}$ and $\zeta_0 = 1/\omega_0^2$ where g is the gravity. The parameters in the file are: $x^{com}_0 = [h^{com} \cdot \sin(\theta); 0; ((h^{com} \cdot \sin(\theta))/\zeta_0)]$; $y^{com}_0 = [0; 0; 0]$; $z^{com}_0 = [h^{com} \cdot \cos(\theta); 0; 0]$; The feet initial positions: $x^{step}_{r,0}$, $y^{step}_{r,0}$,$x^{step}_{l,0}$, and $y^{step}_{l,0}$.
`phase_duration_type`	Name of a script file that contains the duration and sampling time of the phases: `phase_duration_r`→ Swing phase of the right foot. `phase_duration_l`→ Swing phase of the left foot. `phase_duration_b`→ Double support phase. `phase_duration_RT`→ Reaction time. `phase_duration_APA`→ Anticipatory postural adjustments (APA) phase. `phase_duration_start`→Starting phase. `phase_duration_stop`→ Stop phase. `N_r`→ Right foot swing phase sampling time. `N_l`→ Left foot swing phase sampling time. `N_b`→ Double support phase sampling time. `N_RT`→ sampling time of the reaction time. `N_APA`→ Anticipatory postural adjustments (APA) phase sampling time.

Table 1: The constant parameters of the simulation case (part 1)

File/Variable/String	Description
walking_type	to select which type of walking area walking_type→ walking flat. walking_type→ walking airbus stairs. walking_type→ walking flat quick. walking_type→ walking walking flat fixed foot step positions. walking_type→ walking airbus stairs fixed foot step positions.
cop_ref_type	translate the step position to the cop_ref_type→ 'ankle_center' Center of pressure reference centered on the ankle. cop_ref_type→ 'foot_center' Center of pressure reference centered on the middle of the foot.
polyhedron_position	Polyhedron centered: polyhedron_position→ 'ankle_center' on the ankle. polyhedron_position→ 'foot_center' hexagon kinematic limits. polyhedron_position→ 'hexagonTranslation' hexagon kinematic limits with translation.
kinematic_limit	The kinematic limit: kinematic_limit→ '' very simple polyhedron. kinematic_limit→ 'hexagon' on the middle of the foot. kinematic_limit→ 'waist_center' on the middle of the waist.
COM_form	COM trajectory form: COM_form→ 'comPolynomial' COM with piece-wise jerk. COM_form→ 'comExponential' ZMP with piece-wise velocity. COM_form→ 'comPolyExpo' COM with polynomial of exponential.
firstSS	First-foot stepping: firstSS→ 'r' Right foot. firstSS→ 'l' Left foot. firstSS→ 'b' Both feet.
nb_foot_step	Number of steps.

Table 2: The constant parameters of the simulation case (part 2)

6 Script files

In the beginning, the main function `main_EquiSim` calls many other scripts to load human model parameters, creating the experiment test, preparation of storage data from tree type of physical models such as the model used in MPC controller, the physical model without noise, and the physical model with noise.

6.1 `script_constant`

First, this script m-file calls the object-oriented script `classdef_create_robot` to define the human model parameters (see section 7.1), then it calls the object-oriented script `classdef_create_experiment` to create the experiment (case study) (see section 7.2).

6.2 `script_init_storage_qp_result`

This script m-file calls the script `script_init_storage_qp_result` to initialize the MPC storage of MPC iteration from QP result. The storage result contains the COM position, velocity, and acceleration along x-y-z axis and feet positions.

6.3 `script_init_storage_physical_model`

This script m-file is used for initial storage of the physical model states. This `script_init_storage_physical_model.m` initialize object of the class `classdef_physical_model` (see Section 7.3).

6.4 `script_init_storage_sensor_dynamics`

Here, the `script_init_storage_sensor_dynamics.m` file stores the model states recorded from the sensors, i.e. after adding the delay on the states of the physical model.

7 Classes files

The Matlab class is used to define an object that encapsulates data and the operations performed on that data.

7.1 classdef_create_robot

The class `classdef_create_robot` is created to define the properties of the parameters of the human model after choosing the `robot_type` (see Table 1). The script file `"robot_type".m` initiates the human model parameters. The properties that contain the numeric data stored in this object of the class are shown in the table 3.

Parameters	Description
`h_com`	CoM height
`h_com_max`	COM height maximum limit to the floor with respect to the COM height
`h_com_min`	COM height minimum limit to the floor with respect to the COM height
`xcom_0`	Initial standing state default X-CoM
`ycom_0`	Initial standing state default Y-CoM
`zcom_0`	Initial standing state default Z-CoM
`xstep_r_0`	Initial right foot position on the x-axis
`ystep_r_0`	Initial right foot position on the y-axis
`xstep_l_0`	Initial left foot position on the x-axis
`ystep_l_0`	Initial left foot position on the y-axis
`backtoankle`	from back to ankle of foot
`fronttoankle`	from front to ankle of foot
`exttoankle`	from exterior to ankle of foot
`inttoankle`	from interior to ankle of foot
`sole_margin`	from floor to ankle of foot
`xankmax`	stepping forward max
`xankmin`	stepping forward min (if negative, it means stepping backward max)
`yankmin`	width min between ankles
`yankmax`	width max between ankles.

Table 3: The constant parameters `classdef_create_robot`

7.2 classdef_create_experiment

The class `classdef_create_experiment` is created to define the properties of the experiment "case study". The script file `"robot_type".m` initiates the human model parameters. The properties that contain the numeric data stored in this object of the class are shown in the table 4 table 5.

Parameters	**Description**
g	Gravity
omega_temp	$\omega_0 = \sqrt{(g/h^{com})}$
zeta_temp	$\zeta_0 = 1/\omega_0^2$
phase_duration_r	Right foot phase duration
phase_duration_l	Left foot phase duration
phase_duration_b	Double support phase duration
phase_duration_RT	Reaction time phase duration
phase_duration_SPT	APA phase duration
phase_duration_start	Start phase duration
phase_duration_stop	Stop phase duration
N_..	Sampling time N_r N_l N_b N_RT N_SPT N_start N_stop
preview_windows_duration	Prediction horizon
phase_duration	Phase duration definition
phase_duration_iteration	
phase_duration_cumul	
phase_duration_iteration_cumul	
T_..	T_..=phase_duration_../N_..
T_r	
T_l	
T_b	
T_RT	
T_SPT	
T_start	
T_stop	
phase_type	
phase_type_sampling	
phase_duration_sampling	
phase_duration_sampling_cumul	
phase_sampling_length	

Table 4: The constant parameters classdef_create_experiment (part 1)

Parameters	**Description**
`phase_type_decouple`	phase decoupling
`MoS_sampling`	Margin of stability (MoS)
`yaw`	Foot orientation array
`yaw_sampling`	Foot orientation sampling time
`Px_step_ref`	
`plan_hexagon`	Polyhedron from hexagone
`z_leg_min`	
`z_decalage_tot`	
`translate_step_polyhedron_type`	
`OptimCostWeight`	MPC weights
`step_number_pankle_fixed`	Fixed step position after initial step state position
`vcom_ref`	COM reference velocity
`vcom_change`	Change of reference velocity
`vcom_1`	First part of reference velocity
`vcom_2`	Second part of reference velocity
`zfloor_ref`	Horizontal position of the floor
`hcom_ref`	COM reference position
`hcom_ref_max`	
`zeta_up_ref`	
`zeta_down_ref`	
`zstep_l_0`	Left foot step height at t=0
`zstep_r_0`	Right foot step height at t=0
`zstep_l_ref`	Left foot step height reference
`zstep_r_ref`	Right foot step height reference

Table 5: The constant parameters `classdef_create_experiment` (part 2)

7.3 `classdef_physical_model`

The properties of the `classdef_physical_model` and their detailed explanation, see the table below (Table 6). This class has different functions such as

- `add_storage_sensors` function to add noise to the physical model. The noise types added to the model are
  - `rand` is an uniformly distributed pseudorandom numbers
  - `awgn` is a white Gaussian noise to a signal
- `physical_model_iteration_..` functions for different physical models

Properties	Description
`xc`	COM end position of the next iteration along x-axis
`xdc`	COM end velocity of the next iteration along x-axis
`xddc`	COM end acceleration of the next iteration along x-axis
`yc`	COM end position of the next iteration along y-axis
`ydc`	COM end velocity of the next iteration along y-axis
`yddc`	COM end acceleration of the next iteration along y-axis
`zc`	COM end position of the next iteration along z-axis
`zdc`	COM end velocity of the next iteration along z-axis
`zddc`	COM end acceleration of the next iteration along z-axis
`xstep`	foot step position along x-axis
`ystep`	foot step position along y-axis
`zstep`	foot step position along z-axis (not an optimization variable)
`xzmp`	zmp position along x-axis (not an optimization variable)
`yzmp`	zmp position along y-axis (not an optimization variable)
`zzmp`	zmp position along z-axis (not an optimization variable)

Table 6: The properties of the `classdef_physical_model` and their detailed explanation

7.4 classdef_MPC_problem_inputs

The class `classdef_MPC_problem_inputs` is used to define the inputs of MPC controller. The properties of this class and their detailed explanation, see the table below (Table 7).

Properties	**Description**
`g`	Gravity acceleration constant
`omega_temp`	Temporary value of ω
`N`	Number of sample of the preview window
`phase_duration_sampling`	Duration of each sample of the preview window
`phase_type_sampling`	Phase type of each sample of the preview window
`MoS_sampling`	Margin of stability
`zeta_temp`	Value of ζ during each sample of the preview window
`zeta_up`	Value of ζ superior bound during each sample of the preview window
`zeta_down`	Value of ζ inferior bound during each sample of the preview window
`c_init`	Matrix of CoM initial state with row [c;dc;ddc] and with column along axis [x y z]
`dc_ref`	Reference value of CoM velocity during each sample of the preview window with column along axis [x y]
`Px_step`	Px matrix of support foot for the preview window
`yaw`	Feet orientation
`no_double_support`	Matrix of no double support
`no_double_support_capture`	Matrix of no captured double support
`double_support`	Matrix of double support

Table 7: The properties of the `classdef_MPC_problem_inputs` and their detailed explanation