Simni Short Manual

Stefan Bethge
Neurorobotics Research Laboratory, Humbold-Universität zu Berlin

6th April 2014

Contents

1 Introduction

The Simni simulator is a testbed for simple robots that work in a two-dimensional world (restricted). The built-in morphology is imitating the Semni morphology of the Neurorobotics Research Laboratory, Berlin, (see `http://neurorobotics.de`). On top of the physics simulation, it contains an implementation of the ABC learning framework to explore the morphology's own stable and instable fixpoints and hence find trajectories between energy efficient postures.

It provides phase space and motor voltage plots and a graph representation of the interesting postures and connection between them in the explored manifold.

2 Requirements

- A browser, preferrably with a fast JavaScript implementation like V8. Recommended at the time of writing is Google Chrome or Chromium version 30 or newer.
- For compiling the CoffeeScript sources, a recent CoffeeScript compiler. Version 1.7.1 was used during development.

3 Usage

To open up the simulator locally, run

```
python serve.py
```

to start a simple local webserver and open `http://127.0.0.1:8000/simulator.html` in your browser. You should get a window with a Semni outline in the top left and some controls and boxes underneath it. Using your mouse, you can interact with the simulated bodies. Alternatively, the simulator will be available as long as possible under `http://www2.informatik.hu-berlin.de/~bethge/simni/simulator.html`.

3.1 Data plotters

The data plotters are in the top right of the page. To see current data and a history of the last 1000 time steps (96 Hz control loop), click the enable boxes in the top left of each graph. Phase trajectory displays in red, with the hip angle on the x-axis and the knee angle on the y-axis (it might be hard to find if there is no movement). The motor torque is displayed in green for the hip and blue for the knee torque.

3.2 Buttons, Toggles and Shortcuts

There are a number of controls in the interface that are grouped by usage. The "Simulation Controls" allow the selection of morphologies, at the moment single and double pendulum and Semni. "Simulate in Realtime" is intended to make the physics move the robot at about 60 fps which corresponds to about the same speed as realtime. Otherwise the simulation will run as fast as possible. "Pause/Run Simulation" toggles between halting and running again. The graph layouting and other processes independent from the physics will not be paused. While paused, progressing one step can be achieved by pressing "Next step(s)". This means however 10 physics simulation steps and one controller step, c.f. Section 5. The buttons "toggle CSL" and "toggle bounce controller" each enable or disable motor controllers for both

joints. The CSL controller will use the paramters below the buttons, the bounce controller (using constant velocity and changing direction on stall) has default parameters.

The "ABC learning" group contains controls related to the learning framework. The button "toggle explore" will also toggle the CSL controller and will start exploring. It will use the "heurisitc" setting to determine the next csl mode when a fixpoint was detected. This change will be reflected in the select boxes and the CSL parameters on the left. The buttons "Save graph as SVG", "Save graph as JSON" and "Load graph from JSON" should be self explanatory. "Load graph from Semni" loads a list of nodes as they are printed over a serial console from the associated ABC implementation on a hardware Semni robot. All progress will be displayed in the graph below all of the controls after postures are detected. The group "Graph properties" holds settings for the visual graph representation. Repulsion and stiffness relate to the spring simulation between nodes and changes the way the layouting looks and behaves. These will need a browser with support for html range elements. "Animate graph" will toggle whether the graph is continuously redrawn or only once when a new posture is detected or when the user moves or hovers nodes in the graph with the mouse. "Pause graph layouting between new poses" will make the layout algorithm be stopped after a few seconds of processing whenever a posture was found. Otherwise it will run continuously and might slow down the physics simulation. If the page was loaded from a remote url, web workers will be used for the layouting which runs the layouting on a different CPU core than the physics (if possible) and will make this setting unnecessary. "Show node activation" will show the numerical value that indicates how much there is to explore or to still be learned at each node. This is used with the unseen mode strategy to determine which edge to follow when for one node, all possible directions have been taken already. The same information is shown by colour if "Show node activation colors" is activated. "Show Semni postures" will trigger the display of small semni images above each node in the graph. "Show transition labels" will show the CSL mode that was set during a transition from one node to another, in addition to the mode on the node, as these may differ. Finally, "Save graph with every new posture" will save the whole graph as JSON file whenever a new pose is found to have an automatic history of what happened. Further details of the learning process can be found in the web browser's console.

The more frequent buttons have one letter shortcuts as depicted by the [ ] brackets in their button label, so pressing this letter key will call that action.

Next to the simulation visualization, there are options to zoom the view, show current angles and center of mass (COM) and drop in test objects. Dragging the white space of the simulation view with the mouse pans the view. It is also possible to save and load the current position and angles of the semni robot.

4 Morphologies

4.1 Included

- Single Pendulum
- Double Pendulum
- Semni

4.2 Custom

It is possible to add further morphologies by producing a file that holds JS arrays with the point data of the new entity. The points have to be in counter clockwise order (CCW) and have to produce no concave polygons (otherwise they can be separated into convex polygons with the b2Separator class). The separate parts have to be added and connected with joints in-code as is done for the existing ones in the file physics.coffee. Confer the Box2d documentation for the possible joints and settings.

5 Simulator Interna

The simulation uses a version of Box2D (2.1a3) that was ported to JavaScipt. I added a patch so a torque can be applied directly to a joint. The morphology is drawn on an html5 canvas, the graph is drawn directly with svg and the manifold data is drawn with webgl.

Apart from the basic physics simulation, there is a simple motor model and a simple fluid friction model for the joint servos.

5.1 Friction

In addition to the movements of the objects themselves, frition forces occur between two bodies that are in contact. These are directed against the forces that produce the motion and can be considered threefold. If the two bodies remain completely still, *static friction* occurs, which is dependent on the materials and the pressure and prevents movement up to a maximum force F_H^{Max}. So it can be though ofa force F_H equal to the moving force F being applied in opposite direction. Let μ_H be the friction coefficient of static friction and F_N be the force perpendicular to the surface. Then for $v = 0$:

$$F_H = -F \tag{1}$$

$$F_H \leq \vec{F}_H^{Max} = \mu_H \cdot F_N \tag{2}$$

If the applied force is greater than F_H^{Max}, movement begins along the surfaces and the bodies are no longer subject to static friction but to *gliding friction*. This in turn is divided into the *dry friction*, a constant frictional force, and *viscous friction*, which depends on the velocity of the movement. The dry friction force is also called *Coulomb friction*. and is only dependent on the normal force F_N and a coefficient μ_G for the specific material combination that is involved.

$$F_{G_T} = \mu_G \cdot F_N \tag{3}$$

The viscous friction force, however, is additionally dependent on velocity v or the angular velocity ω respectively and a separate coefficient μ_V.

$$F_{G_V} = \mu_V \cdot F_N \cdot v \tag{4}$$

6 ABC learning

The simulator implements a variant of the ABC learning framework that aims at finding semantically important points of the dynamical systems state space. For that goal, the manifold is explored using the CSL controller and that way probes the actual events for their properties.

6.1 CSL modes

Instead of just using the parameters directly, it makes sense to name certain paramter sets as modes which exhibit one of the behavioural modes the CSL can exhibit. The ABC learning henceforth only changes between the modes *release*, *contraction* and *stall*.

6.2 Posture Graph

During the exploration, fixpoints and attractors are detected and saved as a node in the posture graph. Each box displays a small representation of the configuration of the robot and some values are listed in the box. These are the id number of the posture, possibly the node activation and the modes for the hip joint and knee joint in that order.

6.3 Heuristics

There are different ways of deciding which direction to take next (random, unvisited edges), and later on which edge to chose again if all possible directions have been probed. The "unseen" heuristic has some options to prefer changing or keeping the current joint and direction, comparing of which effects was one of the reasons the simulator was made.

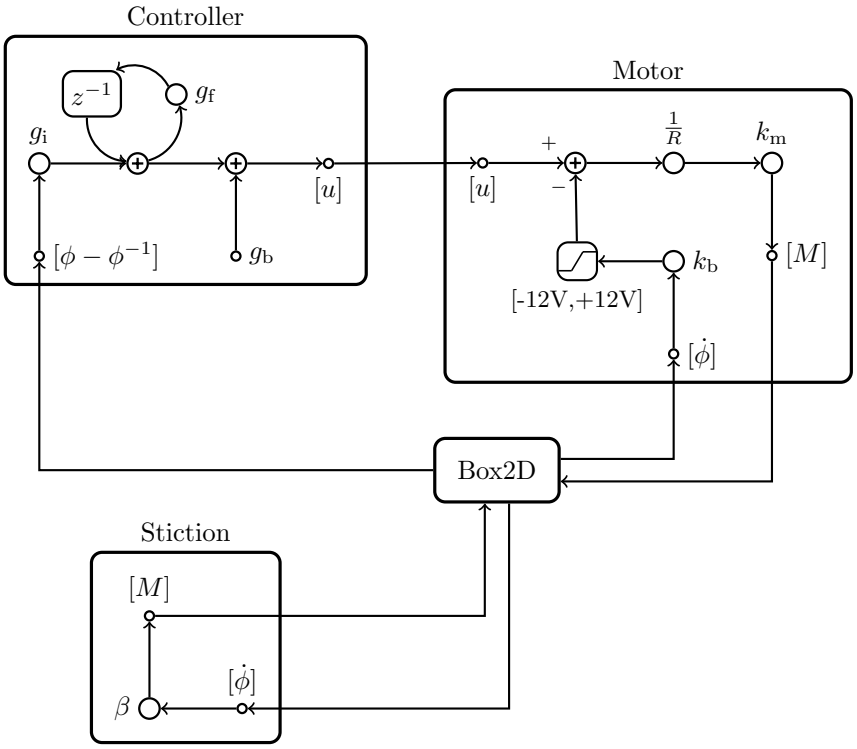

Figure 1: The components of the simulation CSL controller, motor model and friction model and Box2D. Empty circles denote a multiplication, circles with plus sign an addition. Plus or minus on an arrow give the sign for an addition. Values in square brackets give the symbol for an dimension. In this way, u denotes a voltage, M a Torque and $\dot{\phi}$ an angular velocity.