SpinalDynamics

Study the organization of spinal cord neural networks responsible for locomotion

Florin Dzeladini

13 january 2015

Introduction

Goal

Using modelling tool to study the organization of spinal networks responsible for locomotion.

Target application

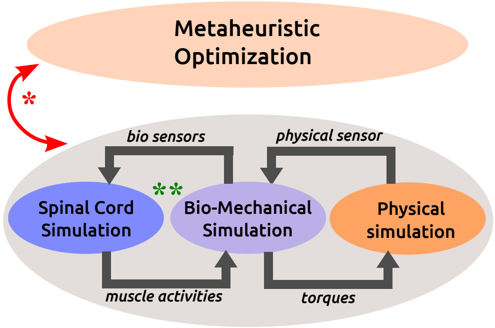
- Improve rehabilitation procedure
- Proof of concept for bio-inspired active prosthesis / orthosis controller

Scientific questions

Possible questions

1. Advantages of bio-inspired controller for orthoses and exoskeleton control
2. Spinal cord modular organization :
   - what are the roles of Central pattern generators and Motor primitives ? [Ivanenko 2014]
   - How does gait transition occur ?
3. Spinal cord input :
   - What are the role of afferent fibers in the generation of walking ? [Geyer&Herr 2010, Ting&al. 2009]
   - Descending fibers : How descending signals are incorporated into the spinal networks to modulate walking (change in speed/slope) [Ijspeert 2008]

Analysis and problem formulation: final goal



Todo* : API Extension to support :
- openSim
- Sim-mechanics

Todo** : Extend the controller structure to facilitate the link with existing bio-mechanical simulator

The library will be used on different simulators

Target physics simulators

1. Webots
2. Simbody - Gazebo
3. SimMechanics

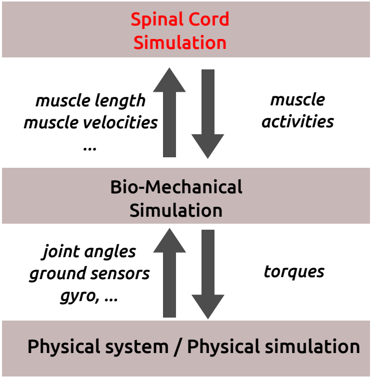
Bio-mechanical simulators

1. Libnmm (for Webots)
2. OpenSim (for Simbody)
3. *to be implemented*

Analysis and problem formulation

We separate the problem into :

- **Neural network : spinal cord simulation**
- Musculoskeletalsystem : bio-mechanical simulation
- Physics : physical simulation or physical system



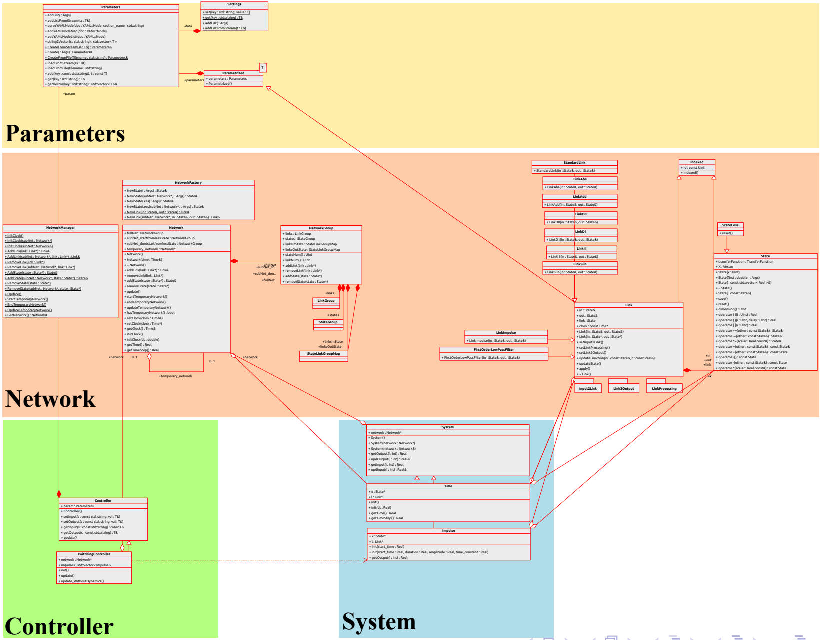
Conception : Spinal cord simulation library

General principle

The simulation of the spinal cord is abstracted as the simulation of a multidimensional dynamical stystem.
The basic components of the library are :

- **Parameters** : class used as a generic multi-purpose container
- **Network** : made of links and states
- **System** : abstract a functional part of a network
- **Controller** : abstract a controller as a set of:
  - inputs
  - outputs
  - internal variables

Conception : Class diagram



Conception : Spinal cord simulation library

General principle : Network

A network is a multidimensional dynamical system

- A network is made of links and states
  - Links describe how states are updated when the network is updated
  - States store the different states of the network
- Any network has a time object : the clock of the network
  - All links have access to the time object
- All states have a memory

Conception : Spinal cord simulation library

General principle : System

A **System** abstracts the encapsulation of a functional part of a network. It is made of :

- States and Links
- a getOutput and a getInput method

Example : the **Time** system.

The **Time** is a **System** used as the basic clock of any network.
It's a one state one link system.
The update is defined as : $x(t+1) = x(t) + dt$

Conception : Spinal cord simulation library

General principle : Controller

A **Controller** is a class with one update method and 3 **Parameters** containers storing

- inputs : const pointer of type T
- outputs : pointer of type T
- internal states : any type T

A typical implementation of a controller will have a **Network** pointer and a container to store **Systems**.

Implementation

General information

- Programming language : c++
- Hosted : bitbuckets (will most probably move to GitHub)
- Documenation : Doxygen
- Coding style : Google C++ Style Guide

Implementation

Coding principle : Test Driven Development

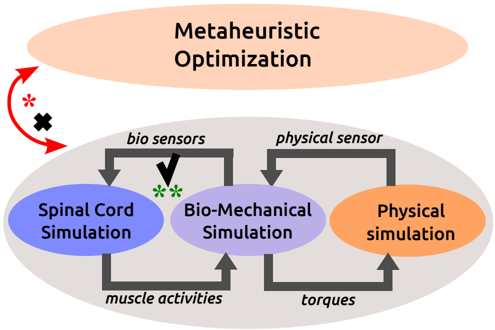
- First write how you will use your program (i.e. write tests)
- Then code to make your program pass the tests

Unit test library : Catch

Advantages

- When writing complex library it is easy to forget that some features have been implemented.
  With TDD we already have a **How-to** *: the* **Unit tests**
- When trying to implement deep changes in a library, it is hard/impossible to know all the different part of the software that will be affected.
  With TDD we know: all tests that don't pass after the changes reflect the affected parts

Achievement



Todo* : API Extension to support :
- openSim
- Sim-mechanics

Todo** : Extend the controller structure to facilitate the link with existing bio-mechanical simulator

The library will be used on different simulators

Target physics simulators

1. Webots : **OK**
2. Simbody : **OK**
3. SimMechanics

Bio-mechanical simulators

1. Libnmm (for Webots) : **OK**
2. OpenSim (for Simbody) : **OK**
3. *to be implemented*

Next steps

Implement bio-inspired walking controller

- *Implement an algorithm for self organization of spinal reflexive pathways [Marques, H.G. 2013]*
- *Implement a reflexive walking controller [Geyer&Herr, 2010]*
- *Implement a central pattern generator based controller [Taga, 1996]*

Twitching controller

Twitch definition : spontaneous motor activity (SMA)

Controller design process

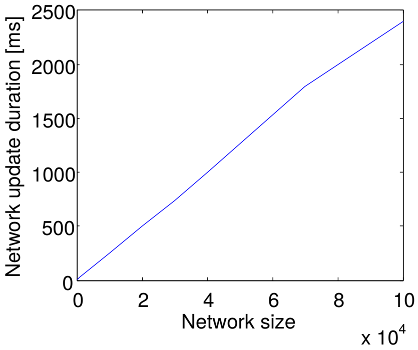
- Design the network architecture : (create Systems and Links child (if needed)
- Implement the controller : choose input / output and parameters

Applied to the twitching controller

- **LinkImpulse** class : child of **Link**
  first order differential equation generating an impulse at a given time
- **Impulse** class : child of **System**
  encapsulation of a LinkImpulse and a State. The output is an impulse.
- **TwitchingController** class : child of **Controller**
  implementation of a network of impulses.

Twitching controller: performance

Network creation time



Network update time