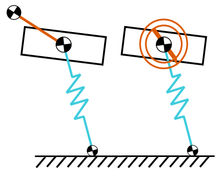
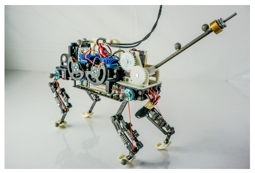
Why Use Robots to Understand Nature?

Accompanying Handout to the Poster Presentation
Simplifying Control through Active Tail Use, SICB 2015


(a) Going from models...


(b) ...to robots...

(c) ...to animals!

We present here an overview of the methods, algorithms and math used to develop and validate our model. We hope this can serve as a reference for those interested in taking a similar approach in their own projects, as well as convince you that bio-robotics can be very useful to understanding nature[4]! We do *not* focus on our results, which are only briefly presented below. Instead, we refer to the poster itself, which will be available for download from `http://biorob.epfl.ch/students/heim` along with other relevant material.

Summary of Results

We explored the use of an active tail during steady-state legged locomotion in the sagittal plane, focusing on two different control objectives: energy-input and body-pitch stabilization. Our findings indicate that by a proper choice of tail morphology, i.e. a long and light tail (which approximates the dynamics of a flywheel), the resulting dynamics allow for separating or *decoupling* the energy-input and body-pitch stabilization control objectives. In other words, the control problem is greatly simplified, which generally results in both higher performance as well as better versatility. We validate these theoretical findings both in **hardware** as well as in **simulation**, making extensive use of optimization algorithms.

Why Rebuild the Solution?

While engineers are typically preoccupied with *how* to build something to do something specific, biologists are usually dealing with the inverse problem: understanding *why* nature converged in a specific design of morphology, behavior etc.. We believe that these two questions are fundamentally linked, i.e. being able to answer one implies being able to answer both questions!
The advantages (from our point of view) of experimenting by rebuilding compared to experimenting with animals are the following:

- Specific parts of the problem can be isolated.
- Parameters can be rather freely changed: *Nature as it could be!*
- Easier/Safer to make your robot do what you want.

Robots in Hardware

Building experiments in hardware can be challenging as it often involves several fields, from mechanics to electronics, computer science and control theory. Despite this, it is also one of the most rewarding approaches! For our experiments, we used the Cheetah-Cub[6], a small cat-like robot developed at BioRob, and added a tail-module. We used a very simple tail-design, with only a single joint actuated by a servo-motor. The tail itself is made out of a detachable carbon-fiber rod with an adjustably weighted tip. This design modular is well approximated as a massless-rod and point-mass, making it very quick and easy to adjust and test different parameters. High-speed (120 fps) film was used as a quick way to develop a good intuition of how different parameters affected the robot's gait. For obtaining qualitative data, we used a *motion capture* system (OptiTrack) record position and orientation, which was then processed using **bt-k and mokka**[1] and analyzed in MATLAB. Below we list the advantages of hardware compared to computer simulations.

- Richer dynamics "for free" (certain dynamics are extremely difficult to model accurately in simulation).
- You can experiment designs with dynamics you don't know yet.
- In some cases, it can be faster to build and run robots than to build a sophisticated simulation.
- Generally provides more intuition for the problem than a simulation.
- Never have numerical problems.

References

[1] A. Barre and S. Armand. "Biomechanical ToolKit: Open-source framework to visualize and process biomechanical data". In: *Computer methods and programs in biomedicine* 114.1 (2014), pp. 80–87.

[2] R. C. Eberhart and Y. Shi. "Particle swarm optimization: developments, applications and resources". In: *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. Vol. 1. IEEE. 2001, pp. 81–86.

[3] B. C. Fabien. *Analytical system dynamics*. Springer, 2008.

[4] A. J. Ijspeert. "Biorobotics: Using robots to emulate and investigate agile locomotion". In: *science* 346.6206 (2014), pp. 196–203.

[5] C. D. Remy, K. Buffinton, and R. Siegwart. "A matlab framework for efficient gait creation". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 190–196.

[6] A. Sprowitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert. "Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot". In: *The International Journal of Robotics Research* 32 (2013).

[7] O. von Stryk. *Numerical solution of optimal control problems by direct collocation*. Springer, 1993.

[8] J. Van Den Kieboom, S. Pouya, and A. J. Ijspeert. "Meta Morphic Particle Swarm Optimization". In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*. Springer, 2014, pp. 231–244.

The Math Behind the Model

For our model we adopted a *multi-body dynamics* approach, using exclusively *rigid-bodies*. We recommend using rigid-body systems, because the dynamics are usually sufficiently accurate, and they are relatively easy to model, simulate and build. Non-rigid-body systems still pose a more difficult challenge for modeling, simulation and optimization. If non-rigidity is important to you, consider going straight to hardware!
We use **Euler-Lagrange equations** to derive the equations of motion (EoM). As a reference, we recommend [3]. The resulting dynamics are both highly *non-linear*, as well as being *non-smooth* or in other words, they are described as a *hybrid system*. This is because the dynamics switch abruptly between different governing EoMs (in our case, between a quasi-ballistic flight-phase and the stance-phase), the switch being triggered by specific events (foot touch-down and foot lift-off). While this isn't overly difficult to model and simulate, it presents a big challenge for applying optimization algorithms.
The equations of motion in general form can be seen in equation 1

$$M_{(q)}\ddot{q} = H_{(q,\dot{q})} + B_{(q)}u_{(t)} \tag{1}$$

where q is the vector of generalized coordinates, $\dot{q}$ and $\ddot{q}$ are the corresponding velocities and accelerations respectively, $M_{(q)}$ is the mass matrix, $H_{(q,\dot{q})}$ contains differentiable forces, $B_{(q)}$ is the *control matrix*, $u_{(t)}$ is the vector of *control inputs* i.e. muscles or motors. There are many numerical integration algorithms that can easily handle EoM in this form. However, due to the mass matrix $M_{(q)}$ it is not always easy to see exactly which parameters affects what. We can therefore directly solve equation 1 to obtain explicit EoM in the form of 3.

$$\ddot{q} = M_{(q)}^{-1}(H_{(q,\dot{q})} + B_{(q)}u_{(t)} + J_{c(q)}^{\intercal}\lambda_{GRF}) \tag{2}$$

$$\ddot{q} = \bar{H}_{(q,\dot{q})} + \bar{B}_{(q)}u_{(t)} \tag{3}$$

It should be noted that while Mathematica (running on a modern PC) can solve this quite quickly for the SLIP-and-flywheel model (see poster), for any slightly more complicated model it can take a week to solve or even not return any solution at all! Thus, this can be a very useful tool for understanding simpler models, but it does not scale well to highly non-linear models.
We used **Mathematica** for deriving and analyzing the EoM, as it is very efficient and particularly good at simplifying symbolic math. This can however also be done with **MATLAB**'s symbolic math toolbox, and for deriving the EoM in the form of equation 1 it is equally effective.

Simulation and Optimization

We used the **Gait Creation MATLAB Tool**[5] , an open-source framework designed specifically for legged locomotion, requiring the Symbolical Math and Optimization toolboxes of MATLAB. The framework provides tools to synthesize optimal open-loop control inputs for obtaining periodic (i.e. steady-state) gaits using gradient-based *trajectory-optimization* algorithms: specifically, single-shooting and direct collocation[7]. These methods converge relatively quickly[1]. Also, tools are provided for evaluating the stability of the produced gait by analyzing the resulting Poincaré map (also known as *return maps*). A great advantage of the framework is that it provides several well commented examples which introduce the user to each tool. A disadvantage is that the toolbox may no longer be under active development. More importantly, gradient-based methods require a good initial guess or they may get stuck in local minima, without producing a meaningful gait. We circumvented this problem by first using a *particle swarm optimization* algorithm[8][2] to generate quasi-periodic gaits, which we then used as initial guesses for the gradient-based methods.
Compared to building a robot in hardware, simulation has the following advantages:

- Generally cheaper (just needs a computer), and requires fewer specialized engineers.
- All information is inherently accessible without needing sensors.
- Parameters can be changed virtually, making large-scale optimization more viable.
- Can (sometimes) be run faster than real-time.
- Easier to isolate very specific dynamics, while ignoring the rest.
- If something goes wrong, neither your students nor robots nor risk getting hurt or damaged.

[1]In our case, optimizations generally took several hours up to overnight. We have heard that using specialized solvers such as **SNOPT** can reduce optimization time to mere seconds.