FARMS: Framework for Animal and Robot Modeling and Simulation

Jonathan Arreguit[1†*], **Shravan Tata Ramalingasetty**[1, 2†*], **Auke Ijspeert**[1*]

***For correspondence:**
jonathan.arreguitoneill@epfl.ch;
shravantr@gmail.com;
auke.ijspeert@epfl.ch

[†]These authors contributed equally to this work

[1]BioRob, School of Engineering, Institute of Bioengineering, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland; [2]Department of Neurobiology and Anatomy, College of Medicine, Drexel University, Philadelphia, USA

Abstract

The study of animal locomotion and neuromechanical control offers valuable insights for advancing research in neuroscience, biomechanics, and robotics. We have developed FARMS (Framework for Animal and Robot Modeling and Simulation), an open-source, interdisciplinary framework, designed to facilitate access to neuromechanical simulations for modeling, simulation, and analysis of animal locomotion and bio-inspired robotic systems. By providing an accessible and user-friendly platform, FARMS aims to lower the barriers for researchers to explore the complex interactions between the nervous system, musculoskeletal structures, and their environment. Integrating the MuJoCo physics engine in a modular manner, FARMS enables realistic simulations and fosters collaboration among neuroscientists, biologists, and roboticists. FARMS has already been extensively used to study locomotion in animals such as mice, drosophila, fish, salamanders, and centipedes, serving as a platform to investigate the role of central pattern generators and sensory feedback. This article provides an overview of the FARMS framework, discusses its interdisciplinary approach, showcases its versatility through specific case studies, and highlights its effectiveness in advancing our understanding of locomotion. In particular, we show how we used FARMS to study amphibious locomotion by presenting experimental demonstrations across morphologies and environments based on neural controllers with central pattern generators and sensory feedback circuits models. Overall, the goal of FARMS is to contribute to a deeper understanding of animal locomotion, the development of innovative bio-inspired robotic systems, and promote accessibility in neuromechanical research.

Draft

Introduction

Animal movement is a result of the complex interactions between the body, the nervous system, the sensory modalities, and the environment. Each of the sub-systems have functional properties that emerge only when they interact with each other (***Dickinson, 2000***). Studying individual components reveals only a part of the systems' function. This makes studying animal movement a challenging and inter-disciplinary scientific endeavor.

To test the validity of a proposed hypothesis, relevant data from animal experiments must be collected. However, there are still several technical challenges associated with collecting animal data. These challenges vary depending on the animal species considered. For example in small animals such as *Drosophila Melanogaster*, measuring ground reaction forces during locomotion is currently infeasible with available technologies. Another fundamental complication that comes up with animal experiments are ethical restrictions. Depending on the animal, the ethical restrictions can make data collection impossible. All these challenges associated with animal experiments

point to the importance of properly specifying the exact measurements required to validate a hypothesis before moving to experimental studies to properly assess the feasibility and avoid any unnecessary efforts and/or harm to the animals.

To handle these challenges, the aptitude to accurately predict the outcome of experiments and identify potential causes for failure can be crucial. Furthermore, developing numerical models based on existing knowledge can be a helpful way of systematically addressing the process of generating hypotheses and predictions to validate and further our understanding of animal movement. Models for studying animal movement include three types: 1) (simple) theoretical or analytical models, 2) numerical models, and 3) physical models, i.e. robots that mimic the behavior of animals (***Ijspeert, 2008***, ***2014***). All these approaches play important roles for studying animal movement motor control. Mathematical models such as inverted pendulum models have played an important role in identifying general principles of animal locomotion (***Holmes et al., 2006***), but they are less well suited to study the interaction of multiple components in more details. As robotics is still a relatively new field, there remain technical limitations to faithfully replicate all physical components of an animal body (e.g. its muscles and skeletal structure). Hence physics-based numerical simulations present an interesting alternative and are the focus of this article.

Neuromechanical simulations

Extending the scope of theoretical or mathematical models leads to considering computer simulations, in particular *neuromechanical simulations*, i.e. computer simulations of animals to study their movement that include multiple components such as a physical body, neural models and/or muscles models (***Edwards, 2010***).

Compared to a research approach using a robotics platform, neuromechanical simulations exhibit many advantages: they can operate at the same scales and dynamical properties as the biological experiments, which is something that often remains unfeasible with robots; they can reduce technical and technological limitations, as well as the costs for running the experiments; they do not require any kind of maintenance and repairs, which can be time consuming when running experiments; they provide access to information and facilitate collection of data which could normally be challenging to obtain in complex experimental setups; they can be shared, which is favorable for reproducibility and facilitating collaborations when working in large projects across several institutions. Neuromechanical simulations have nevertheless limitations to keep in mind when drawing conclusions from the results: they require substantial modeling efforts and a computational cost versus accuracy trade-off; and they might require real-world experimental verification and validation to ensure the reliability of the findings.

Neuromechanical simulations cannot fully replace both biological experiments and robot experiments, but they can complement each other for gathering additional information and drawing conclusions.

The idea of using neuromechanical simulations to study animal movement is not new. In the 1990s, Karl Sims showed that diverse animal morphologies and locomotory behaviours could be systematically evolved using genetic algorithms (***Sims, 1994b***,a). ***Geyer and Herr*** (***2010***) studied in 2010 how sensory information could be used to generate bipedal walking patterns in a simple 2D simulation. ***Ekeberg and Pearson*** (***2005***) investigated how reflex rules could trigger a hind limb walking patterns in the cat. ***Daun-Gruhn*** (***2011***) investigated how sensory information could generate the motion of the limbs in a stick insect. ***Hamlet et al.*** (***2018***) investigated how curvature-based control rule combined with sensory information could achieve optimal swimming of the lamprey. There are also several neuromechanical simulations related to salamanders. ***Ijspeert*** (***2001***) investigated using a neuromechanical simulation in 2.5 physics dimensions how the salamander could achieve walking and swimming patterns. ***Harischandra et al.*** (***2010***) investigated how this could also be extrapolated to a 3D simulation of the salamander. They were able to achieve walking and swimming, although using a simplified control approach without neural models. ***Liu et al.*** (***2018***, 2020) achieved a series of different gaits, including walking and swimming like in the previous

Draft

works and also presented how turning could be achieved. And finally, ***Suzuki et al.*** (***2021***) found that using a combination of feed-forward oscillators combined with sensory information, it was possible to achieve different types of gaits that would emerge in a salamander model, like trotting gaits, but also lateral sequence walks, which other models up to then could not explain.

A need for a modeling and simulation framework

From a development perspective, each of the works mentioned above had to re-implement models and code from scratch. They did not benefit from the possibility of sharing a neuromechanical simulation framework. To avoid developing solutions and tools that already exist, there is a strong need to contribute to the infrastructure of the neuromechanical simulation framework to further our understanding in animal motor control. The objective in improving this infrastructure is not to replace the usual paradigms, but to facilitate them. Our goal is to propose an open-source modular framework that builds on existing tools, able to simulate locomotion with musculoskeletal models, biological neural network models, and the appropriate sensors for the studied animals in different environments and able to further the development of neuromechanical models. As will be discussed later, there exist a few other simulation frameworks such as OpenSim and others, but none offered all the options we were aiming at in a single framework (namely the ability to simulate multiple types of morphologies, environments, muscle types, and controller types).

To this aim, we developed FARMS, a Framework for Animals and Robots Modeling and Simulation. This framework is meant to serve as a platform to primarily facilitate the study of locomotion, and to feature a range of tools to help design models with realistic dynamics, to run controllers based on neural models and for comparative studies with the animals and/or robots. It also aims to support optimization methods for parameter tuning and 3D visualisation for debugging and rendering high quality videos for publication. We aim to make Framework for Animals and Robots Modeling and Simulation (FARMS) an open-source community-driven modular project in the future to accustom to the current needs of the scientific community. As such, FARMS could help bridge the gap to connect the diverse works of researchers from different fields, including biomechanics, neuroscience and robotics.

At an implementation level, FARMS leverages and integrates available open-source software into a set of implemented packages developed to facilitate coupling between the different components. In essence, FARMS can be viewed as a basis upon which wrapper code is written to glue existing software together. It was designed with the intention of extensibility and modularity throughout its implementation. More precisely, considerable effort was put into integrating wrapper functions such that users could create and include their own models and controllers and run physics-based simulations, where different physics engines could also be supported. If a certain functionality is unavailable, then a simulator agnostic plugin can be written to cater to the need. The modular approach was also chosen with the intention of reducing required dependencies. For example, the farms_network package used in (***Lobato-Rios et al., 2022***) to simulate sparse neural networks can be used independently as well as integrated with FARMS.

The most important aspect of FARMS is the data standard developed in farms_core. By following a functional programming approach, we break down the neuromechanical simulation into modules that operate on data (for instance, state information of links, joints, contacts, muscles, and external forces). This allows us to make FARMS extensively modular. Furthermore, data is automatically saved throughout simulations. It becomes natural and easy to catalog experiments to be shared and reproduced by the community.

In the following sections, we define the core principles of FARMS, and discuss each of the core modules in detail. Experiments of neuromechanical simulations are then demonstrated to showcase the main capabilities of the framework. Finally, we present some concluding remarks and propose future directions.

Draft

Framework

FARMS is a modular framework written primarily in the *Python* programming language, where various components can be used independently to accustom user needs. It is focused on reusing other available open-source projects wherever possible to take advantage and contribute back to existing infrastructure. By definition, a framework refers to the basic structure on top of which other things are built on. Establishing the fundamental structure allows (i) researchers to define neuromechanical experiments, (ii) developers to extend the functionality of existing framework, and (iii) share and exchange simulation results with the community.

Within FARMS, a neuromechanical simulation at top level can be broken down into four components (***Figure 1***): (i) modeling, (ii) simulation, (iii) analysis, and (iv) data being the bridge connecting the different components. Optimization and learning algorithms constitute an additional layer for manipulating each of these components for producing the desired animal or robot behaviour.

- **Modeling:** Two basic necessities for any neuromechanical simulation are the descriptions of an animal or robot model and its controller for generating motions. Modeling block covers the tools and aspects of generating these sub-blocks.
- **Simulation:** Once both the body and controller are described, the next step consists in performing either a forward or inverse dynamics physics simulation with the controller in the loop. Simulation involves integrating (i) a physics engine to simulate articulated rigid bodies, interactions with the environment (e.g. land or water) and (ii) the control laws to compute the actions to generate movements.
- **Analysis:** Finally, during and at the end of a simulation, the progress of an experiment can be monitored though the collected data. The analysis component deals with aspects of performing post-processing, visualising and replaying simulation data, and producing representations for scientific analysis and sharing.
- **Data:** At every step of the framework, the information flow is recorded and tracked. This is performed with the data block. Each module can read and write to it. This enables FARMS' modularity as data can be shared between blocks and can be populated by any independent tool.

Data

Data structures are the backbone of any numerical simulation. In FARMS, the data block maintains and records information at every step of the simulation process. It decouples the different steps in a neuromechanical simulation and makes FARMS a truly modular framework. There are several other advantages to this approach. By recording all data, simulations become reproducible not only by the user but by the community. Accessibility to the shared data permits reading the information and recreating the original experiment starting from modeling to analysis phase in any order. The Hierarchical Data Format (HDF5) is used to store and organize this data, permitting to reduce disk space usage (***The HDF Group, 1997***).

Code distribution in packages

FARMS has been split into a series of packages to leverage modularity, enabling the users to pick the specific components required for their experiments, and to facilitate the development and additions of any contributor. The main package that glues the components of FARMS together is farms_core as it focuses on standardizing the data structures that are required for connecting the information between the different modules. In particular, Cython was used in order to benefit from the advantages of a compiled language and run faster simulations, while keeping the code simple for developers and tightly integrated with the Python language.

Draft

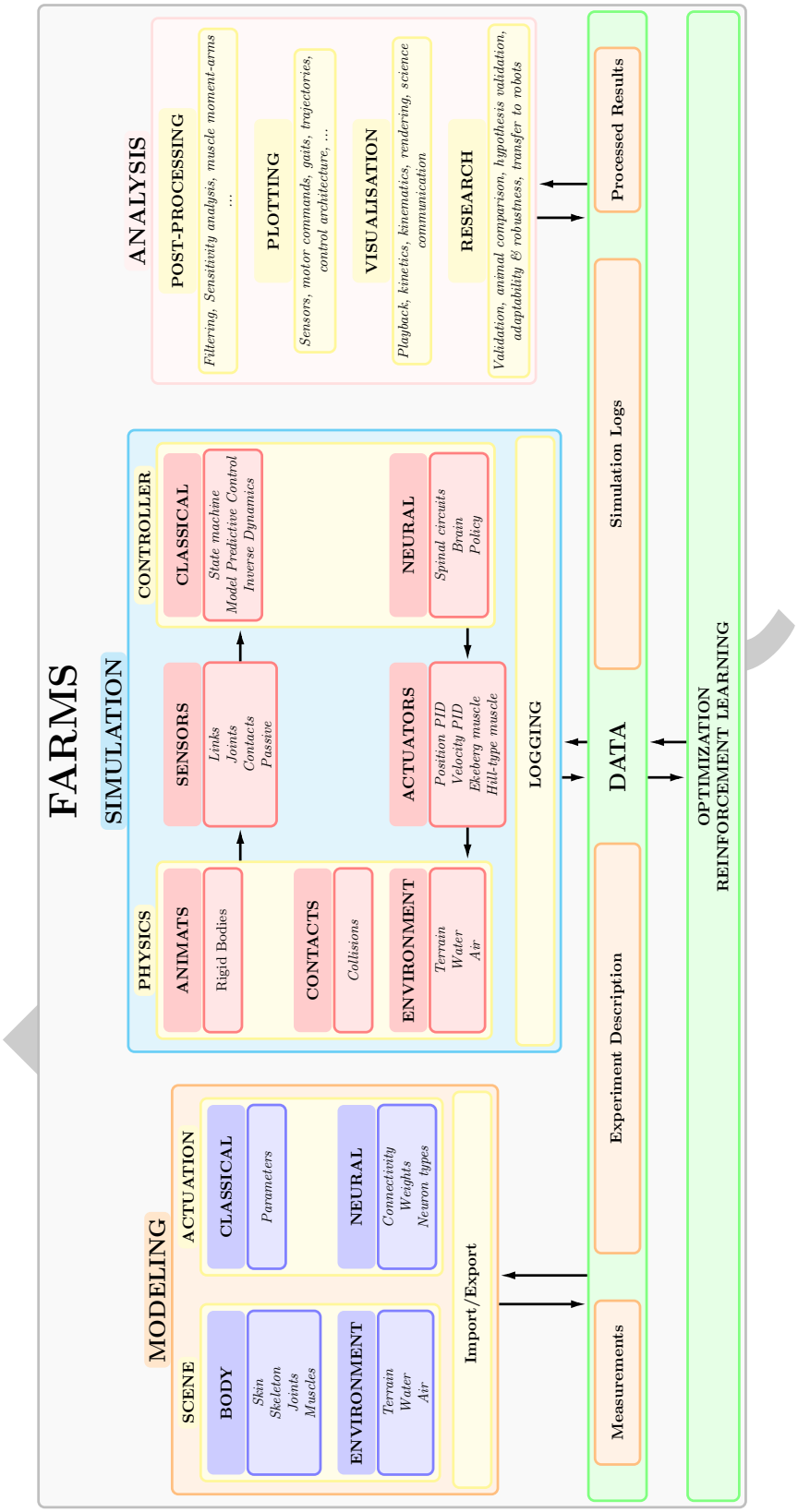


Figure 1. FARMS Architecture

Modeling

A neuromechanical simulation can be described by the following minimum elements: (i) Animal or robot bodies, where in this case the body represents both the geometric and dynamical properties, (ii) a controller, and (iii) the environment with which the animal or robot interacts. Modeling each of the three elements is crucial for successfully running neuromechanical simulations but is often taken for granted to be readily available. Moreover, generating models is itself an iterative process and models need to be regularly modified when new measurements or experimental data are available. This implies that there is an important requirement for rapidly reconfiguring the models when a simulation fails or does not agree with experimental data.

Bodies

An experiment can consist of one or more bodies, including animals, robots or both. As modeling a robot or an animal in FARMS is treated in the same way overall, they will be referred to as **animats** hereafter.

Animat

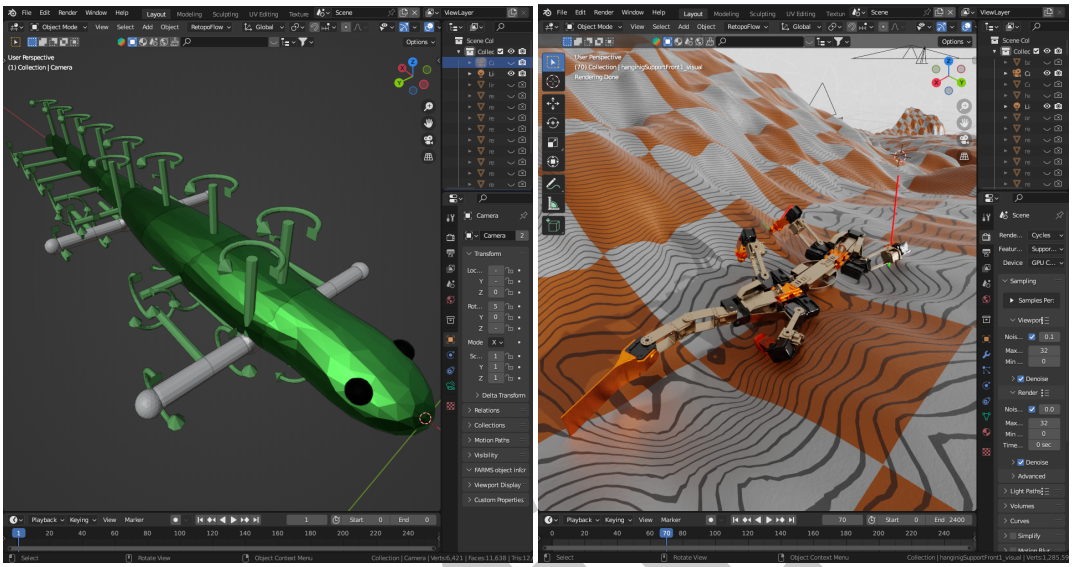
An animat consists of the following elements:

- Skin: An external covering that interacts with the environment. For robots in general, the skeleton directly interacts with the environment and it can be considered as the skin. A special case for a robot with skin is described by ***Thandiackal et al.*** (***2021***), where the water-proof suit of an eel-like robot would be considered as the skin.
- Skeleton: A collection of elements (rigid-bodies) that forms the structural frame to support the entire body. Individual elements are described by their location and inertial elements (mass, centre-of-mass, inertia).
- Joints: The articulation of the body emerges from the definition of joints. There are four types of joints; ball-and-socket, slide, hinge, and a "free joint" which creates floating bodies. Joints define the kinematic and dynamic relationship between any two elements of a skeleton.
- Actuators: Apply forces on the body to generate motion. These can be motors (that apply torques or forces at the joints) or muscles (that apply forces on the skeleton).

Even though we represent an animal and a robot under the same description as an animat, the necessary information to define them is obtained differently. Robots are typically designed by humans in Computer Aided Design (CAD) software and usually have a well-defined fixed geometry. However, animal bodies are reconstructed either based on specimen scans [Computerized Tomography (CT) or Magnetic Resonance Imaging (MRI)] or anatomical reference data and requires careful consideration of the intra-animal variability that exists. The scans are then exported to CAD software, where they are reworked to fit a desired simulation file format. Making use of common standards for robot simulation, FARMS relies primarily on the Simulation Description Format (SDF), originally developed by the Open Source Robotics Foundation for the Gazebo robot simulator (***Koenig and Howard, 2004***). This format is commonly used for robotics simulations and contains all the elements required to define the animat models in FARMS. While SDF is the main format used for all the models shown in this article, the modularity of FARMS permits integrating support for other formats as well. For example, preliminary support is already present for Unified Robotics Description Format (URDF) and MuJoCo Format (MJCF), two other commonly used formats in robotics simulation originally developed for Gazebo and MujoCo (***Todorov et al., 2012***) respectively. *OSIM*, and *WBT* are some additional examples of commonly used file formats developed for OpenSim (***Seth et al., 2011***) and Webots (***Michel, 2004***) respectively. While those formats are not officially supported yet, it is possible to convert those models to be compatible with FARMS (See farms_osim) and further integration remains a future possibility. Finally, as converting from CAD software to any of these file formats is not always straightforward, a package to facilitate this step, or for the design of new animats in general, is described in the following section.

Draft

Blender addon



(a) Modeling **(b)** Simulation visualization and rendering

Figure 2. Blender for modeling and rendering using the farms_blender addon. The modeling example shows the salamander with a display of the joints' locations and their rotations. The rendering on the right shows a view of a simulation of Krock, a crocodile-like robot, in an uneven terrain after running the simulation and logging data to allow playback within Blender.

The farms_blender package is a Blender (***Community, 2022***) addon, providing an interface to users to generate the models and visualize completed simulations, as shown in ***Figure 2b***. SDF, URDF and MJCF are based on Extensible Markup Language (XML) and are typically written in a text editor. The interface provides the ability to set all the links' poses and geometries, the joints' poses, as well as the inertials through an interface to facilitate this step. Visual guides can help with validation, as shown in ***Figure 3***.

The addon was also developed to fully replay the results of the simulation in Blender. This provides a useful interface for reviewing the results of a simulation, showing and overlaying additional information such as interaction forces with the environment, and rendering the simulation results to video for high-quality publications.

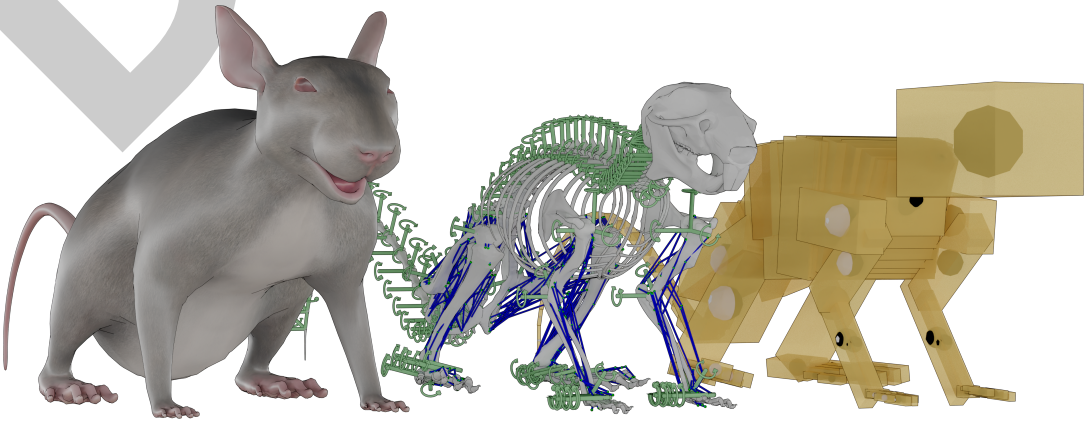


Figure 3. Scene modeling in FARMS with Blender. From left to right visualizes the following elements of a Mouse body (i) skin, (ii) skeleton, joints and muscles, (iii) inertials (inertia tensor, mass and center-of-mass)

Draft

Controller

A controller defines the elements that send the commands to the actuators that will ultimately determine how the physical elements move over time. To study animal and robotics movement, several different control strategies are possible. The possible actuation methods are (i) model based control and (ii) neural control. Model based control methods are mainly derived from control theory developed for robotics. These include methods like inverse dynamics, state-machines and Model Predictive Control (MPC). Our expertise relies mainly on neural control, which is why controllers based on neural circuits with Central Pattern Generators (CPGs) and sensory feedback loops are mainly described and presented in the following sections. The use of neural controllers based on deep neural networks using Reinforcement Learning (RL) is also common in recent works. Naturally, the framework permits the ability to implement the other types of controllers as well.

Networks

Neural control is more oriented towards neural networks either inspired from biology or machine learning techniques such as RL. Modeling them involves describing the type of neurons, their properties and their connectivity. FARMS allows for an easy integration of external python libraries to simulate the networks of choice within the controller module. Currently, there is a lot of support for machine learning libraries and modeling artificial neural networks would mainly rely on those existing libraries. For biological networks, we could broadly classify them into spiking and non-spiking neural networks. For spiking networks, there are open-source libraries such as NEST (***Gewaltig and Diesmann, 2007***), Brian (***Stimberg et al., 2019***) or Neuron (***Hines and Carnevale, 1997***). For sparse, non-spiking networks there is no library to the best of our knowledge that supports modeling and simulation of locomotion networks. As part of FARMS, we developed a platform and simulator agnostic sparse neural network software package called farms_network. Farms_network provides commonly used neural models for locomotion circuits. ***Figure 4*** shows an example of sparse locomotor network based on ***Danner et al.*** (***2017***) modeled and visualized using the library developed as part of FARMS. The farms_network package provides standard neural models at different levels of abstraction which have been commonly used in past research, such as phase oscillators (***Kuramoto, 1975***; ***Ijspeert et al., 2007***), Hopf oscillator (***Righetti et al., 2006***), Matsuoka model (***Matsuoka, 1985***), FitzHugh–Nagumo model (***FitzHugh, 1961***), Leaky-Integrator model (***Ijspeert, 2001***), Hodgkin-Huxley model (***Hodgkin and Huxley, 1952***) and non-spiking activity-based model (***Ermentrout, 1994***).

Actuation

In FARMS, the controller also includes the actuation system, which constitutes the motor dynamics for a robot or the muscle for an animal. In the case of animal locomotion, the biomechanics can play an important role for obtaining the proper gait patterns. Muscles are not a feature commonly included as part of physics engines as most of the physics engines are in practice built for robotics communities. As part of FARMS, we developed the farms_muscle package to provide users with different muscle models. Two particular types of muscles commonly used in literature have been implemented, Hill-type muscle model (***De Groote et al., 2016***; ***Geyer and Herr, 2010***; ***Millard et al., 2013***) at different levels of abstractions in a computationally efficient manner, and the Ekeberg antagonistic muscle pair model (***Ekeberg, 1993***).

Environment

The physics engine, discussed in the physics ***subsection*** , is responsible for computing the dynamics of the bodies and their interactions with the environment. With FARMS, we aim to support different environmental media, including land, water, sand and air.

Draft

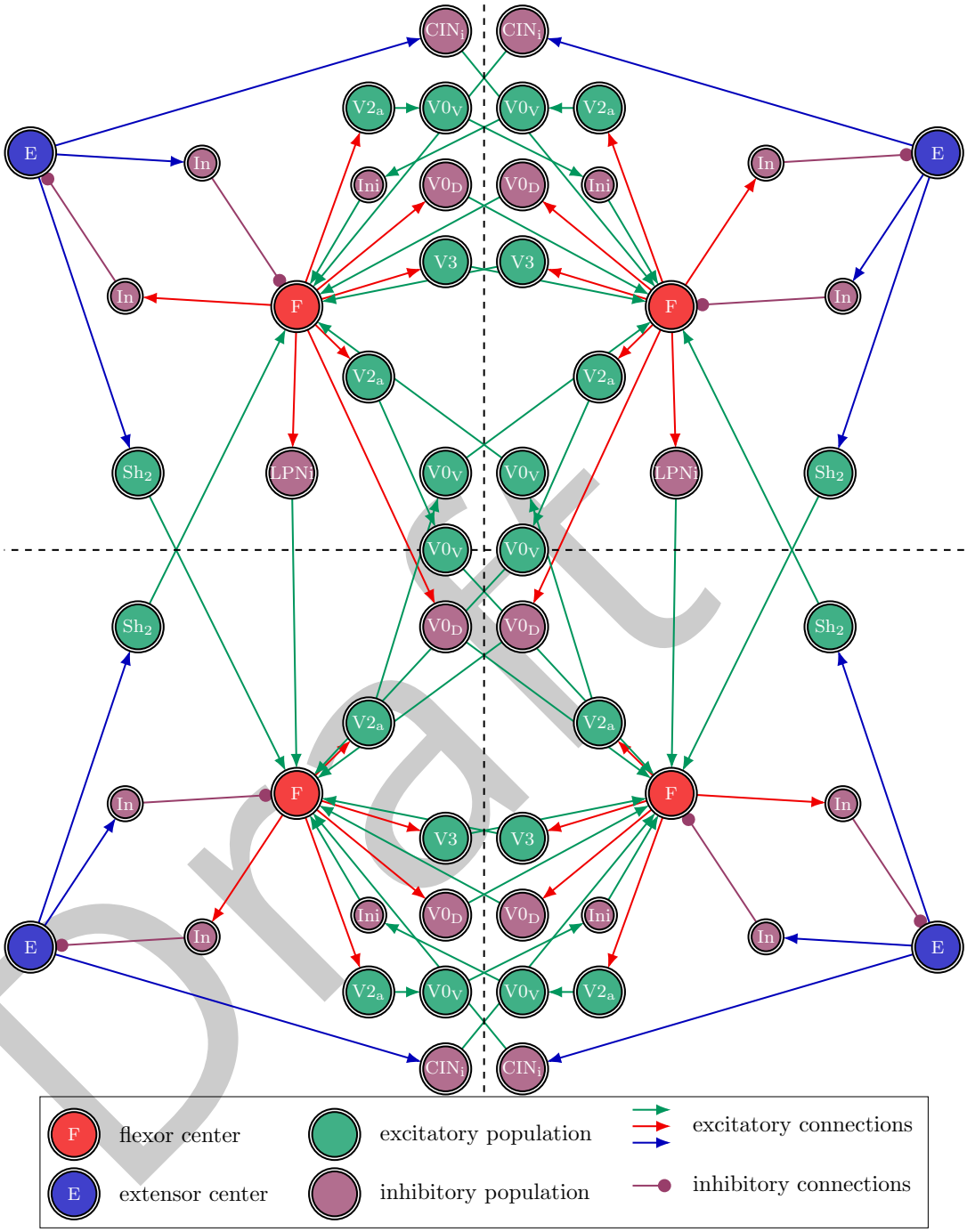


Figure 4. Example of neural network generation and visualization in FARMS. The network is a computational model of central spinal circuits for quadruped limb coordination proposed by ***Danner et al.*** (***2017***)

Arena

Most physics engines destined for robot simulations only support rigid bodies, meaning one is restricted to modeling rigid terrains. Some simulators support simplified hydrodynamic environments based on drag forces as a computationally simple approach to emulating interaction with fluids, which is supported in FARMS. For more complex interactions, we aim to fill this gap by developing a physics simulator agnostic to water, sand, and air media using particle-based simulations. While modeling the environment, it is thus necessary to define the type of the medium and its

Draft

properties.

Custom physics

As part of FARMS, we provide access to both phenomenological models of fluid dynamics with a drag model in farms_bullet and farms_mujoco, and more sophisticated fluid simulations based on Smoothed-Particle Hydrodynamics (SPH) methods (***Bender, 2017***; ***Puri et al., 2013***; ***Ramachandran et al., 2021***), with preliminary support which can be found in farms_sph. More precisely, farms_sph provides an interface to the PySPH library (***Puri et al., 2013***; ***Ramachandran et al., 2021***) and integrates alongside the rigid physics engine similarly to (***Angelidis et al., 2022***) using two-way coupling.

The drag model is given by

$$\vec{f}_i = \vec{c}_{f,i} \odot \vec{v}_i^2 \tag{1}$$

$$\vec{\tau}_i = \vec{c}_{\tau,i} \odot \vec{\omega}_i^2, \tag{2}$$

where f_i is the force and τ_i it the torque applied on link i, v_i and ω_i are respectively the linear and angular velocities of link i in local frame and c_f and c_τ are drag coefficients, which are multiplied to the velocities using the Hadamard product.

Support for buoyancy can also be applied with a vertical force depending on the link volume. For both the drag model and the SPH integration, the forces calculated by the fluid models are computed independently of the rigid-body physics engine and are applied as external forces with the engine such that the rigid-body physics engine takes care of integrating the equations of motion. This step is repeated at each physics step. As of this date, the PySPH library supports simulating of sand and gas dynamics, see ***Ramachandran et al.*** (***2021***) for more information. While sand and air are not officially supported yet, the same approach can be used to implement them.

Simulation

Once the animat, the controller, and the environment are fully described, the next step is to perform numerical simulations. For neuromechanical simulations, this step mainly involves accounting for the controller commands and performing forward simulation of an animat model with the physics simulator, where the physics constraints must be solved and the dynamics integrated over time. Leveraging existing physics engines, FARMS provides wrappers that help bridge modalities of the simulation block.

Physics engines

The physics engine is the core of a neuromechanical simulation. The common steps performed by a physics engine are described by ***Geijtenbeek*** (***2013***) as:

- *Collision detection* checks if the body intersects with itself or other physical objects in the scene.
- *Forward dynamics simulations* computes the linear and angular accelerations of each simulated body.
- *Numerical integration* update positions, orientations, and velocities of bodies, based on the previously computed accelerations.

The steps outlined above simulate the articulated rigid bodies and their interactions with rigid terrain. The choice of physics engine/simulator is extremely important for stable simulations of the specific experiment. There are several simulators that employ different strategies to perform the three steps described above to perform the numerical simulation.

Table 1 shows a short list of the available physics engines commonly used in animal and robot simulations. The unique prospective of FARMS is its potential to provide a common interface to multiple of these simulators. FARMS supports Bullet (***Coumans and Bai, 2016***) with farms_bullet and MuJoCo (***Todorov et al., 2012***) with farms_mujoco as its primary physics engines. We explored

Draft

Table 1. Overview and comparison of features for different physics engines/simulators.

	Webots	Gazebo	OpenSim	PyBullet	MuJoCo	Scone	Isaac	MyoSuite	FARMS
Release	1996	2004	2007	2003	2015	2019	2021	2022	2023
Physics engine	ODE	Dart Bullet	Simbody	Bullet	MuJoCo	Simbody HyFyDy	PhysX	MuJoCo	MuJoCo Bullet
Programing languages	C/C++ Python	C++	C++ Python	C++ Python	C/C++ Python	Lua Python	C++ Python	Python	Python Cython
Muscles	No	No	Yes	No	Yes	Yes	No	Yes	Yes
Neural models	No	No	No	No	No	No	No	No	Yes
Data logging	No	Yes (ROS)	No	No	No	Yes	Yes	Yes	Yes
Fluids	Drag	Partial	No	No	Drag	No	?	No	Drag/SPH
GUI/Headless	GUI	Both	Both	Both	Both	GUI	Both	Yes	Both
Playback	No	Rosbag	No	No	No	Yes	Yes	Yes	Blender
Plotting	Live	Yes (ROS)	Static	No	Live	Static	?	No	Static
Optimization	No	No	No	No	No	Yes	Gym	Gym	Yes
Inv. dyn.	No	No	Yes	Yes	Yes	Yes	?	No	Yes
Modeling	Yes	Yes	Yes	No	No	No	Yes	No	Yes
License	Apache2.0	Apache2.0	Apache2.0	zlib	Apache2.0	Apache2.0	Propr.	Apache2.0	Apache2.0

several of these frameworks and physics engines in the past years in parallel to our research. Starting with Webots (***Michel, 2004***), we found that its use of the ODE physics engine made it unstable for complex models with large numbers of links as the ODE physics engine solves in maximal coordinates. We also explored Gazebo (***Koenig and Howard, 2004***) before its change to ignition, which was an interesting option for its availability of four physics engines at the time, but was not ideal due to its lack of cross-platform support when working with collaborators. OpenSim (***Seth et al., 2011***) was interesting for its availability of muscle models, but is mainly destined towards the community studying human motion as a standard human musculoskeletal model is available, but integrating new models was found more challenging than with other physics engine and ran slower for equivalent simulations. On the other hand, PyBullet (***Coumans and Bai, 2016***) was particularly easy to use to create new models, although it lacked some features required for the neuromechanical simulations that were planned, such as muscle and fluid models. However, these missing features could be implemented within FARMS, and we created a farms_bullet package interfacing to the Bullet engine. During our experience with PyBullet, a drawback found was joint damping, which is an essential component needed for stability with muscle models. In PyBullet, joint damping is implemented in an explicit manner, rather than with an implicit formulation, leading to instabilities for high values of damping. We later integrated MuJoCo (***Todorov et al., 2012***) in farms_mujoco when it was released as open-source, as we found that it featured an implicit implementation of joints damping and permitted for stable simulations, facilitating tuning and simulation of muscle-controlled neuromechanical experiments. To summarize, the Bullet and MuJoCo physics engines were mainly selected for their ease of installation and use, their ease for setting up new models, their capability to simulate rigid multi-bodies in reduced coordinates for improved stability when a high number of links are present in an animat, and their computational speed when solving animat locomotion simulations.

In the more recent years, new physics engines such as Scone (***Geijtenbeek, 2019***), Isaac sim (***Liang et al., 2018***), as well as others have been released. As MuJoCo was found to be suitable for our current research, we have not yet explored in detail these new engines. While support for the other engines presented in ***Table 1*** is not yet present, FARMS' modularity could still permit providing an interface to them in the future depending on research needs, as these simulation platforms also include some features of interest. A framework providing access to multiple physics engines / simulators could be beneficial as this could for example facilitate validations. For a comprehensive review on physics simulators from a robotics perspective, refer to ***Collins et al.*** (***2021***).

Draft

Simulation step

Sensors

At the end of a time step of the numerical integration of the physics block, sensors implemented in the FARMS framework interface collect the complete information of the state of the simulation. This includes exteroceptive signals such as contact forces, ground reaction forces and proprioceptive signals such as muscle spindle feedbacks, joint angles and joint velocities.

Controller

Depending on the decisions made during the modeling phase, the simulation can either be running model based control or a neural control. For either of these approaches, the controller block will have complete access to the current state of the simulation through the sensors block. The controller can compute the necessary actions in the following time step of the simulation.

Actuators

The outputs of the controller block can be (i) muscle activation commands, (ii) desired joint positions and velocities or (iii) desired joint torques. FARMS will ensure the correct actuation is performed depending on the definitions of the body and control in the modeling phase.

Logging

At each of the four stages described above, the complete state of the block at every time step is logged into the data block. This ensures complete reproducibility of the simulation at any point of time. Logging simulation data is extremely important for further analysis and post-processing and this is seamlessly ensured by the FARMS framework.

Optimization

One critical aspect of modeling is defining the parameters of a model. Neuromechanical simulations offer the advantage that they permit exploring the role of the parameters for various components of an experiment, including the morphology and controller of an animat or the role of the environment and its physics. In engineering and research applications, it is common to seek a specific behavior out of the simulation, but finding the parameters to obtain this behavior can be cumbersome. Inspired by (***Auerbach et al., 2014***), we implemented support for evolutionary algorithms in farms_evolution. Indeed, simulations can be coupled with evolutionary algorithms and fitness functions can be defined to tune the parameters of neural and muscular models, which can be difficult to measure from the animals directly or dangerous to test on robot hardware.

Evolutionary algorithms have been employed in a large number of works including studies of lampreys (***Ijspeert and Kodjabachian, 1999***), salamanders (***Ijspeert, 2001***), ants (***Guo et al., 2018***) or humans (***Dzeladini et al., 2014***). More precisely, (***Ijspeert and Kodjabachian, 1999***; ***Ijspeert, 2001***) use Genetic Algorithms, (***Guo et al., 2018***) uses Covariance Matrix Adaptation Evolution Strategy and (***Dzeladini et al., 2014***) uses Particle Swarm Optimization, all of which optimize a single cost function. The farms_evolution packages leverages the Pymoo library (***Blank and Deb, 2020***), which provides a range of algorithms for single and multi-objective optimization, with or without constraints, for optimising the parameters of for a simulation task.

Analysis

At the end of a successful or unsuccessful simulation, it is important to analyse the performance. Tools that help visualize and show data in a manner that allows for critical analysis will be part of the FARMS analysis toolbox. The analysis block can be subdivided into (i) post-processing, (ii) plotting, (iii) visualization and (iv) outreach. With the data containing the complete model information and the simulation logs, the analysis block can combine this information into useful ways to represent data for neuromechanical simulations.

Draft

Plotting and visualisation aspects are mainly performed using Blender (***Community, 2022***). This includes replaying the simulation by overlaying additional data computed by the post-processing block to provide deeper insights to the researcher. High-quality visualization tools go a long way in helping researchers in understanding the extremely high dimensional information gathered during the simulations.

Demonstrations

In the following sections, we demonstrate various experiments with different animats, using different controllers in different environments to showcase the features and capabilities that FARMS provides. Videos of the results for each of the experiments are distributed separately.

Robots

We developed and support a set of robots that are used in the Biorobotics laboratory at EPFL. An illustration showing the different robots that we work with is found in ***Figure 5***. These robots are primarily based on eel and salamander-like morphologies, although we plan to also support additional robots morphologies in the future which are actively being designed and made available in the community.

Figure 5. Examples of robots in FARMS These robots have also been featured in literature; Pleurobot in ***Karakasiliotis et al.*** (***2016***), Orobot in ***Nyakatura et al.*** (***2019***), K-rock in ***Melo et al.*** (***2017***), Salamandra Robotica II in ***Crespi et al.*** (***2013***), Agnathax in ***Thandiackal et al.*** (***2021***) and Amphibot in ***Crespi et al.*** (***2005***); ***Crespi and Ijspeert*** (***2006***). Pleurobot and Salamandra Robotica II have been used in previous works to study salamander locomotion. The salamander model shown in the center allows for a size comparison between these robots and the studied animal.

We replicated the behaviors presented in ***Ijspeert et al.*** (***2007***), where a salamander-like robot, Salamandra Robotica II (***Crespi et al., 2013***), is capable of transitioning from swimming in water to walking on land and vice versa. To generate these different locomotor patterns, the controller uses a network of weakly-connected oscillators, based on central pattern generators, and replicated from the work of ***Ijspeert et al.*** (***2007***) which simulate the spinal cord dynamics.

The simulation results obtained for transitions from swimming to walking are shown in ***Figure 6***. This experiment was performed using the MuJoCo physics engine. An arena featuring a ramp and a body of water was created such that the simulated robot could go from one media to the other. To demonstrate how FARMS can include custom physics through MuJoCo's external forces, the forces from the water are applied using a simple drag model provided by ***Equation 1*** and ***Equation 2***, and a buoyancy force allows the robot to swim at the surface of the water given its mass density. The

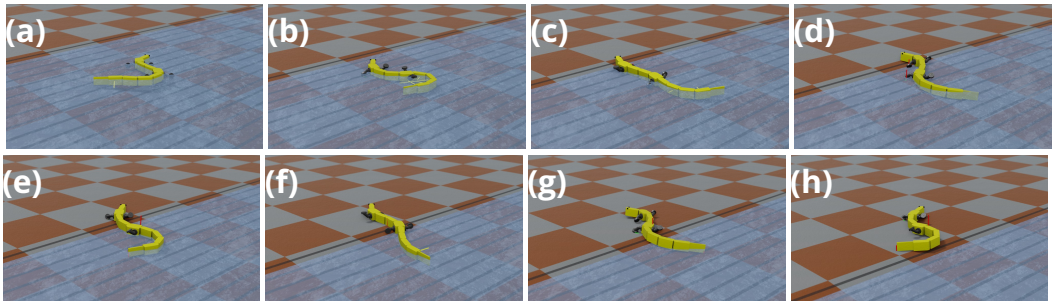


Figure 6. Salamandra Robotica II performing a swim-to-walk transition Sequence of rendered snapshots shown at different times separated by 1.5 [s]. The robot is swimming from right to left in 6a and comes into contact with the ramp in 6b. In 6c the robot transitions from swimming to walking by changing the descending drive commands and the limbs activate. The trunk oscillators synchronize with the limb oscillators, inducing the body to switch from a travelling wave to a standing wave. From 6c to 6h, the robot has switched to a trotting gait and walks out of the water.

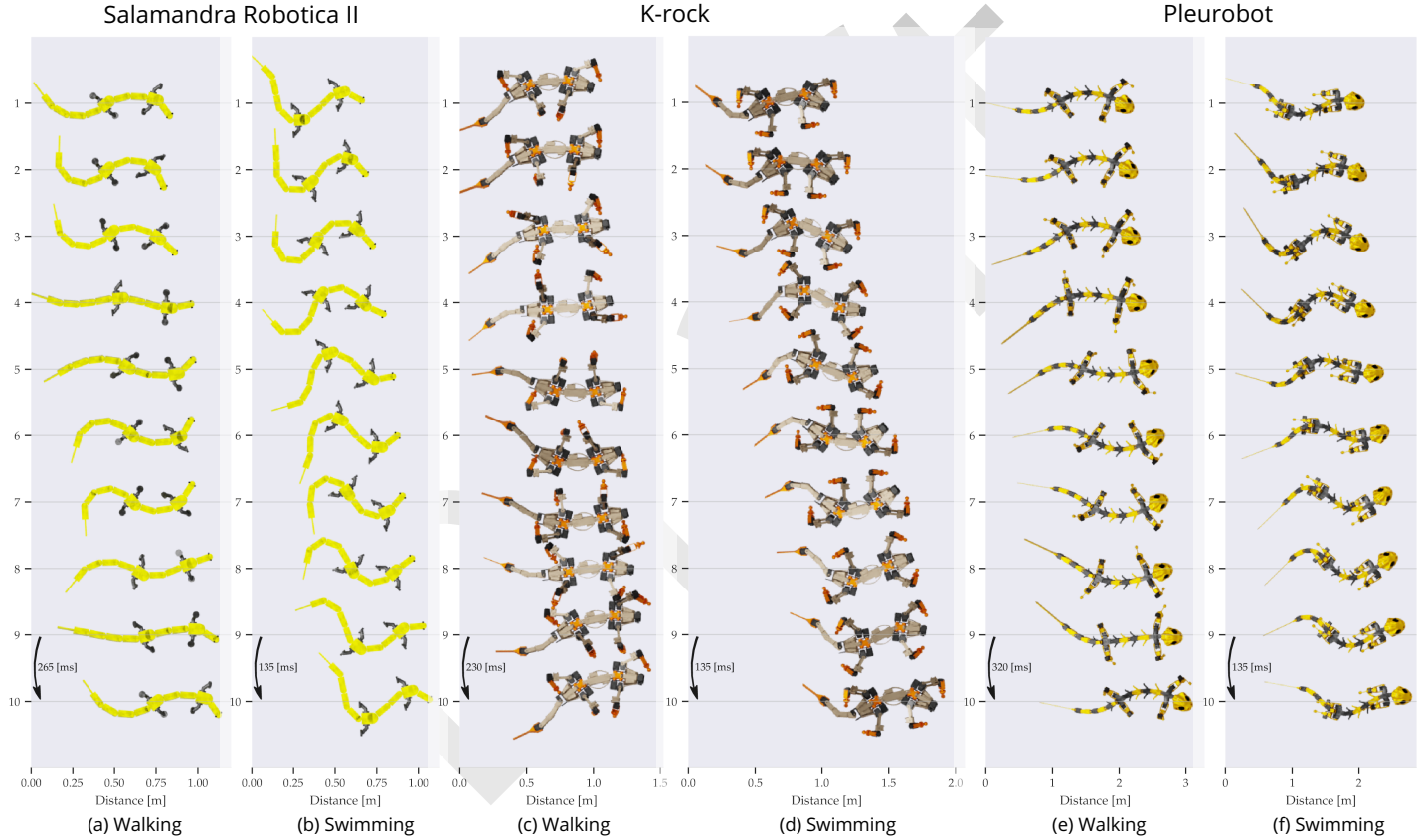


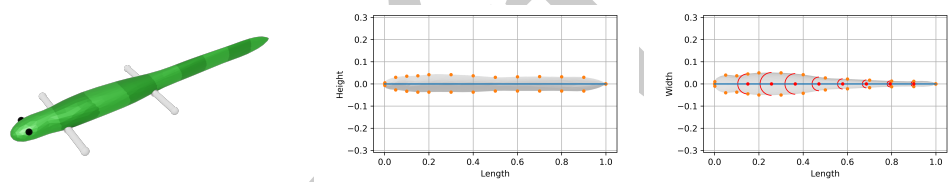
Figure 7. Salamandra Robotica II, K-rock and Pleurobot walking and swimming gaits in simulation. The robots are shown from above with snapshots along time. The spines for each robot exhibit travelling waves which allow them to propel themselves forward in water.

descending drive signals were modulated based on passing a threshold value of contact with the ground such that walking patterns would be generated when the robot is in contact with the ramp. Similarly to the original work of ***Ijspeert et al.*** (***2007***), we find in ***Figure 6*** that the simulated robot is successfully capable of swimming in the water and switching to a walking pattern once it reaches the ground and is then capable of walking itself out of the water. All the features demonstrated here are available through FARMS, including the robot model, the ramp arena, the custom fluid physics, and the oscillator controller.

By adapting the proposed oscillator controller to the morphology of the other robots, we show in ***Figure 7*** that walking and swimming gaits can also be obtained with the K-rock and Pleurobot robots models using an adapted network architecture. The particular difference between these robots and Salamandra Robotica II is that they have a higher number of Degrees of Freedoms (DoFs) in the limbs and different numbers of joints within the trunk as well. While still using a double chain of oscillators for the trunk joints, the network is adapted by also using pairs of oscillators for each joint of the limbs.

Salamander

Extending the salamander-like robots to animal simulations, a salamander model was designed based on the work of ***Harischandra et al.*** (***2010***). By leveraging the Blender 3D modeling software with the farms_blender package, we provide tools for generating a 3D shape, as illustrated in ***Figure 8***. This resulted in an approximate, but comparable shape to the animal. The mass was computed assuming constant density close to water such that the models could float slightly below the surface. The resulting salamander model is 10 [cm] long and weighs 3 [g]. It contains 11 body joints allowing the body to bend laterally and 4 DoF for each limb, including 3 DoF at the interface with the body, and one DoF at the elbow/knee.



(a) Salamander model morphology **(b)** Side **(c)** Top

Figure 8. Salamander shape validation and a render of its final morphology In the two views, side and top, the orange points represent the measurements obtained from the animal. The greyed area represents the final shape of the generated 3D Model. The red points represent the position of the joints along the body. Spherical cuts are applied to the body such that no mass overlap is present when the joints are rotated, which is a useful property for swimming applications as density remains constant.

Similarly to the robots demonstrations, a controller was implemented to showcase walking and swimming using an oscillator network. However, contrary to the robots presented above in position control, each of the DoF for the salamander is modeled as a revolute joint controlled by a flexion-extension muscle pair model proposed by ***Ekeberg*** (***1993***) and governed by:

$$\tau_j = \underbrace{\alpha_j(M_{F,j} - M_{E,j}) + \beta_j(M_{F,j} + M_{E,j})\Delta\varphi_j}_{\tau_{active,j}} + \underbrace{\beta_j\gamma_j\Delta\varphi_j + \delta_j\dot{\varphi}_j}_{\tau_{passive,j}}, \tag{3}$$

where, similarly to ***Ijspeert and Kodjabachian*** (***1999***); ***Ijspeert*** (***2001***), α_j represents the active gain, β_j the stiffness gain, γ_j the intrinsic stiffness, δ_j the damping coefficient, $M_{F,j}$ and $M_{E,j}$ the neural output activities for the flexor and extensor obtained from a pair of oscillators, $\Delta\varphi_j$ the position and $\dot{\varphi}_j$ the velocity for each joint j. In particular, we differentiate between the active torque $\tau_{active,j}$ and passive torque $\tau_{passive,j}$.

Just like for the previous demonstrations, this model is also capable of exhibiting both swimming and walking patterns based on a descending drive signal. The main difference is that all the robots in the previous experiments were position controlled, while the joints of the salamander are torque controlled using the muscle implementation from ***Equation 3***. For properly simulating the dynamics of the Ekeberg muscle model, we found that the MuJoCo physics engine was beneficial thanks to its implicit implementation of damping compared to the Bullet engine. Indeed, increasing to high values of damping δ_j would lead to stable expected behaviors in MuJoCo, while would lead to jitters and numerical instabilities in the Bullet physics engine.

Draft

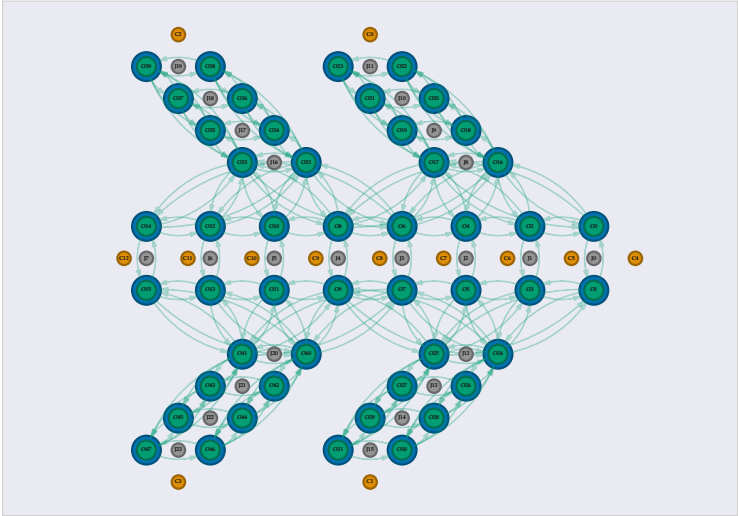


Figure 9. Salamander connectivity for an oscillator network. The network is inspired from the work of ***Ijspeert et al.*** (***2007***). Here, we explore a connectivity based on weak couplings where oscillators are only connected to neighbors. One of the key differences with ***Ijspeert et al.*** (***2007***) is that the limb oscillators have no direct couplings between them. Although it is known that long-range connections between the limb CPGs exist, here we found that this connectivity was sufficient for the salamander model to achieve walking and swimming patterns, as shown in the demonstrations below.

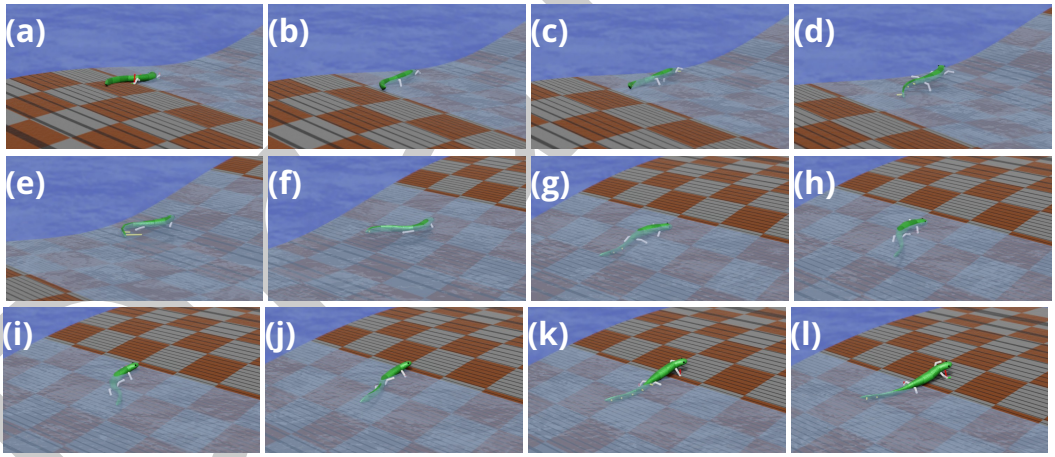


Figure 10. Simulated salamander crossing a river Sequence of rendered snapshots separated by 1.0 [s]. A salamander is simulated in an arena featuring a river, where a ground created using a heightmap with two banks separated by a river is found. The water has a velocity of 3 [cm/s] flowing from left to right. The salamander is initialized on one side of the river and moves towards the other side. In 10a, the salamander comes into contact with the water, but does not switch to a swimming gait until 10d, simiarly to the experiment with Salamandra Robotica II in ***Figure 15***. From 10d to 10g it swims to the other side pushed by the water flow until it comes into contact with the ground and switches to a walking gait. From 10g to 10l it walks up and out of the water.

In ***Figure 10***, We demonstrate how the salamander can cross a river, and thus perform walking-to-swimming and swimming-to-walking transitions. In that experiment, a water current is also present and the appropriate descending signals are computed to drive the model to follow a path to the other side of the river while switching to the appropriate gait.

Combining the oscillator network with a muscle was found to yield stable walking gaits. We demonstrate in ***Figure 11*** that the salamander is capable of walking over a rough terrain. In this experiment, the passive properties of the muscles are useful for providing compliance to the limbs

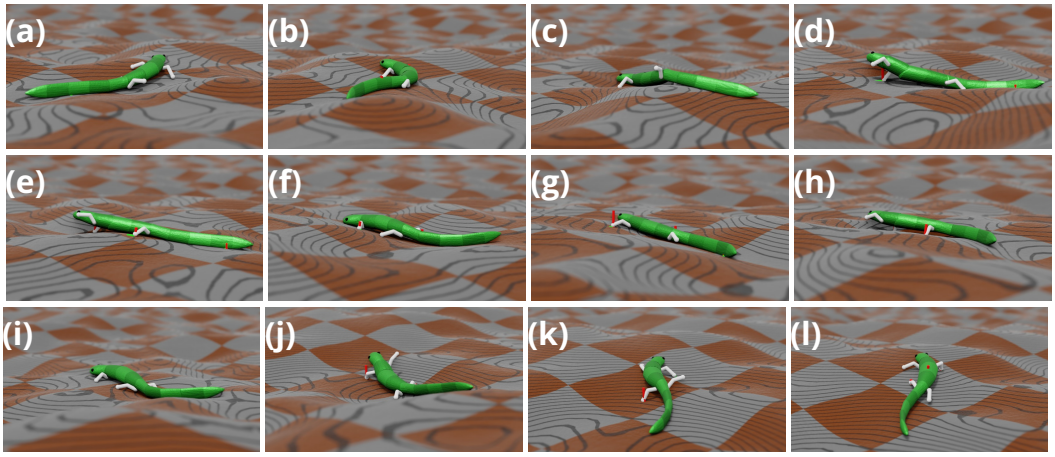


Figure 11. Simulated salamander walking in rough terrain Sequence of rendered snapshots separated by 5.0 [s]. A salamander crosses a terrain made with a heightmap generated with Perlin noise. Using the network shown in ***Figure 16***, the salamander is capable of robustly walking across the terrain. Its ability to navigate this terrain is jointly due to its sprawling posture and passive muscle properties that allow it to bend along the terrain deformation.

and permit the limbs to interact with the ground without predefined trajectories. This model is walking blindly without any knowledge of the surrounding terrain, but shows how it can handle it in a robust manner.

Snake

To extend the salamander results, we also demonstrate how the oscillator network and the passive properties of a muscle model (***Ekeberg, 1993***) can lead to obstacle-based locomotion. A flat arena is set with regularly placed vertical circular pegs with a fixed radius along a grid, as shown in ***Figure 12***. The friction coefficient between the body and the arena is isotropic and of 0.2 such that the snake model can slip with ease, but can not advance without pushing on the surrounding pegs. Using a simple double-chained oscillator network for a limbless snake-like body, i.e. the same controller as the salamander but without the limb oscillators, the model is capable of traversing the obstacle configuration. As the model escapes the pegs area, it is unable to move beyond as it cannot push itself against obstacles anymore. Additional experiments for different arena configurations with different values of pegs radius and spacing are provided in the appendix in ***Figure 17***, ***Figure 18*** and ***Figure 19***. Each of these experiments were performed with no changes in descending signals using the same oscillator network with fixed parameters. This showcases how passive properties of the body can be simulated and interact with the environment with many points of contacts, both with the ground and the pegs, when simulated using the MuJoCo physics engine.

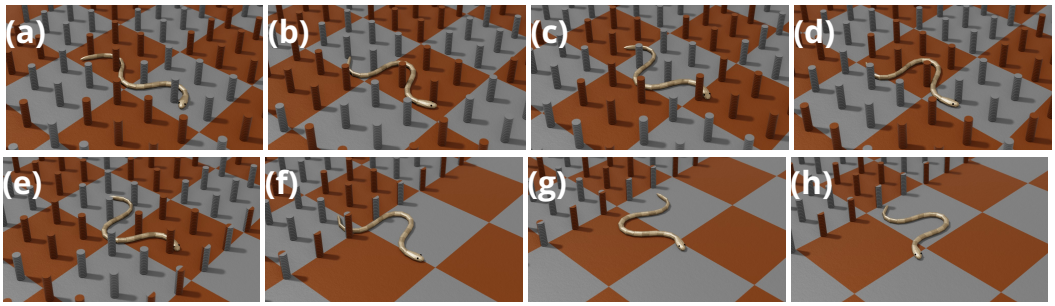


Figure 12. Snake obstacle-aided locomotion in arena 1 Sequence of rendered snapshots separated by 2.5 [s]. The pegs have a diameter of 4 [cm] and are separated by 21 [cm] from centre-to-centre.

Draft

Mouse

The section demonstrates the use case of FARMS in developing complex musculoskeletal models from animal data, employing the development of a whole-body musculoskeletal model of a mouse as outlined in ***Tata Ramalingasetty et al.*** (***2021***) as a case study. Developing a musculoskeletal model comprises modeling the skeletal segments (links and inertials), identifying the relationship between segments (joints) for articulation, and finally developing the body musculature (actuators). For the mouse model, 108 skeletal segments were identified, and clean mesh data set were created based on CT scans and sample zoological references. Individual bone mesh segments were then loaded in blender, and using the farms_blender plugin the relationship between the bones were identified and constrains between them were introduced as joints to have an articulated mouse skeletal system. In total, the mouse skeletal system consisted of 225 DoFs. Next, for each bone the inertial properties were automatically estimated based on the convex hulls that represent the net volume encapsulated by the skin, soft-tissues, muscles, and the bone with constant density of water (1000 kg/m^3). The mouse model was then exported from blender using farms_blender plugin to the simulation description format (SDF) supported by FARMS. Along with the skeletal system the mouse skin with texture was also made available. Currently, the mouse skin is only for visual purposes and does not influence the dynamics of the simulations.

Modeling the musculature as Hill-type muscles to actuate the skeletal model is a tedious job. It involves recreating muscle geometry spanning over the bones and estimating muscle parameters that dictate the dynamics of force generation in the muscles as in figure 13 panels 2 and 3. ***Tata Ramalingasetty et al.*** (***2021***) modeled the detailed musculature for actuating mouse hindlimbs and forelimbs. The study highlights the features of modeling muscles; (i) transferring available muscle attachments and parameters from one model to another (ii) creating muscle paths from CT scan data and (ii) estimating unknown muscle properties using numerical optimization techniques.

Detailed mouse hindlimb musculature was first developed by ***Charles et al.*** (***2016a***). The full skeletal model described in ***Tata Ramalingasetty et al.*** (***2021***) differed in the bone geometry from the Charles model due to differences in the mouse specimens used to develop the respective models. Mesh registration computes a coordinate transformation matrix from any point on one mesh to another. ***Tata Ramalingasetty et al.*** (***2021***) used CloudCompare, an open-source software to perform mesh registration on the hindlimbs of the two mouse models. For complex meshes it is possible and recommended to include reference landmarks on each of the meshes. This feature has now been implemented in farms_blender and allows users to use it directly within the blender environment. The affine transformation obtained from the mesh registration was then used to transfer the attachment points between the models. This automated pipeline eases the transfer process and limits the errors when working with different animal specimens. Panel 1 of figure 13 shows an example of using the mesh registration technique to register the femur bones between the mouse models.

For the forelimbs, there were no previously existing musculoskeletal models to begin with. In the study by ***DeLaurier et al.*** (***2008***), a three-dimensional atlas of the mouse embryo's forelimb at embryonic day 14.5 was created using Optical Projection Tomography and digital segmentation techniques. Using the mesh registration technique described earlier, we could identify insertion and origin points on the full model forelimb based on the mouse forelimb atlas (Figure 13 panel 4). farms_blender provides the necessary tools to easily add intermediate attachment points for muscles in 3D over the bones to reconstruct muscle geometry based on scans such as the mouse forelimb atlas.

Hill-type muscles are parameterized by the optimal fiber length (l_m^o; the length of muscle-fiber at which the muscle produces maximal active force), the tendon slack length (l_t^s; the length of the tendon below which the muscle transfers no force to the attached bones), the maximum muscle-fiber velocity (υ_m^{max}), the pennation angle when the fiber length is at its optimal (α_o), and the maximum isometric force (F_0^m).

Draft

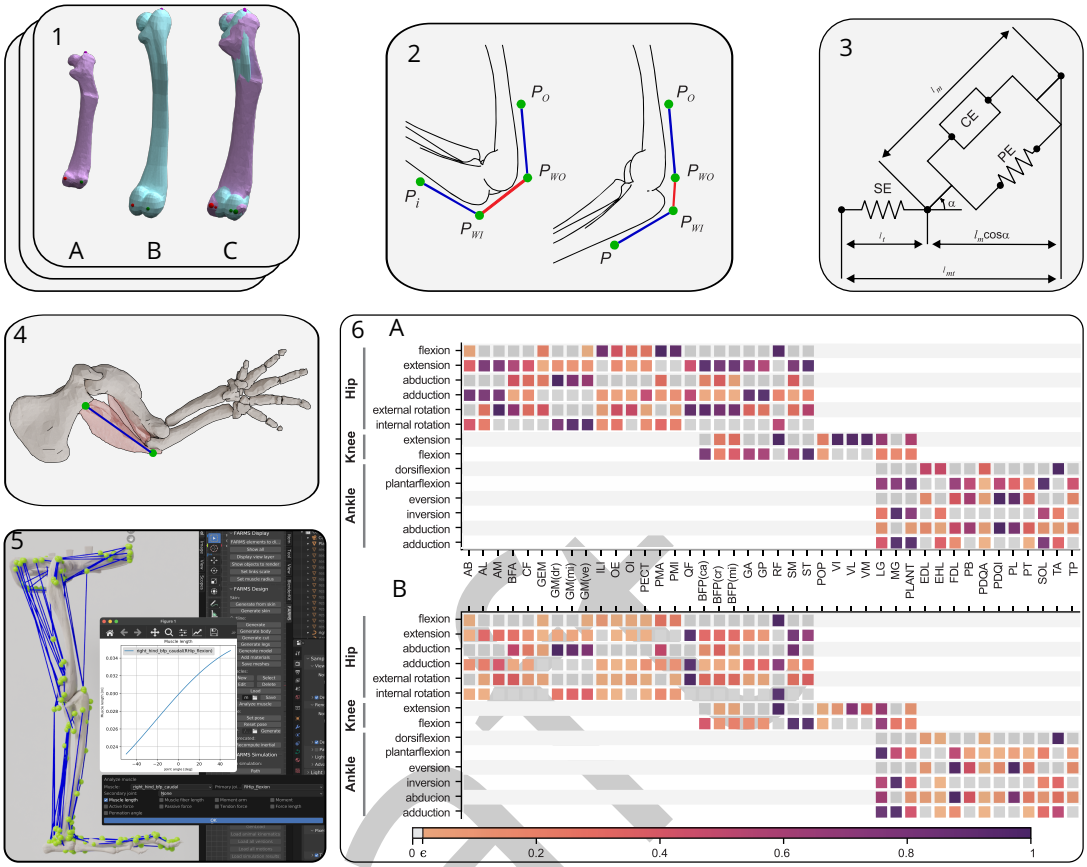


Figure 13. Development of mouse whole-body musucloskeletal model. **Panel 1**: Example of femur bone mesh registration process to transfer muscle attachment landmarks from femur model described by ***Charles et al.*** (***2016b***) (1A) to whole-body mouse model described in ***Tata Ramalingasetty et al.*** (***2021***) (1B). 1C validates the successful transfer process. **Panel 2**: Polyline approximation of muscle paths used in FARMS to describe muscle paths. The example shows the change in extensor muscle length spanning across the knee joint. (Figure reproduced from ***Tata Ramalingasetty et al.*** (***2021***)). **Panel 3**: Schematic representation of the Hill-type muscle model comprising active and passive force producing elements. **Panel 4**: Identification of muscle paths for forelimb muscles based on CT data from ***DeLaurier et al.*** (***2008***). **Panel 5**: FARMS plugin in blender to model and analyze musculoskeletal models. The example shows the hindlimb muscusloskeletal model loaded in blender using farms_blender and the feature to analyze generate plots for performing muscle analysis. **Panel 6**: Comprehensive overview of hindlimb muscle and joint relationship based on maximum muscle moment-arms (A) and moments (B). Grey boxes indicate joint functions for joints the muscle spans but has zero influence over. ϵ indicates a very low non-zero positive value.

For each hindlimb muscle α_o and F_0^m remained the same as reported in ***Charles et al.*** (***2016b***). Length dependent parameters l_m^o and l_t^s were scaled based on the algorithm described in ***Modenese et al.*** (***2016***). The algorithm has been re-implemented and is now currently available with farms_blender for use in other musculoskeletal model development. The forelimb musculoskeletal model was the first of its kind for a mouse. Different strategies had to be combined to identify forelimb muscle parameters. Parameters l_m^o, F_0^m and α_o were extracted and scaled accordingly to the full model from measurements made by ***Mathewson et al.*** (***2012***). An adaptation of the numerical optimization technique formulated by ***Manal and Buchanan*** (***2004***) was implemented in farms_blender to estimate l_t^s. The extended algorithm now considers all the DoFs a muscle operates on and removed the restriction that muscle fibers only operate in the ascending region of the force-length curve while optimizing for l_t^s.

The fully parameterized whole-body mouse musculoskeletal model was then used to preform a comprehensive analysis to understand the relationship between muscles and joint movements.

Figure 13 panel 6 shows the action of every muscle in the mouse hindlimb on all the joints it spans over in terms of normalized muscle moment-arm and moments. For a more detailed discussion on the analysis refer to ***Tata Ramalingasetty et al.*** (***2021***).

We adapted the mouse musculoskeletal model developed by ***Tata Ramalingasetty et al.*** (***2021***) to simulate hindlimb locomotion with a reduced set of muscles (***Tata Ramalingasetty et al., 2023***). Here, we reduced the model complexity both at the level of the skeletal system and the musculature. Tools provided by the FARMS framework facilitated the process of model simplification. The modeled skeletal system had 18 degrees-of-freedom (DoFs) 6 of which were actively controlled by Hill-type muscles and other 12 represented passive DoFs modeled as spring-damper systems. Each hindlimb had 3 active DoFs (hip, knee and ankle) in the sagittal plane. Every muscle in the hindlimb represented a group of synergistic muscles. Overall, each hindlimb had 8 muscles with 6 mono-articular and 2 bi-articular muscles (Figure 14).

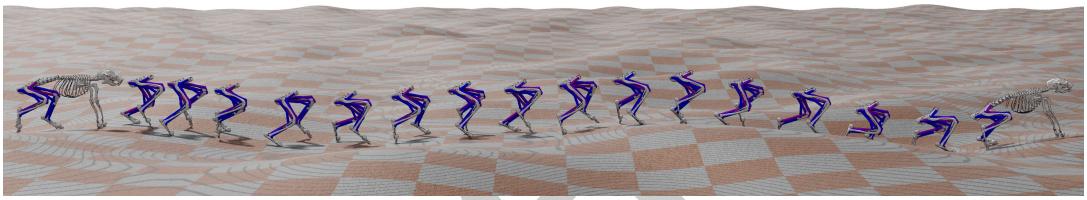


Figure 14. A sequence gait snapshots over time of hindlimb mouse locomotion on uneven terrain controlled by a sensory driven spinal neural model similar to one described in ***Markin et al.*** (***2016***)

The spinal circuitry for locomotion was modeled as a two layered neuronal network initially proposed by ***McCrea and Rybak*** (***2008***) and later implemented in a neuromechanical simulation by ***Markin et al.*** (***2016***). It included rhythm generators, each controlling one limb, that defined the locomotor frequency and flexor-extensor alternation. The rhythm generators projected their activity to the pattern formation circuits which generated muscle synergies and created muscle-specific activation patterns, which was used to drive muscles of the musculoskeletal model. Each population described as a non-spiking activity-based neuron model with persistent (slowly inactivating) sodium current (INaP) for excitatory rhythm and pattern centers.

Proprioceptive (spindle Ia and II, and Golgi tendon Ib) and exteroceptive feedback signals provided sensory feedback to the spinal circuit model. Spindle group II stretch feedback from tibialis anterior and iliopsoas and Golgi tendon Ib feedbacks from medial gastrocnemius and soleus were connected to the rhythm generators (affecting timing of phase-transitions). Basic Ia and Ib reflexes were organized at the motoneuron level. Proprioceptive feedback signals were modeled as described by ***Markin et al.*** (***2010***) and are currently available as part of farms_muscle package. farms_blender along with farms_muscle extends support to simulate complex Hill-type based musculoskeletal models in both Bullet and MuJoCo physics simulators.

Conclusion

In this work, we have introduced a novel framework, FARMS, built for developing and performing neuromechanical simulations of animals and robots to study movement control. Its modular design allows independent use of its components depending on individual research requirements. FARMS is driven to maximize the utility of existing open-source projects, thereby harnessing their strengths and contributing back to existing infrastructure. We have discussed the different components of the framework in-depth and presented a suite of demonstrations to provide an overview of the features and capabilities. This included presenting how to design and generate models of animals and robots along with neural network models to control them in neuromechanical simulations spanning across diverse modalities such as crawling, walking, and swimming.

In the rich landscape of simulation packages, the position and utility of FARMS becomes clear when compared to the capabilities of other existing frameworks. Physics engines such as ODE, Bul-

Draft

let, and MuJoCo offer exceptional performance and customization. When used directly, they serve as the ideal choice for applications that require specific features and maximum performance, albeit at the expense of integrated modeling and post-processing analysis tools, which FARMS readily provides. Webots, with its consolidated interface, remains an attractive choice for introductory learning and the construction of robot models. Nevertheless, when it comes to handling more complex models, interfacing with learning and evolutionary algorithms, or having access to neural and muscle models, FARMS may be a preferable alternative. OpenSim, a well-known tool in the field, excels in conducting musculoskeletal human experiments and facilitates locomotion reverse-engineering through motion capture. It provides detailed muscle models and an interface, which make it attractive when existing models are already available. However, the creation of new models proves more challenging in OpenSim, and is more limited for forward simulations in terms of performance and ease of use. Isaac gym, on the other hand, stands out for its reinforcement learning capabilities with robots, as it supports running multiple robot simulations in parallel thanks to its Graphics Processing Unit (GPU) support. However, it currently lacks comprehensive support for muscle and neural models as well as complex environments, all of which are central features of FARMS. To summarize, FARMS provides an attractive balance of performance, versatility, and user-friendly tools. While each framework has its strengths and ideal use cases, we believe that FARMS shines in its ability to support complex model creation, integrate with a variety of algorithms, and facilitate a deeper understanding of both animal locomotion and bio-inspired robotic systems. However, for scenarios that prioritize the highest possible performance, simpler models, human musculoskeletal experiments, or specific reinforcement learning tasks, other engines may be more suitable.

In the pursuit of the continual enhancement of FARMS, we outline several avenues of future work. First, we aim to enrich the diversity of models by incorporating a wider variety of animals, robots, and arenas. This will further elevate the comprehensive nature of the framework and its applicability to a broader range of studies. We are also committed to refining the biomechanical and musculoskeletal models in the framework, to foster a more accurate and detailed representation of biological systems. This will be complemented by the integration of spiking neural models, which will allow for a more faithful portrayal of the complexity of neural information processing. Recognizing the importance of sensory inputs in both animal locomotion and robotic control, we plan to expand support for various sensors such as cameras and Inertial Measurement Unit (IMUs). We also intend to integrate SPH, validating this feature to ensure reliable and accurate fluid simulation. In line with this, we aim to develop particle simulations for complex terrains, such as sand, to better model and understand locomotion in these environments. Additionally, we also consider supporting soft skins and bodies, reflecting the compliance and versatility seen in many biological organisms. Concerning new file formats, we are considering to transition from the current default SDF file format to the more versatile Universal Scene Description (USD) format by default for better integration with other modeling tools, while still providing support for SDF, URDF and MJCF within FARMS. To establish the fidelity of our simulations, we intend to undertake rigorous sim-to-real validations. Moreover, we will support reinforcement learning algorithms to allow for the exploration of efficient and adaptive control strategies in both biological and robotic systems. We also aim to provide support for workflow managers such as Snakemake (***Mölder et al., 2021***) or SciLuigi (***Lampa et al., 2016***), which will facilitate the reproducibility and scalability of studies conducted using FARMS. Finally, with a goal of democratizing access to this interdisciplinary research, we plan to develop a user-friendly interface that further lowers the barrier to entry for researchers interested in leveraging FARMS for their work.

To conclude, our aim is for FARMS to serve as a platform for advancing the study of animal and robotics systems' motion. It provides an array of tools, designed to facilitate the creation of models with realistic dynamics, the development of traditional or neural controllers, and comparative studies between animals and/or robotic systems. FARMS further accommodates optimization methodologies for refined parameter tuning, and 3D visualizations enhancing the debugging pro-

Draft

cess and enabling the production of high-quality videos suitable for scholarly publications. Our aspiration is to cultivate FARMS as an open-source, community-driven project, continually evolving to meet the dynamic demands of the scientific community. Thus it is our hope that FARMS could help bridge between different fields of research, including biomechanics, neuroscience, and robotics.

Acknowledgments

This research was supported in part by the Human Frontier Science Program (HFSP-RGP0027/2017); in part by the European Research Council (ERC) under the European Union's Horizon 2020 research (Synergia project, Salamandra); in part by the innovation programme (grant agreement No 951477); European Union's Horizon 2020 Research and Innovation Program under Agreement 720270917 (SGA1) and Agreement 785907 (SGA2); and in part by the National Institutes of Health (NIH) under Grant R01NS112304, Grant R01NS115900.

We would like to thank members from Biorobotics Laboratory (BioRob) - EPFL, Neuroengineering Laboratory - EPFL, Danner Lab - Drexel University, Neurobiology and Anatomy, for valuable feedback on the simulator development.

References

Angelidis E, Bender J, Arreguit J, Gleim L, Wang W, Axenie C, Knoll A, Ijspeert A. Gazebo Fluids: SPH-based Simulation of Fluid Interaction with Articulated Rigid Body Dynamics. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; 2022. p. 11238–11245. doi: 10.1109/IROS47612.2022.9982036.

Auerbach J, Aydin D, Maesani A, Kornatowski P, Cieslewski T, Heitz G, Fernando P, Loshchilov I, Daler L, Floreano D. RoboGen: Robot Generation through Artificial Evolution. In: *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems* MIT Press; 2014. p. 136–137. https://direct.mit.edu/isal/article/doi/10.1162/978-0-262-32621-6-ch022/99059/RoboGen-Robot-Generation-through-Artificial, doi: 10.1162/978-0-262-32621-6-ch022.

Bender J, SPlisHSPlasH Library; 2017. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.

Blank J, Deb K. Pymoo: Multi-Objective Optimization in Python. IEEE Access. 2020; 8:89497–89509. doi: 10.1109/ACCESS.2020.2990567.

Charles JP, Cappellari O, Spence AJ, Hutchinson JR, Wells DJ. Musculoskeletal Geometry, Muscle Architecture and Functional Specialisations of the Mouse Hindlimb. PLOS ONE. 2016 Apr; 11(4):e0147669. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0147669, doi: 10.1371/journal.pone.0147669.

Charles JP, Cappellari O, Spence AJ, Wells DJ, Hutchinson JR. Muscle Moment Arms and Sensitivity Analysis of a Mouse Hindlimb Musculoskeletal Model. Journal of Anatomy. 2016; 229(4):514–535. https://onlinelibrary.wiley.com/doi/abs/10.1111/joa.12461, doi: 10.1111/joa.12461.

Collins J, Chand S, Vanderkop A, Howard D. A Review of Physics Simulators for Robotic Applications. IEEE Access. 2021; 9:51416–51431. doi: 10.1109/ACCESS.2021.3068769.

Community BO, Blender - a 3D Modelling and Rendering Package. Stichting Blender Foundation, Amsterdam; 2022. https://www.blender.org/. Blender Foundation.

Coumans E, Bai Y, Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning; 2016. https://pybullet.org/wordpress/.

Crespi A, Badertscher A, Guignard A, Ijspeert AJ. AmphiBot I: An Amphibious Snake-like Robot. Robotics and Autonomous Systems. 2005 Mar; 50(4):163–175. https://www.sciencedirect.com/science/article/pii/S0921889004001721, doi: 10.1016/j.robot.2004.09.015.

Crespi A, Ijspeert AJ. AmphiBot II: An Amphibious Snake Robot That Crawls and Swims Using a Central Pattern Generator. In: *Proceedings of the 9th International Conference on Climbing and Walking Robots* Brussels, Belgium; 2006. p. 19–27. https://infoscience.epfl.ch/record/142759?ln=en.

Crespi A, Karakasiliotis K, Guignard A, Ijspeert AJ. Salamandra Robotica II: An Amphibious Robot to Study Salamander-Like Swimming and Walking Gaits. IEEE Transactions on Robotics. 2013 Apr; 29(2):308–320. doi: 10.1109/TRO.2012.2234311.

Draft

Danner SM, Shevtsova NA, Frigon A, Rybak IA. Computational Modeling of Spinal Circuits Controlling Limb Coordination and Gaits in Quadrupeds. eLife. 2017 Nov; 6:e31050. https://doi.org/10.7554/eLife.31050, doi: 10.7554/eLife.31050.

Daun-Gruhn S. A Mathematical Modeling Study of Inter-Segmental Coordination during Stick Insect Walking. Journal of Computational Neuroscience. 2011 Apr; 30(2):255–278. https://doi.org/10.1007/s10827-010-0254-3, doi: 10.1007/s10827-010-0254-3.

De Groote F, Kinney AL, Rao AV, Fregly BJ. Evaluation of Direct Collocation Optimal Control Problem Formulations for Solving the Muscle Redundancy Problem. Annals of Biomedical Engineering. 2016 Oct; 44(10):2922–2936. https://doi.org/10.1007/s10439-016-1591-9, doi: 10.1007/s10439-016-1591-9.

DeLaurier A, Burton N, Bennett M, Baldock R, Davidson D, Mohun TJ, Logan MP. The Mouse Limb Anatomy Atlas: An Interactive 3D Tool for Studying Embryonic Limb Patterning. BMC Developmental Biology. 2008 Sep; 8(1):83. https://doi.org/10.1186/1471-213X-8-83, doi: 10.1186/1471-213X-8-83.

Dickinson MH. How Animals Move: An Integrative View. Science. 2000 Apr; 288(5463):100–106. https://www.sciencemag.org/lookup/doi/10.1126/science.288.5463.100, doi: 10.1126/science.288.5463.100.

Dzeladini F, van den Kieboom J, Ijspeert A. The Contribution of a Central Pattern Generator in a Reflex-Based Neuromuscular Model. Frontiers in Human Neuroscience. 2014; 8. https://www.frontiersin.org/articles/10.3389/fnhum.2014.00371.

Edwards DH. Neuromechanical Simulation. Frontiers in Behavioral Neuroscience. 2010 Jul; 4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2914529/, doi: 10.3389/fnbeh.2010.00040.

Ekeberg Ö. A Combined Neuronal and Mechanical Model of Fish Swimming. Biological Cybernetics. 1993 Sep; 69(5):363–374. https://doi.org/10.1007/BF01185408, doi: 10.1007/BF01185408.

Ekeberg Ö, Pearson K. Computer Simulation of Stepping in the Hind Legs of the Cat: An Examination of Mechanisms Regulating the Stance-to-Swing Transition. Journal of Neurophysiology. 2005 Dec; 94(6):4256–4268. https://www.physiology.org/doi/abs/10.1152/jn.00065.2005, doi: 10.1152/jn.00065.2005.

Ermentrout B. Reduction of Conductance-Based Models with Slow Synapses to Neural Nets. Neural Computation. 1994 Jul; 6(4):679–695. doi: 10.1162/neco.1994.6.4.679.

FitzHugh R. Impulses and Physiological States in Theoretical Models of Nerve Membrane. Biophysical Journal. 1961 Jul; 1(6):445–466. https://www.sciencedirect.com/science/article/pii/S0006349561869026, doi: 10.1016/S0006-3495(61)86902-6.

Geijtenbeek T. Animating Virtual Characters Using Physics-Based Simulation. Utrecht University; 2013.

Geijtenbeek T. SCONE: Open Source Software for Predictive Simulation of Biological Motion. Journal of Open Source Software. 2019 Jun; 4(38):1421. http://joss.theoj.org/papers/10.21105/joss.01421, doi: 10.21105/joss.01421.

Gewaltig MO, Diesmann M. NEST (NEural Simulation Tool). Scholarpedia. 2007 Apr; 2(4):1430. http://www.scholarpedia.org/article/NEST_(NEural_Simulation_Tool), doi: 10.4249/scholarpedia.1430.

Geyer H, Herr H. A Muscle-Reflex Model That Encodes Principles of Legged Mechanics Produces Human Walking Dynamics and Muscle Activities. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2010 Jun; 18(3):263–273. doi: 10.1109/TNSRE.2010.2047592.

Guo S, Lin J, Wöhrl T, Liao M. A Neuro-Musculo-Skeletal Model for Insects With Data-driven Optimization. Scientific Reports. 2018 Feb; 8(1):2129. https://www.nature.com/articles/s41598-018-20093-x, doi: 10.1038/s41598-018-20093-x.

Hamlet CL, Hoffman KA, Tytell ED, Fauci LJ. The Role of Curvature Feedback in the Energetics and Dynamics of Lamprey Swimming: A Closed-Loop Model. PLOS Computational Biology. 2018 Aug; 14(8):e1006324. https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006324, doi: 10.1371/journal.pcbi.1006324.

Harischandra N, Cabelguen JM, Ekeberg Ö. A 3D Musculo-Mechanical Model of the Salamander for the Study of Different Gaits and Modes of Locomotion. Frontiers in Neurorobotics. 2010; 4. https://www.frontiersin.org/articles/10.3389/fnbot.2010.00112.

Draft

Hines ML, Carnevale NT. The NEURON Simulation Environment. Neural Computation. 1997 Aug; 9(6):1179–1209. https://doi.org/10.1162/neco.1997.9.6.1179, doi: 10.1162/neco.1997.9.6.1179.

Hodgkin AL, Huxley AF. Currents Carried by Sodium and Potassium Ions through the Membrane of the Giant Axon of Loligo. The Journal of Physiology. 1952 Apr; 116(4):449–472. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392213/.

Holmes P, Full RJ, Koditschek D, Guckenheimer J. The Dynamics of Legged Locomotion: Models, Analyses, and Challenges. SIAM Review. 2006 Jan; 48(2):207–304. http://epubs.siam.org/doi/10.1137/S0036144504445133, doi: 10.1137/S0036144504445133.

Ijspeert AJ. Biorobotics: Using Robots to Emulate and Investigate Agile Locomotion. Science. 2014 Oct; 346(6206):196–203. https://www.science.org/doi/full/10.1126/science.1254486, doi: 10.1126/science.1254486.

Ijspeert AJ. A Connectionist Central Pattern Generator for the Aquatic and Terrestrial Gaits of a Simulated Salamander. Biological Cybernetics. 2001 Apr; 84(5):331–348. https://doi.org/10.1007/s004220000211, doi: 10.1007/s004220000211.

Ijspeert AJ. Central Pattern Generators for Locomotion Control in Animals and Robots: A Review. Neural Networks. 2008 May; 21(4):642–653. http://www.sciencedirect.com/science/article/pii/S0893608008000804, doi: 10.1016/j.neunet.2008.03.014.

Ijspeert AJ, Crespi A, Ryczko D, Cabelguen JM. From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model. Science. 2007 Mar; 315(5817):1416–1420. http://science.sciencemag.org/content/315/5817/1416, doi: 10.1126/science.1138353.

Ijspeert AJ, Kodjabachian J. Evolution and Development of a Central Pattern Generator for the Swimming of a Lamprey. Artificial Life. 1999 Jul; 5(3):247–269. https://doi.org/10.1162/106454699568773, doi: 10.1162/106454699568773.

Karakasiliotis K, Thandiackal R, Melo K, Horvat T, Mahabadi NK, Tsitkov S, Cabelguen JM, Ijspeert AJ. From Cineradiography to Biorobots: An Approach for Designing Robots to Emulate and Study Animal Locomotion. Journal of The Royal Society Interface. 2016 Jun; 13(119):20151089. https://royalsocietypublishing.org/doi/full/10.1098/rsif.2015.1089, doi: 10.1098/rsif.2015.1089.

Koenig N, Howard A. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3; 2004. p. 2149–2154 vol.3. doi: 10.1109/IROS.2004.1389727.

Kuramoto Y. Self-Entrainment of a Population of Coupled Non-Linear Oscillators. In: Araki H, editor. *International Symposium on Mathematical Problems in Theoretical Physics* Lecture Notes in Physics, Berlin, Heidelberg: Springer; 1975. p. 420–422. doi: 10.1007/BFb0013365.

Lampa S, Alvarsson J, Spjuth O. Towards Agile Large-Scale Predictive Modelling in Drug Discovery with Flow-Based Programming Design Principles. Journal of Cheminformatics. 2016 Nov; 8(1):67. https://doi.org/10.1186/s13321-016-0179-6, doi: 10.1186/s13321-016-0179-6.

Liang J, Makoviychuk V, Handa A, Chentanez N, Macklin M, Fox D. GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning. In: *Proceedings of The 2nd Conference on Robot Learning* PMLR; 2018. p. 270–282. https://proceedings.mlr.press/v87/liang18a.html.

Liu Q, Yang H, Zhang J, Wang J. A New Model of the Spinal Locomotor Networks of a Salamander and Its Properties. Biological Cybernetics. 2018 Aug; 112(4):369–385. https://doi.org/10.1007/s00422-018-0759-9, doi: 10.1007/s00422-018-0759-9.

Liu Q, Zhang Y, Wang J, Yang H, Hong L. Modeling of the Neural Mechanism Underlying the Terrestrial Turning of the Salamander. Biological Cybernetics. 2020 Jun; 114(3):317–336. https://doi.org/10.1007/s00422-020-00821-1, doi: 10.1007/s00422-020-00821-1.

Lobato-Rios V, Ramalingasetty ST, Özdil PG, Arreguit J, Ijspeert AJ, Ramdya P. NeuroMechFly, a Neuromechanical Model of Adult Drosophila Melanogaster. Nature Methods. 2022 May; 19(5):620–627. https://www.nature.com/articles/s41592-022-01466-7, doi: 10.1038/s41592-022-01466-7.

Manal K, Buchanan TS. Subject-Specific Estimates of Tendon Slack Length: A Numerical Method. Journal of Applied Biomechanics. 2004; 20(2):195–203. doi: 10.1123/jab.20.2.195.

Draft

Markin SN, Klishko AN, Shevtsova NA, Lemay MA, Prilutsky BI, Rybak IA. Afferent Control of Locomotor CPG: Insights from a Simple Neuromechanical Model: Afferent Control of Locomotor CPG. Annals of the New York Academy of Sciences. 2010 Jun; 1198(1):21–34. http://doi.wiley.com/10.1111/j.1749-6632.2010.05435.x, doi: 10.1111/j.1749-6632.2010.05435.x.

Markin SN, Klishko AN, Shevtsova NA, Lemay MA, Prilutsky BI, Rybak IA. A Neuromechanical Model of Spinal Control of Locomotion. In: Prilutsky BI, Edwards DH, editors. *Neuromechanical Modeling of Posture and Locomotion* Springer Series in Computational Neuroscience, New York, NY: Springer; 2016.p. 21–65. https://doi.org/10.1007/978-1-4939-3267-2_2, doi: 10.1007/978-1-4939-3267-2_2.

Mathewson MA, Chapman MA, Hentzen ER, Fridén J, Lieber RL. Anatomical, Architectural, and Biochemical Diversity of the Murine Forelimb Muscles. Journal of Anatomy. 2012 Nov; 221(5):443–451. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3482352/, doi: 10.1111/j.1469-7580.2012.01559.x.

Matsuoka K. Sustained Oscillations Generated by Mutually Inhibiting Neurons with Adaptation. Biological Cybernetics. 1985 Oct; 52(6):367–376. https://doi.org/10.1007/BF00449593, doi: 10.1007/BF00449593.

McCrea DA, Rybak IA. Organization of Mammalian Locomotor Rhythm and Pattern Generation. Brain Research Reviews. 2008 Jan; 57(1):134–146. https://linkinghub.elsevier.com/retrieve/pii/S0165017307001798, doi: 10.1016/j.brainresrev.2007.08.006.

Melo K, Horvat T, Ijspeert A. K-Rock, a Bio-robot Outside the Lab, Back In Nature. The 8th International Symposium on Adaptive Motion of Animals and Machines (AMAM2017). 2017; https://adaptivemotion.org/AMAM2017/files/proceedings/P19.pdf.

Michel O. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems. 2004 Mar; 1(1):5. https://doi.org/10.5772/5618, doi: 10.5772/5618.

Millard M, Uchida T, Seth A, Delp SL. Flexing Computational Muscle: Modeling and Simulation of Musculotendon Dynamics. Journal of Biomechanical Engineering. 2013 Feb; 135(2):0210051–02100511. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3705831/, doi: 10.1115/1.4023390.

Modenese L, Ceseracciu E, Reggiani M, Lloyd DG. Estimation of Musculotendon Parameters for Scaled and Subject Specific Musculoskeletal Models Using an Optimization Technique. Journal of Biomechanics. 2016 Jan; 49(2):141–148. http://www.sciencedirect.com/science/article/pii/S0021929015006363, doi: 10.1016/j.jbiomech.2015.11.006.

Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, Forster J, Lee S, Twardziok SO, Kanitz A, Wilm A, Holtgrewe M, Rahmann S, Nahnsen S, Köster J, Sustainable Data Analysis with Snakemake; 2021. https://f1000research.com/articles/10-33, doi: 10.12688/f1000research.29032.2.

Nyakatura JA, Melo K, Horvat T, Karakasiliotis K, Allen VR, Andikfar A, Andrada E, Arnold P, Lauströer J, Hutchinson JR, Fischer MS, Ijspeert AJ. Reverse-Engineering the Locomotion of a Stem Amniote. Nature. 2019 Jan; 565(7739):351–355. https://www.nature.com/articles/s41586-018-0851-2, doi: 10.1038/s41586-018-0851-2.

Puri K, Ramachandran P, Pandey P, Kaushik C, Godbole P. PySPH: A Python Framework for SPH. In: *8th International SPHERIC Workshop* Trondheim, Norway; 2013. .

Ramachandran P, Bhosale A, Puri K, Negi P, Muta A, Dinesh A, Menon D, Govind R, Sanka S, Sebastian AS, Sen A, Kaushik R, Kumar A, Kurapati V, Patil M, Tavker D, Pandey P, Kaushik C, Dutt A, Agarwal A. PySPH: A Python-based Framework for Smoothed Particle Hydrodynamics. ACM Transactions on Mathematical Software. 2021 Dec; 47(4):1–38. http://arxiv.org/abs/1909.04504, doi: 10.1145/3460773.

Righetti L, Buchli J, Ijspeert AJ. Dynamic Hebbian Learning in Adaptive Frequency Oscillators. Physica D: Nonlinear Phenomena. 2006 Apr; 216(2):269–281. https://www.sciencedirect.com/science/article/pii/S0167278906000819, doi: 10.1016/j.physd.2006.02.009.

Seth A, Sherman M, Reinbolt JA, Delp SL. OpenSim: A Musculoskeletal Modeling and Simulation Framework for in Silico Investigations and Exchange. Procedia IUTAM. 2011; 2:212–232. https://linkinghub.elsevier.com/retrieve/pii/S2210983811000228, doi: 10.1016/j.piutam.2011.04.021.

Sims K. Evolving 3D Morphology and Behavior by Competition. Artificial Life. 1994 Jul; 1(4):353–372. doi: 10.1162/artl.1994.1.4.353.

Sims K. Evolving Virtual Creatures. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* SIGGRAPH '94, New York, NY, USA: Association for Computing Machinery; 1994. p. 15–22. https://doi.org/10.1145/192161.192167, doi: 10.1145/192161.192167.

Draft

Stimberg M, Brette R, Goodman DF. Brian 2, an Intuitive and Efficient Neural Simulator. eLife. 2019 Aug; 8:e47314. https://doi.org/10.7554/eLife.47314, doi: 10.7554/eLife.47314.

Suzuki S, Kano T, Ijspeert AJ, Ishiguro A. Spontaneous Gait Transitions of Sprawling Quadruped Locomotion by Sensory-Driven Body–Limb Coordination Mechanisms. Frontiers in Neurorobotics. 2021; 15. https://www.frontiersin.org/articles/10.3389/fnbot.2021.645731.

Tata Ramalingasetty S, Danner SM, Arreguit J, Markin SN, Rodarie D, Kathe C, Courtine G, Rybak IA, Ijspeert AJ. A Whole-Body Musculoskeletal Model of the Mouse. IEEE Access. 2021; 9:163861–163881. doi: 10.1109/ACCESS.2021.3133078.

Tata Ramalingasetty S, Markin SN, Lockhart AB, Arreguit J, Shevtsova NA, Ijspeert AJ, Rybak IA, Danner SM, An Integrated Neuromechanical Model of the Mouse to Study Neural Control of Locomotion. Adaptive Motion of Animals and Machines Organizing Committee; 2023. https://doi.org/10.18910/92326, doi: 10.18910/92326.

Thandiackal R, Melo K, Paez L, Herault J, Kano T, Akiyama K, Boyer F, Ryczko D, Ishiguro A, Ijspeert AJ. Emergence of Robust Self-Organized Undulatory Swimming Based on Local Hydrodynamic Force Sensing. Science Robotics. 2021 Aug; 6(57). https://robotics.sciencemag.org/content/6/57/eabf6354, doi: 10.1126/scirobotics.abf6354.

The HDF Group, Hierarchical Data Format, Version 5; 1997.

Todorov E, Erez T, Tassa Y. MuJoCo: A Physics Engine for Model-Based Control. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*; 2012. p. 5026–5033. doi: 10.1109/IROS.2012.6386109.

Draft

Appendix

Robots

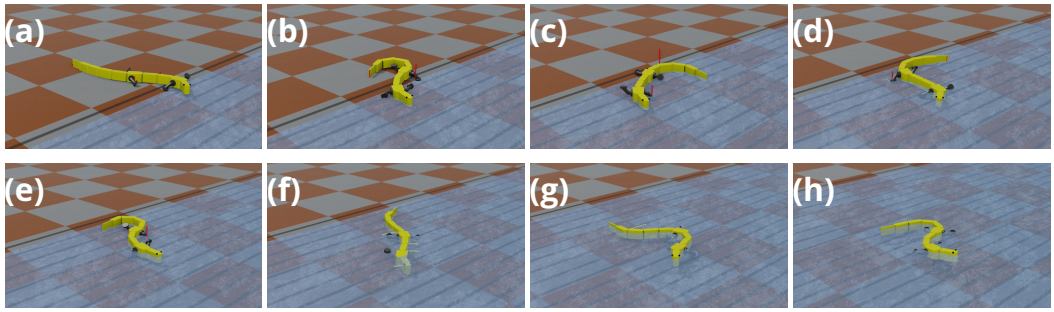
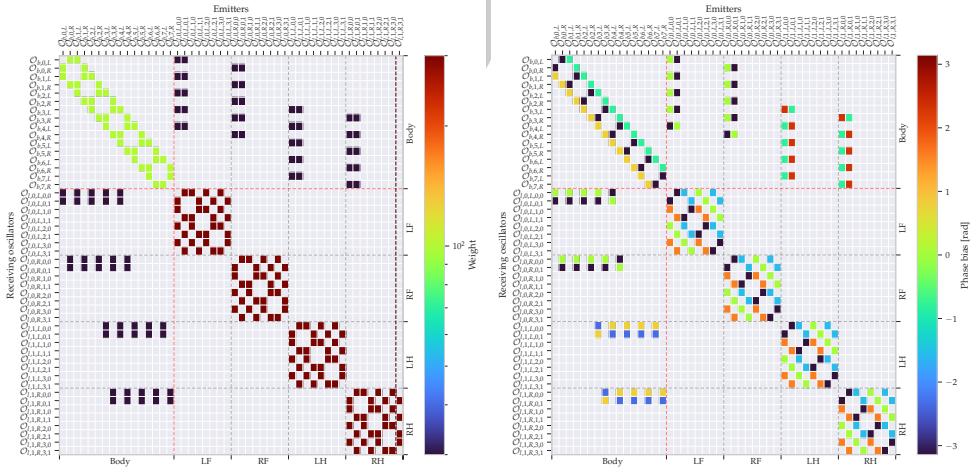


Figure 15. Salamandra Robotica II performing a walk-to-swim transition Sequence of rendered snapshots separated by 1.5 [s]. The robot is walking from right to left in 15a and comes into contact with the water. Throughout 15b to 15f, the robot keeps walking even though it is in contact with the water as it is still in contact with the ramp and the contact force threshold between the environments and the body has not been surpassed. Between 15f and 15g, the body is deep enough into the water to lift the body and the robot transitions from walking to swimming. The limbs saturate and move back and the trunk switches from a standing wave to a travelling wave, allowing it to propel itself forward in the water.

Salamander



(a) Connectivity weights

(b) Connectivity phase bias

Figure 16. Details for the salamander connectivity for an oscillator network. The coupling weights between pairs of oscillators are given in ***Figure 16a*** and the phase bias are provided in ***Figure 16a***.

Snake

Draft

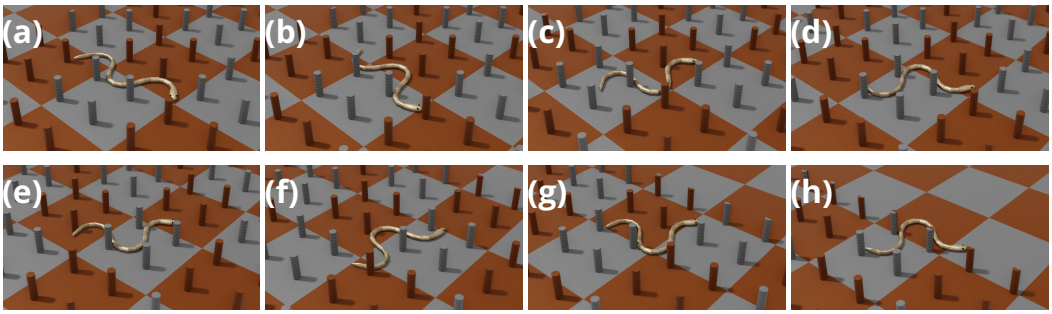


Figure 17. Snake obstacle-aided locomotion in arena 2 Sequence of rendered snapshots separated by 2.5 [s]. The pegs have a diameter of 4 [cm] and are separated by 27 [cm] from centre-to-centre.

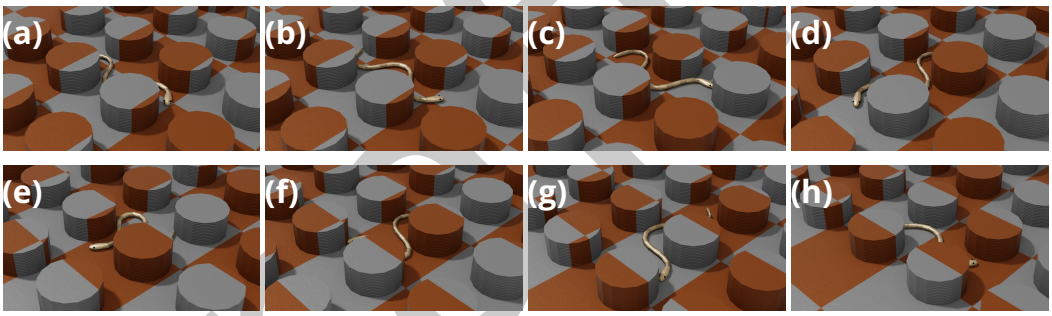


Figure 18. Snake obstacle-aided locomotion in arena 3 Sequence of rendered snapshots separated by 2.5 [s]. The pegs have a diameter of 30 [cm] and are separated by 45 [cm] from centre-to-centre.

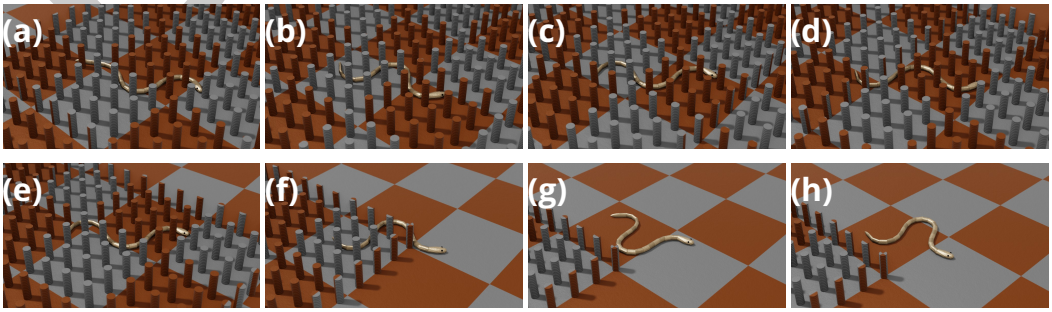


Figure 19. Snake obstacle-aided locomotion in arena 4 Sequence of rendered snapshots separated by 2.5 [s]. The pegs have a diameter of 4 [cm] and are separated by 14 [cm] from centre-to-centre.

Draft