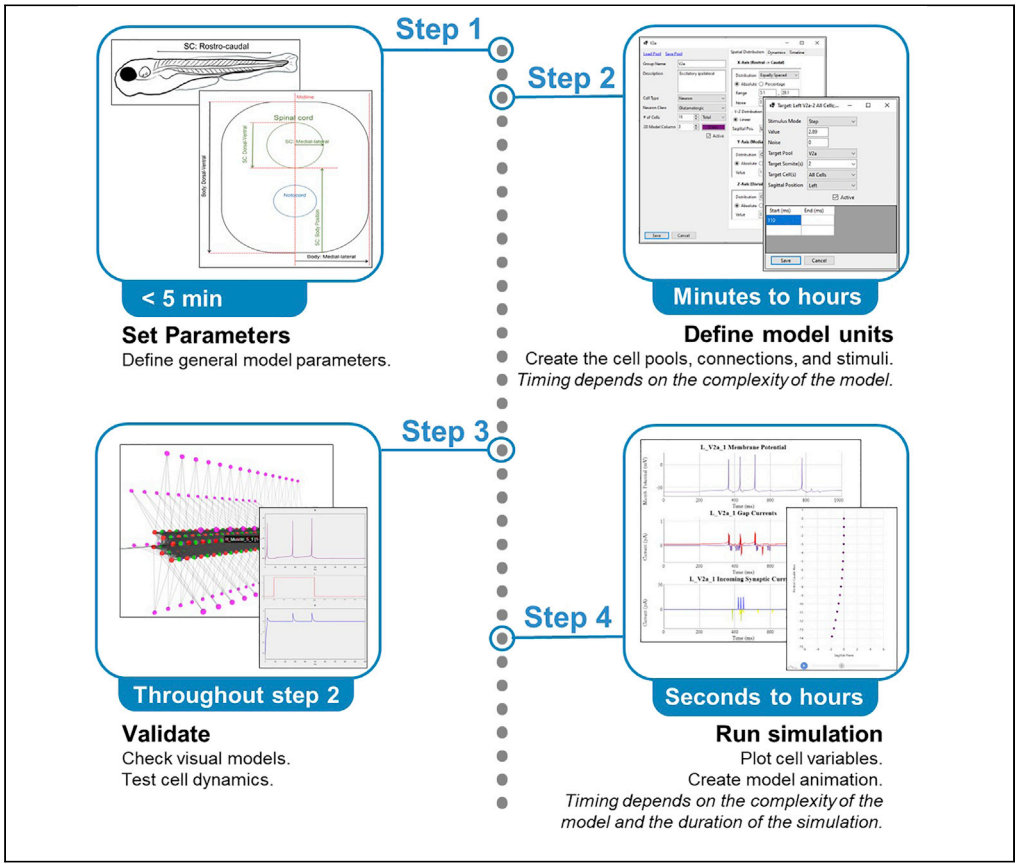
STAR Protocols

CellPress
OPEN ACCESS

Protocol

*Sili*Fish: A desktop application to model swimming behavior in developing zebrafish (*Danio rerio*)



*Sili*Fish is an open-source desktop application to model and study zebrafish swimming. Here, we explain how to define the general parameters of the model, define cell populations, place them within the spinal cord, and define their projections. We explain how to run a simulation and how to visualize the network output and single-cell activity. The choice of C# as the programming language allows higher speed performance, simulating models with larger spinal circuits in less time.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Emine Topcu, Yann Roussel, Tuan V. Bui

etopc053@uottawa.ca (E.T.)
tuan.bui@uottawa.ca (T.V.B.)

Highlights

Easy-to-use graphical user interface for constructing spinal circuit neural models

2D and 3D views of models to visualize cell placements and projections

Interactive plots of membrane potential, currents, and swimming output statistics

Topcu et al., STAR Protocols 4, 101973
March 17, 2023
https://doi.org/10.1016/j.xpro.2022.101973

STAR Protocols

CellPress
OPEN ACCESS

Protocol

*Sili*Fish: A desktop application to model swimming behavior in developing zebrafish (*Danio rerio*)

Emine Topcu,[1,3,*] Yann Roussel,[2] and Tuan V. Bui[1,4,*]

[1]Department of Biology, Center for Neural Dynamics, Brain and Mind Research Institute, University of Ottawa, Ottawa, ON K1N 6N5, Canada

[2]École Polytechnique Fédérale de Lausanne, Blue Brain Project, 1202 Geneva, Switzerland

[3]Technical contact

[4]Lead contact

*Correspondence: etopc053@uottawa.ca (E.T.), tuan.bui@uottawa.ca (T.V.B.)
https://doi.org/10.1016/j.xpro.2022.101973

SUMMARY

***Sili*Fish is an open-source desktop application to model and study zebrafish swimming. Here, we explain how to define the general parameters of the model, define cell populations, place them within the spinal cord, and define their projections. We explain how to run a simulation and how to visualize the network output and single-cell activity. The choice of C# as the programming language allows higher speed performance, simulating models with larger spinal circuits in less time.**

For complete details on the use and execution of this protocol, please refer to Roussel et al. (2021).[1]

BEFORE YOU BEGIN

Zebrafish undergo relatively rapid development of their motor control. Merely hours after fertilization, zebrafish embryos under a millimeter in length will show coiling movements consisting of large amplitude body bends. After hatching at about two days post-fertilization, larval zebrafish (now millimeters in length) display swimming patterns such as beat-and-glide swimming consisting of episodes of left-right alternating small-amplitude tail beats. Neural circuits residing in the spinal cord are primarily responsible for these locomotor movements.

We previously generated computational models of spinal circuits to simulate several locomotor movements of embryonic and larval zebrafish.[1] To facilitate the creation of newer models of the nervous system of developing zebrafish for the study of swimming, *Sili*Fish was developed as a Windows-based software with an easy-to-use GUI for model creation, running simulations, and graphical display of analysis of swimming output.

This protocol explains how computational models of spinal neural circuits for swimming can be generated using *Sili*Fish and how to perform simulations of these circuits and display their output graphically. While several simulators of neural circuits are publicly available, several aspects of *Sili*Fish were developed to study spinal circuits and their control of swimming in zebrafish. For example, there are functions to facilitate the construction of circuits within an elliptic cylindrical structure like the spinal cord. There is also a feature for converting spinal cord output to movements that can be visualized through animation. *Sili*Fish is suitable for modeling spinal circuits for swimming at any life stages and for any species. Exact knowledge of the dimensions of zebrafish (cf. Dou et al.)[2] and the spinal cord can be useful in setting conduction delays between neurons. However, those conduction delays can be set by the user explicitly; therefore, prior knowledge of the physical dimensions is not necessary.

This section also explains the installation steps.

Version

The current protocol is for *Sili*Fish version 1.0.

Installation

Timing: <5 min

The executable file is available at https://github.com/Bui-lab/SiliFish/releases. Depending on the operating system, SiliFish.UI_1.0_32bit.exe or SiliFish.UI_1.0_64bit.exe can be directly downloaded to the preferred location on the hard drive. There are no installation steps; however, some antivirus software or Windows Defender can give a warning as the software is not officially certified. Those warnings need to be overridden.

Note: This protocol describes the use of the initial version of *Sili*Fish. New versions may be created to add useful new features. No changes are currently planned to the model creation and execution of simulation described herein. The procedures described below are expected to be applicable to future versions of *Sili*Fish.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
*Sili*Fish 1.0	The present protocol	https://doi.org/10.5281/zenodo.7378974 https://github.com/Bui-lab/SiliFish/releases
Microsoft Windows	Version 7[3] or later 32-bit or 64-bit	https://www.microsoft.com
Modeling spinal locomotor circuits for movements in developing zebrafish	Roussel et al.[1]	DOI: https://doi.org/10.7554/eLife.67453
Other		
Hard drive	160 MB for installation > 10 GB free space for outputs	
Memory	Minimum 8 GB >=16 GB preferred	

STEP-BY-STEP METHOD DETAILS

Switching to custom model generation

Timing: <1 min (for step 1)

*Sili*Fish provides the models generated by our previous study,[1] which are referred to as predefined models throughout this protocol. These models are accessible through the **Single Coil**, **Double Coil**, and **Beat and Glide** radio buttons at the top of the main window (Figure 1). These predefined models have very limited customization. To showcase the full features of *Sili*Fish, the steps of generation of fully customizable Custom Models is explained in this protocol. The following step shows how to switch to custom model creation mode.

1. Click on the **Custom** radio button at the top to switch the software to Custom Model mode (Figure 1).

Note: Switching between predefined and custom models will not cause you to lose data. However, clicking the Clear Model link will. Caution: Currently, there is no undo in *Sili*Fish.

Note: There are customizable versions of the predefined models that are saved as JSON files[4] and are located in the DataSet 1.0 folder on GitHub. (JSON is a commonly used

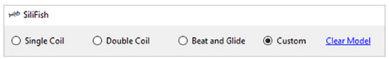


Figure 1. Switching to custom model generation
The software can be switched between predefined models (Single coil, double coil, and beat and glide) and custom model creation using the radio buttons on the top of the window.

human-readable text-based file format). In the program, you can click on the **Load Model** link and do the following steps on an already created model rather than creating from scratch, which will expedite the learning curve.

Setting general parameters

Timing: <5 min

The following section describes the general parameters related to the physical dimensions of the body and the spinal cord the user can define.

The **General** tab contains information on the size and structure of the model animal studied (Figure 2).

2. Define **Model Name** and **Description**.
3. Define body size.

Note: There is no specific unit of measure (UoM) used within *Sili*Fish in terms of length. The user can select any unit that is appropriate but needs to use the same UoM across the software for any length and speed measures. It is recommended to consider both the dimensions of the animal, the minimum distance between two cells, and the conduction velocity of the connections to determine the appropriate UoM. As many of the conductance velocities in the literature are reported as m/s and *Sili*Fish uses ms as the time unit, mm would be an appropriate starting point.

Note: The data types used in the code allow a minimum of 15 digits precision with an approximate range of $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$. Given appropriate UoM will prevent exceeding these limits.

Note: Depending on how the spatial distributions are defined in Creating Cell Pools and Connections, the spinal cord and the body can be considered an elliptic cylinder or a rectangular prism.

Note: The representation of the spinal cord and the body as an elliptic cylinder or a rectangular prism is an oversimplification intended to keep data entry relatively easy in the user interface. A more anatomically correct distribution can be modeled by entering absolute coordinate values in Creating Cell Pools and Connections.

   a. **Dorsal-Ventral**: Enter the value of the size of the fish body in the dorsal-ventral axis.
   b. **Medial-Lateral**: Enter the value of the size of the fish body from midline to the most lateral point.
4. Define the spinal cord size.
   a. **Dorsal-Ventral**: Enter the value of the size of the spinal cord in the dorsal-ventral axis.
   b. **Medial-Lateral**: Enter the value of the size of the spinal cord from midline to the most lateral point.
   c. **Rostro-Caudal**: Enter the length of the spinal cord.
   d. **Body Position**: Enter the distance of the spinal cord from the ventral region of the body.

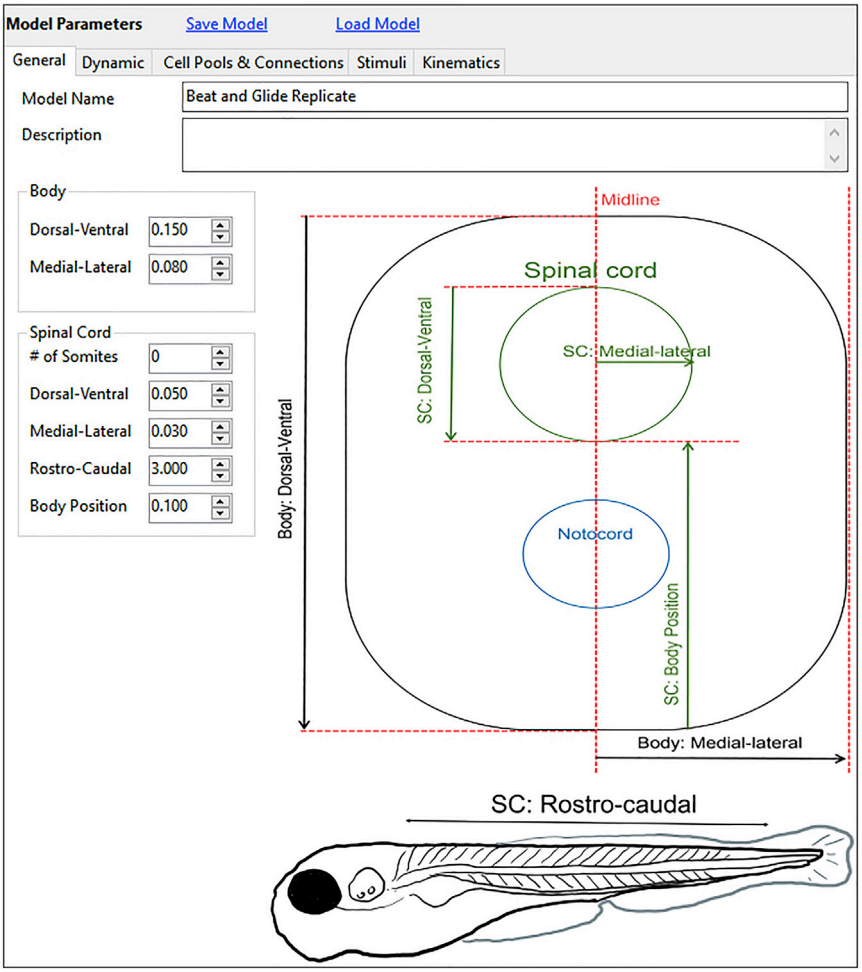


Figure 2. Setting general parameters

The General tab contains information about the model (like name and description) and the size of the model animal.

Setting dynamic parameters

Timing: <5 min

The following section explains how to update the values that will define the dynamics of the simulation, how cells will behave to a certain stimulus, and the timing of the propagation of electrical activities from one cell to another.

The **Dynamic** tab includes parameters relevant to the cellular dynamics of the cells or the model in general, like conduction velocity and reversal potentials.

5. **Conduction Velocity**: Enter the default conduction velocity down the axon of a neuron.

Note: The conduction velocity is used to calculate the time it takes for a change in membrane potential at the cell body to propagate without attenuation to the presynaptic terminal. The unit of measure is the UoM used to define the length values in Setting General Parameters, divided by time (ms). As explained in Creating Cell Pools and Connections, this value can be overridden in specific neuronal groups.

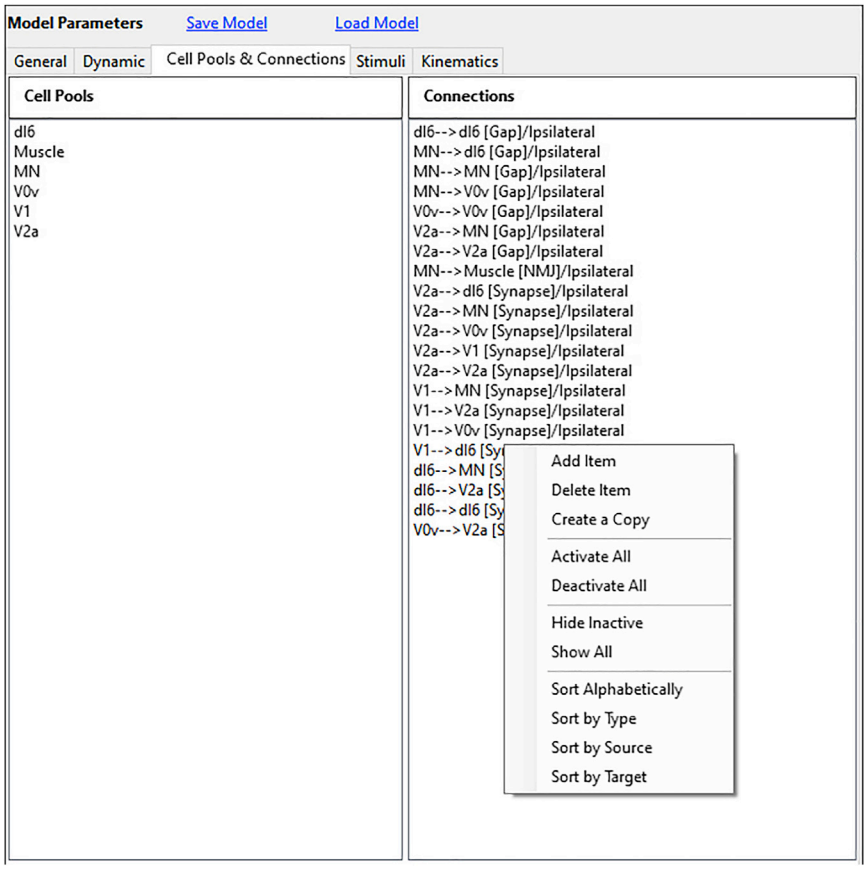


Figure 3. Creating cell pools & connections

The Cell Pools & Connections tab lists all the defined cell pools and the projections between them. By right-clicking the mouse, you can access different menu items to create, edit, delete the cell pools, or change how they are displayed in the list. See also Figures 4 and 9.

6. **E_ach, E_glu, E_gly,** and **E_gaba**: Enter the reversal potentials of acetylcholine, glutamate, glycine, and GABA, respectively.

   Note: As explained in Creating Cell Pools and Connections, these values can be overridden in specific neuronal groups. The UoM of reversal potentials is assumed to be mV.

Creating cell pools and connections

Timing: Depends on the complexity of the model

The following section explains how to define cell populations and connections between specific cells.

*Sili*Fish allows the definition of different neuronal and muscle cell groups, called cell pools, with different intrinsic properties, distribution patterns, and activity timelines. Furthermore, each cell pool can have electrical (gap) and/or synaptic (chemical) connections to themselves or other cell pools. The **Cell Pools & Connections** tab provides the user interface to create these cell pools and connections.

Defining cell pools and connections are the most crucial part of the model definition.

7. Create a cell pool: By right-clicking on the mouse on the **Cell Pools** panel and selecting Add Item menu, open the Cell Pool edit form (Figures 3 and 4).

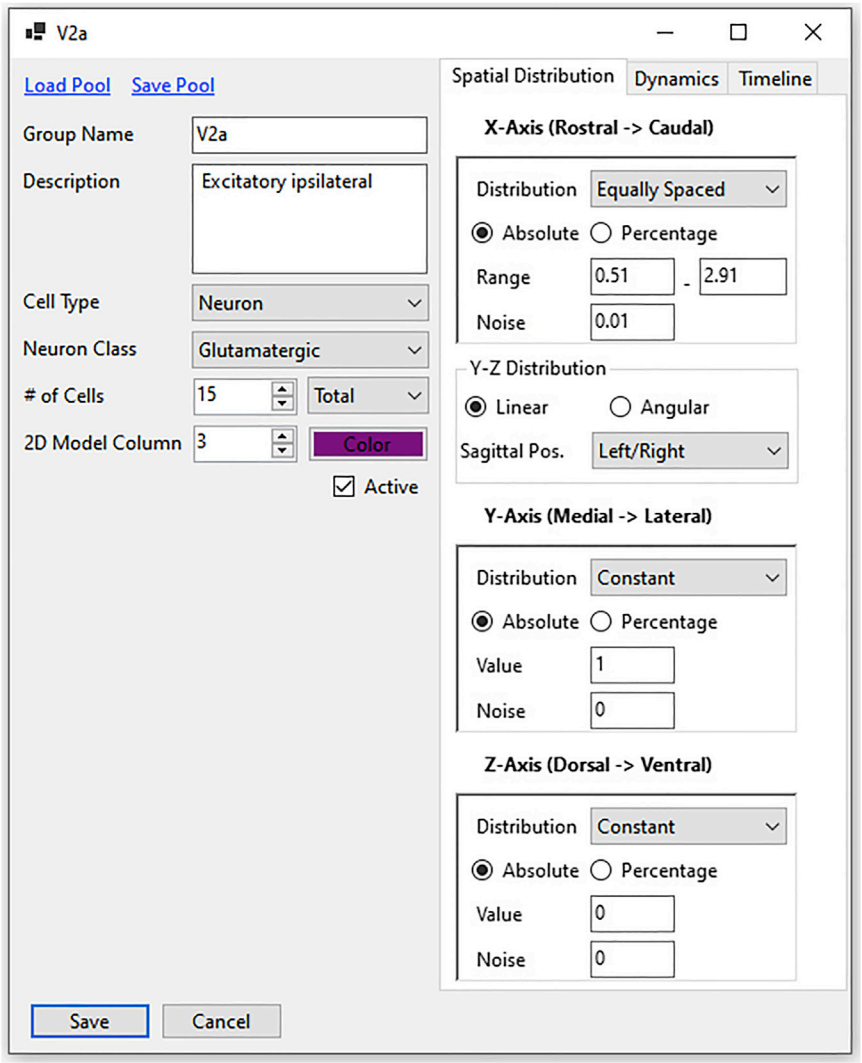


Figure 4. Editing a cell pool

Cell Pool edit form is where the user can define the cells' intrinsic properties, spatial distributions, and activation timelines. See also Figures 5, 6, and 7.

a. Enter a unique **group name** for the cell pool.

Optional: Enter the description field.

b. Select whether the **cell type** is a neuron or a muscle cell.

Optional: If the neuron is selected as the cell type, you can set the **neuronal class** (glutamatergic, GABAergic, etc.). This selection will allow the default reversal potentials to be used while creating the synapses.

c. **# of cells:** Define the number of cells within a cell pool.

Note: Number of cells can be defined as a total number for the model or as a number per somite.

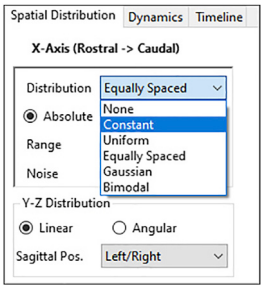


Figure 5. Types of distributions

Many values in SiliFish can be defined as a distribution (Uniform, Gaussian, Bimodal, Equally Spaced) or as a specific number.

d. **2D Model Column** is a multiplier of the y-axis used in creating a 2D plot, as explained in expected outcomes.

Note: 2D Model Column value is only for display purposes and does not affect the placement of the cells.

e. Select the **color** that will be used to represent these cells in 2D and 3D models, as well as plots.

Note: Using different colors for interacting cells will make it easier to understand the visual results. For example, if a cell is receiving multiple incoming currents from different cells, the plot displaying the incoming currents will show each current with the color of the source cell pool. The cell pool based coloring will allow reading the plots quickly.

f. Set whether the cell is **active** or not.

Note: It is possible to deactivate a cell pool without removing it from the model to run some quick tests.

g. Enter the **spatial distribution** of the cells to create more realistic models (Figures 5 and 6).

Note: Make sure you use the same UoM for distances and spatial distributions throughout the data entry.

Note: As there are many options in creating spatial distributions, it is recommended that the 3D model is frequently checked, as explained in the expected outcomes, to make sure the settings are entered properly.

Note: If the spatial distribution you are trying to achieve cannot be defined with the available options, you can divide the distribution into subparts and create multiple cell pools with the same intrinsic properties and connectivity patterns.

Note: For the spatial placement of the cells, different distributions can be selected to determine the cell's coordinates in each axis (Figure 5, shown for the x-axis). When constant distribution is selected, all the cells in the cell pool are assigned the same value in the given axis. When uniform distribution is selected, the cells are placed randomly using uniform distribution. You can define the minimum and maximum values the specific axis can take. When equally spaced distribution is selected, the cells are placed at the same intervals within the given minimum and maximum range. When Gaussian distribution is selected, the cells will be placed with a normal distribution,

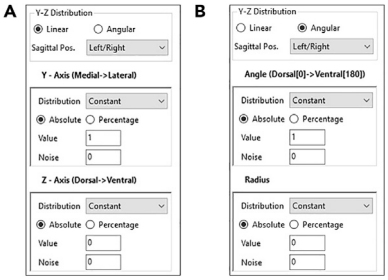


Figure 6. The Y-Z distribution modes

(A and B) The Y and Z axes distributions can be defined as (A) a linear distribution or (B) an angular distribution. The panels A and B are two different modes of the form based on the Linear/Angular selection.

with the given mean and standard deviation values. The cells can also be placed in a bimodal distribution with the given mean and standard deviation values.

Optional: If constant or equally spaced distributions are selected, you can also define a noise parameter. This optional noise value will be used as a standard deviation for a Gaussian distribution of mean one, which will, in turn, be a multiplier to the value generated by the distribution.

i. **X-Axis Distribution**: Enter the rostro-caudal distribution pattern of the cells in the cell pool.

Note: Using the Absolute/Percentage selection, you can select whether the x-values will be absolute coordinates or will be calculated by the maximum length of the axis, which is the spinal cord's rostral-caudal length defined in Setting General Parameters.

ii. **Y-Z Distribution**: Enter the medial-lateral (y-axis) and dorsal-ventral (z-axis) distribution pattern of the cells in the cell pool.

Note: The spatial distributions of the cells in the dorsal-ventral and medial-lateral axes can be defined in two ways: Linear or Angular (Figure 6). The linear distributions within the Y-Axis (Medial-Lateral) and Z-Axis (Dorsal-Ventral) are defined similarly to the X-Axis distribution explained above. If the percentage option is selected, the neurons will be placed within the spinal cord, whereas the muscle cells will be placed within the body. You can define the coverage's angle and radius range for angular distributions. The angle is 0° in the most dorsal region and 180° for the most ventral region. The distribution is defined only for half of the body. Using the sagittal position dropdown, you can define whether the cells exist on the left, right, or both sides.

Note: In the current version of *Sili*Fish GUI, only spinal neurons are considered and will be placed within the spinal cord. However, it is possible to simulate peripheral neurons by giving absolute coordinates rather than percentages.

Note: *Sili*Fish GUI allows the definition of only one medial-lateral or dorsal-ventral range of the body. However, the spinal cord of zebrafish is known to vary in size along the rostro-caudal axis. With this non-uniform size distribution in mind, *Sili*Fish does not check whether a coordinate is within the cylindrical or prismatic boundaries defined by the medial-lateral or dorsal-ventral ranges that is inputted by the user. For example, even if the user defines absolute y and x-axis values outside of this medial-lateral or dorsal-ventral range, *Sili*Fish would run without any issues. Thus, the user defined medial-lateral and dorsal-ventral ranges are used in combination with the **Percentage** option to facilitate assignment of coordinates based on some user-defined reference ranges.

h. You can define the intrinsic properties of the cells within the pool through the **Dynamics** tab.

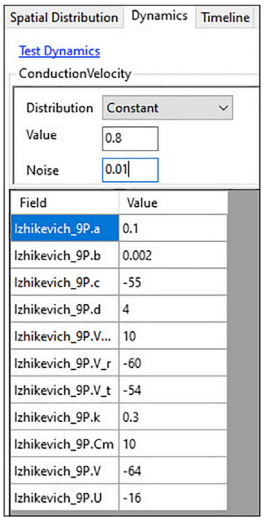


Figure 7. Setting dynamical properties of neurons

*Sili*Fish uses Izhikevich simple neuronal model for neurons. The parameters can be entered in the Dynamics tab on the Cell Pool edit form. All parameters can be defined as a specific number or a distribution to add some randomization to the model.

Note: Throughout *Sili*Fish, specific UoMs are used for consistency: All membrane potentials (V) are in millivolts (mV), current (I) is in picoAmperes (pA), resistance (R) is in gigaohms (GΩ), capacitance (C) is in picoFarads (pF).

i. If the **conduction velocity** of a specific cell pool is different from the rest of the model, it can be overridden here.

Note: The conduction velocity can be a specific number or defined as a distribution. If not entered, the conduction velocity set at the Setting Dynamic Parameters will be used.

ii. Enter the **Izhikevich values** (for neurons only) to define the membrane potential dynamics of the cell[1,5] (Figure 7).

$$C\frac{dV}{dt} = k*(V - V_r)*(V - V_t) - u + I$$

$$\frac{du}{dt} = a*(b*(V - V_r) - u)$$

$$if\ V > V_{max} \rightarrow V = c; u = u + d$$

Note: The Izhikevich model shows how the membrane potential (V) and the feedback current (u) change across a certain incoming current I:

V is the membrane potential that changes throughout the simulation. **u** is the recovery variable that allows the relaxation of the cell after an action potential. The UoM of **u** is pA. The initial values of **V** and **u** can be set through the grid (Figure 7). Parameters **a** and **b** represent the time scale and the sensitivity of **u**, respectively. They are unitless within the formalism used by the Izhikevich model. The membrane potential **V** resets to value **c** (in mV) after a spike. Similarly, recovery variable u is incremented by the value **d.** V_{max} represents the maximum membrane potential at the time of a spike, and V_r is the resting membrane potential. V_t is the threshold potential for a spike. All membrane potentials are in mV. **k** is an approximation of the subthreshold region of the fast component of the I-V relationship of the neuron. Finally, **Cm** is the membrane capacitance (pF).

iii. **Leaky Integrator values** (for muscle cells only): Enter the resistance and capacitance values of the muscle cells.

Note: *Sili*Fish uses the leaky-integrator model for muscle cells, where R and C are the resistance and the capacitance of the muscle cells, respectively. The UoM of resistance and capacitance are GΩ and pF, respectively.

Note: It is possible to see how a neuron behaves to a specific stimulus or calculate its rheobase value using the Test Dynamics tool (Figure 8). This feature is beneficial in generating cell populations that mimic the firing behavior observed in other studies in the literature.

Optional: It is possible to define a timeline with one or more start and end times through the **Timeline** tab. The cells will be silenced outside these time windows. If no timeline is entered, the cells will be active throughout the run. This option allows users to investigate the effects of silencing specific populations of cells on circuit and motor activity.

Note: For complex models, deactivating cells can also help visualize them in 2D and 3D models. Inactive cells and cell pools are not displayed in the 2D and 3D models. Temporary deactivation can help complex models to be more manageable.

Note: Creating the cell pools with the right parameters is crucial in generating a successful model. As there are many parameters that can be set in creating a cell pool, it can be very time-consuming. The **Create Copy** menu item accessible through the mouse right-click on the **Cell Pools & Connections** tab (Figure 3) allows for creating a copy of an existing cell pool, which helps create similar cell populations. Duplicating a cell pool using **Create Copy** option will also create corresponding duplicates of its connections.

Note: If different models share cell pools with similar characteristics, you can save a cell pool using the **Save Pool** link (Figure 4) in one model and load it back in another.

8. After defining the cell pools, define the projections from one cell pool to another by mouse right-clicking and selecting **Add Item** on the **Connections** list (Figure 3). A window to edit the projection properties will open (Figure 9).
   a. Select the **Source Pool** and **Target Pool.**

   Note: In the case of a gap junction, which cell pool is defined as the source pool or the target pool is not important.

   b. **Axon Reach**: Enter whether the connections are ipsilateral, contralateral, or bilateral.
   c. Depending on source and target pool cell types, select whether the **connection type** is a gap junction, a synapse, or a neuromuscular junction.
   d. Define how the distance between two cells will be calculated.

   Note: There are two distance modes you can select from: Euclidean and Manhattan. In Euclidean mode, the distance between the two points is calculated as the shortest path between the two. In Manhattan distance, the sum of x, y, and z distances is used.

   Note: Manhattan distance calculation may be helpful, for example, in the case of a neuron projecting first in the dorsal-ventral axis and then contralaterally.

   e. To add some randomization, you can define the **probability** of the presence of the connection between two cells.
   f. Enter the maximum conductance value (in nS) for the connection using the **weight** field.

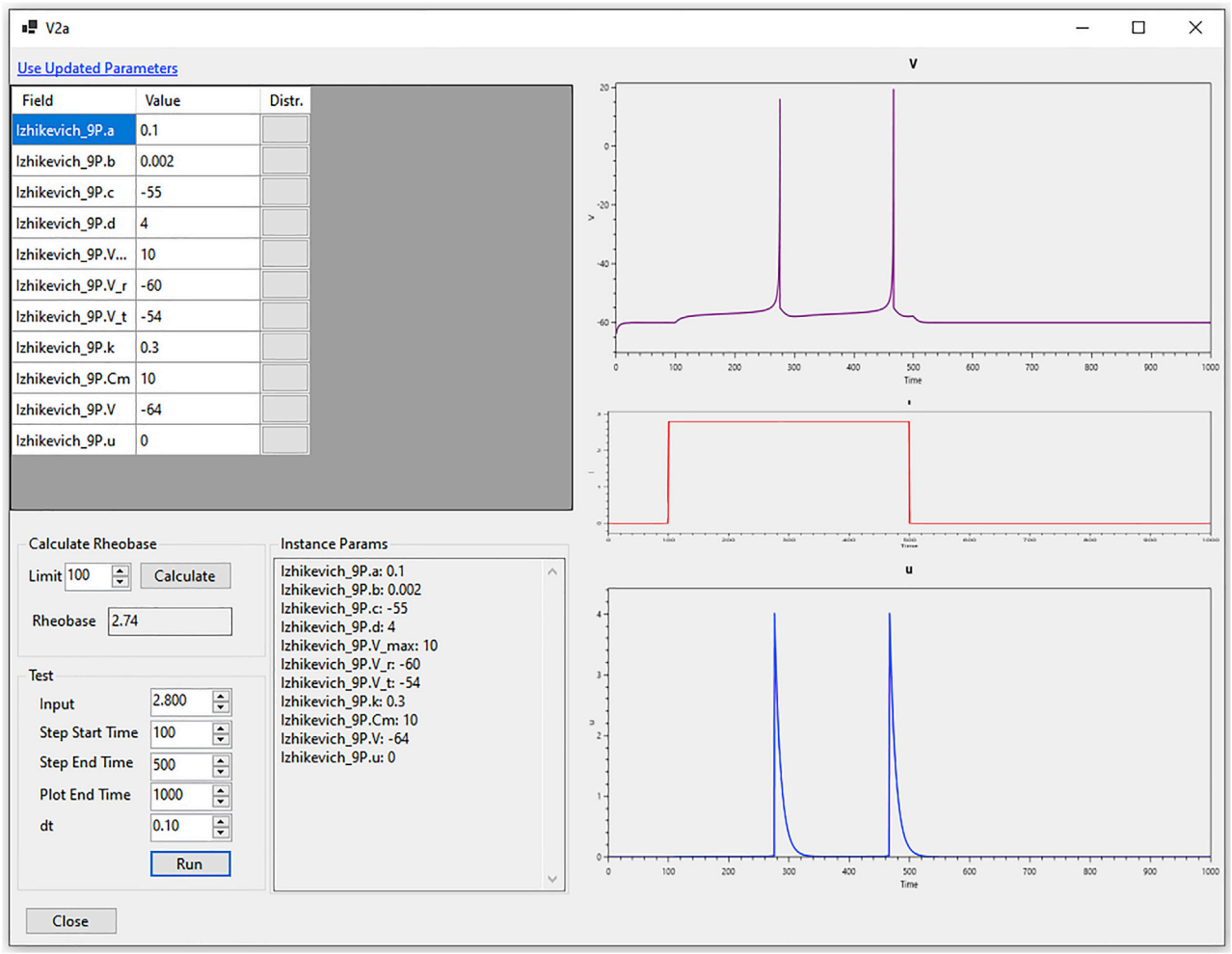


Figure 8. Test dynamics form

Test Dynamics form allows the user to calculate the rheobase of a neuron and test how it behaves to a specific stimulus.

g. *Sili*Fish will automatically populate the **name** of the connection. You can override this value if you prefer to use another naming convention.

Note: If you change the name of a connection, *Sili*Fish will stop automatic naming for this connection.

Optional: You can enter a description of the details of the cell pool.

h. Enter the limits of the reach of the connection.

Note: In the case of the model consisting of somites, you can define whether the connection can be within the same or to a different somite. The minimum and maximum reach fields define the possible extent of the connection. The ascending and descending reach fields define the maximum extent in the spinal axis (the x-coordinate).

Optional: The time it takes for a potential change from one cell to another is calculated as distance/conduction velocity. If the model requires a delay on top of this calculation, you can define it here. You can also enter a fixed duration, in which case the distance, the

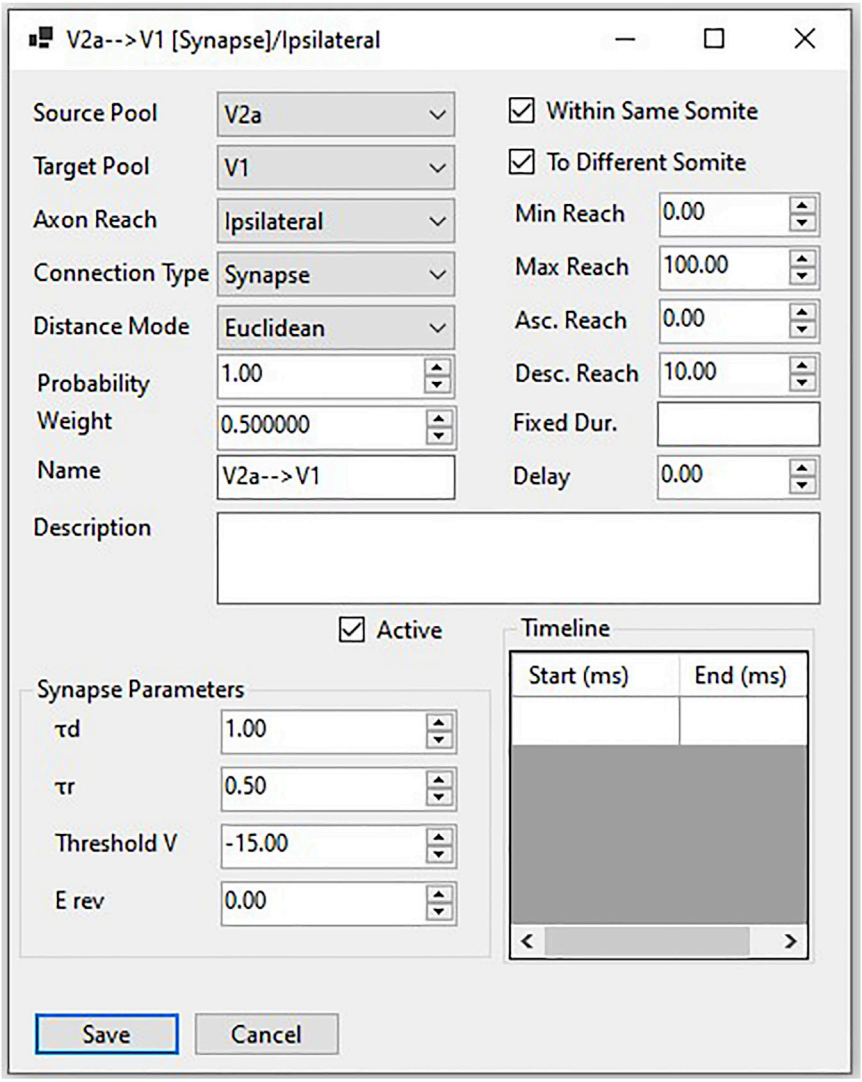


Figure 9. Defining projections

The user can define multiple features of a connection from one cell pool to another: the connection type and weight, the intrinsic properties, the reach information, and the activation timeline.

conduction velocity, and the delay values will be obsolete. All durations are assumed to be in ms.

i. If the connection is a chemical synapse, the time course of the current is a sum of two exponentials. Enter the following **synapse parameters:** drop and rise decay times (**τd** and **τr**), the threshold membrane potential (**Threshold V**), and the reversal potential (**E rev**).

Optional: It is possible to define a timeline with one or more start and end times. The connections will be inactive in the times outside these time windows. The connections will be active throughout the run if no timeline is entered.

Optional: You can deactivate a connection by unchecking the Active check box on the connection edit form. Deactivation can help test different scenarios without removing the connections from the model.

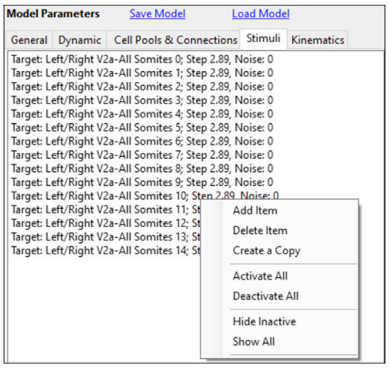


Figure 10. Defining stimuli

The Stimuli tab lists all the defined stimuli. You can access different menu items to create, edit, or delete a stimulus by right-clicking the mouse. See also Figure 11.

Defining stimuli

Timing: <5 min

The following section explains how electric stimuli can be applied to cells in the model.

If the model requires an external stimulation to be applied, you can define them by clicking the Stimuli tab, right-clicking to open the pop-up menu, and selecting Add Item (Figure 10). The stimulus edit form will open (Figure 11).

9. You can define stimuli that will be applied to one or multiple cells or the whole cell population within a cell pool.
   a. Enter the **stimulus type** (step, ramp, or Gaussian) and corresponding values: stimulus value for the step stimulus, the start and finish values for the ramp stimulus, and the mean and the standard deviation values for the Gaussian stimulus.
   b. Enter the cells to which the stimulus will be applied.

   Note: A stimulus can be applied to the desired cells located in all somites, a single somite, or a range of somites (e.g., 2–7). Similarly, within a somite, the stimulus can be applied to all cells, a single cell, or a range of cells.

   Note: The UoM of all stimuli and currents are in picoamperes (pA).

   Optional: It is possible to define a timeline with one or more start and end times. The stimulus will be zero in the times outside these time windows. Otherwise, the stimulus will be active throughout the run if no timeline is entered.

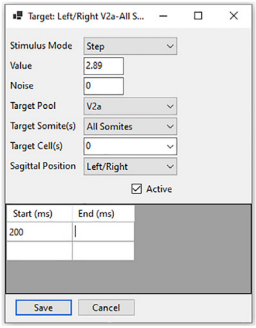


Figure 11. Stimulus edit form

The stimulus edit form is where the user can define a stimulus's amount, timing, and target.

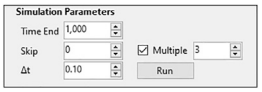


Figure 12. Setting simulation parameters

You can enter the simulation duration, the time that will be skipped for initial conditions to subside, the unit of time used in calculations, and how many times the simulation will be run.

Optional: You can deactivate a stimulus by unchecking the **Active** check box on the stimulus edit form. This can help test different scenarios without removing the stimulus from the model.

Setting simulation parameters

Timing: <1 min

The following section explains how to set the length of the simulations.

10. In the **Simulation Parameters** section, enter how long the model will be run (**Time End** – in ms) and how much time will be used to wait for the initial conditions to subside (**Skip** – in ms) (Figure 12).

Note: The simulation results within the first period that is skipped will not be visible in any of the plots or animations.

Note: To decide the right amount of time to skip, you can run the model for a short amount of time without any stimulus and look at the plots to see when the model comes to an equilibrium state.

Optional: By default, *Sili*Fish uses 0.1 ms as the unit of time in running the iterations in the model. If the model requires a faster or slower response time for the cells, you can set the **Δt** parameter.

Optional: It is possible to run a simulation multiple times to collect data for statistical analysis. If certain parameters are probabilistically assigned, the parameter values will be reassigned in each run. To run a simulation multiple times, you need to check the **Multiple** check box and enter the number of times you want to run the simulation for. The cellular information of each run and the episode information will be saved as JSON and CSV files in the output folder.

Note: Depending on the complexity of the model and the duration, multiple runs may take a long time. Make sure the model is running to your expectation before working on statistical analysis.

Note: When the simulation is run multiple times, you can only see the plots of the last run.

11. Save the model as a human-readable JSON file using the Save Model button for future use.

Note: Some changes may be easier done on the model JSON file than using the *Sili*Fish UI. If you want to change all of the noise standard deviation values of the X-coordinates, for example, a text editor's find/replace feature will be faster than opening up every cell group and updating its X-distribution parameters. Instead, you can save the model as a JSON file, open the file with a text editor of your choice, edit it, and reload it from *Sili*Fish (Figure 13).

Setting kinematic parameters

Timing: <5 min

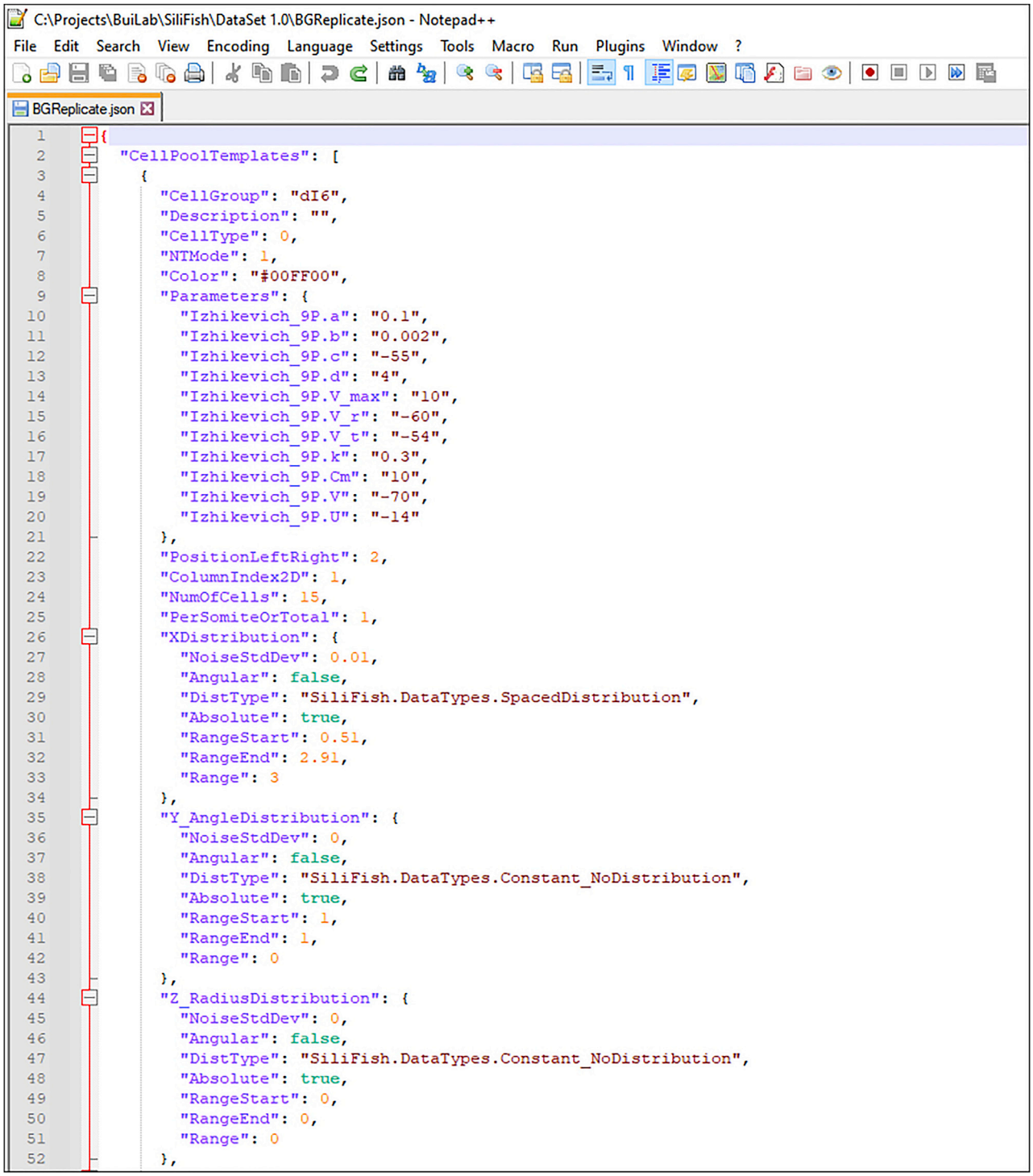


Figure 13. An alternative way to edit models
You can save the model templates as a JSON file for transferability and easy editing.

After the model is run, you can visualize the behavior of the fish by converting the muscle cells' membrane potentials to muscle contractions. How to update the parameters of this visualization is explained below.

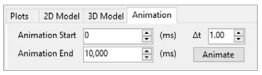


Figure 14. Setting animation parameters
The animation start and finish times and the time resolution (Δt) can be set by the user.

The parameters of muscle membrane potential to muscle contraction conversion can be set by clicking the **Kinematics** tab.

The force generated by the contraction of the muscle cells in each somite depends on the activity of the motoneuron innervating that somite. The details of this conversion can be found in our previous model.[1] In summary, the membrane potential of the muscle cells in each somite is converted into oscillation angle using the following formula.

$$\theta_i'' + 2\zeta\,\omega_0\theta_i' + \omega_0^2\theta_i = \delta\left(V_{right\ muscle} - V_{left\ muscle}\right)$$

12. Enter the kinematic parameters: the **damping coefficient** (ζ), the natural oscillation frequency (ω_0), and the **conversion coefficient** (δ).

    Optional: The Alpha and Beta values are used to calculate the conversion coefficient by the δ = α + ß*R formula, where R is the average of the resistance of the muscle cells within a somite. If the calculated value is 0, the Conversion Coefficient entered by the user will be used.

    Note: Entering zero to alpha and beta values will make sure the entered conversion coefficient is used for every somite.

13. Enter the **Boundary** and **Delay** values that will be used to detect tail beat episodes.

    Note: If the tip of the tail moves in the y-axis at least the Boundary amount, it will be considered the start of an episode. If the dislocation from the central axis is less than the Boundary amount on the left and right for the duration of Delay, it will be considered a rest between episodes.

    Note: Unlike other parameters, kinematic parameters can be modified after running the model. The kinematic parameters are used only to generate the body movement animation using the run results; these parameters do not affect the neuronal activity of the cells.

Generating animation

Timing: <5 min

How to generate the swimming animation using the parameters set in Setting Kinematic Parameters is explained below.

14. Enter the **Animation Start** and **Animation End** time in milliseconds (Figure 14). By default, these values are set to 0 and the simulation end time respectively.
15. Enter **Δt**, the unit of time, for the animation. Using an animation Δt greater than simulation Δt will create a smaller animation file.
16. Click the **Animate** button. The animation will be generated as an HTML file and displayed on the window.

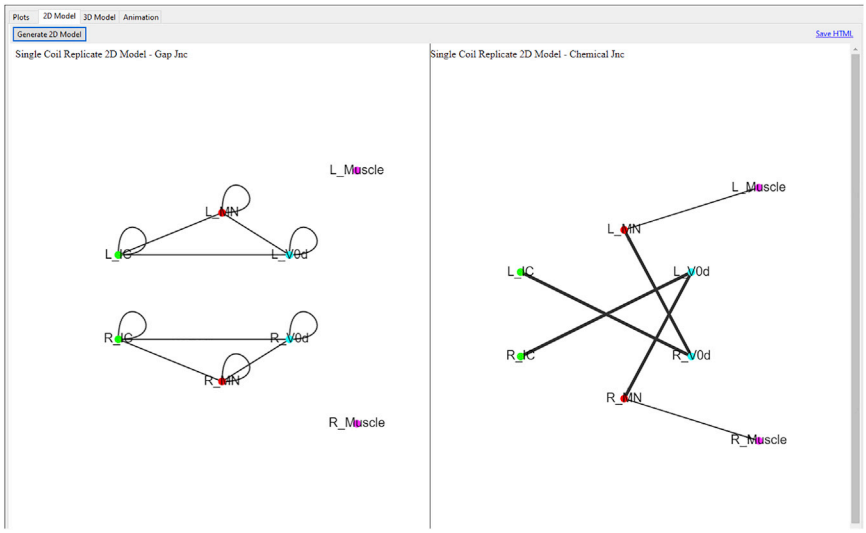


Figure 15. 2D models

The interactive 2D models display the cell pools and their connections. Generating 2D models is a good first step in testing the model's accuracy.

EXPECTED OUTCOMES

*Sili*Fish calculates each cell's membrane potential changes based on their intrinsic properties and the inputs they receive (stimuli, gap junctions, synapses, and neuromuscular junctions) across a timeline. The output of *Sili*Fish can be split into three broad groups: (1) Model visualization, (2) Membrane electrical properties of the cells, and (3) Generated motion.

Model visualization: *Sili*Fish offers to visualize the models in 2D and 3D. The 2D model (Figure 15) displays a summary of connections between the cell pools. The 3D model, on the other hand (Figure 16), displays individual cells at their actual locations within the model. These visuals help test whether the model is defined properly.

> ***Note:*** The interactive plots are generated by using third-party tools: the 2D model by force-graph,[6] 3D models by 3d-force-graph,[7] the HTML plots by DyGraph,[8] and the animations by AmCharts.[9] The animation outputs require an internet connection. The other interactive plots (2D/3D models and plots) use the internet to generate more streamlined files. However, these plots can still be generated for users that are offline at the cost of larger HTML files. Note that in all cases, the connection with third-party tools is only to download the JavaScript files necessary to create the models/plots/animations. There is no data sent from the user to any server.

Membrane electrical properties of the cells: You can see how the membrane potentials, input currents, or applied stimuli change over time (Figure 17). You can define which cell pools, how many and which somites or cells will be plotted. In addition, *Sili*Fish offers faster image plots and interactive HTML plots in which you can zoom in and out and hover over to get information on the data. Viewing the image plots first is recommended to figure out which region you need to get more information on and create the HTML plots for that specific time frame to make the plotting more efficient. *Sili*Fish also displays how many plots will be generated based on your selection. It is recommended to keep the number small to make sure your hardware will suffice for quick visualization.

Generated motion: As mentioned in the Setting Kinematic Parameters, the membrane potential differences between the left and right muscle cells are converted into tail beats. *Sili*Fish generates plots to display various statistics that can be useful to visualize or quantify tail beats (Figure 18). These

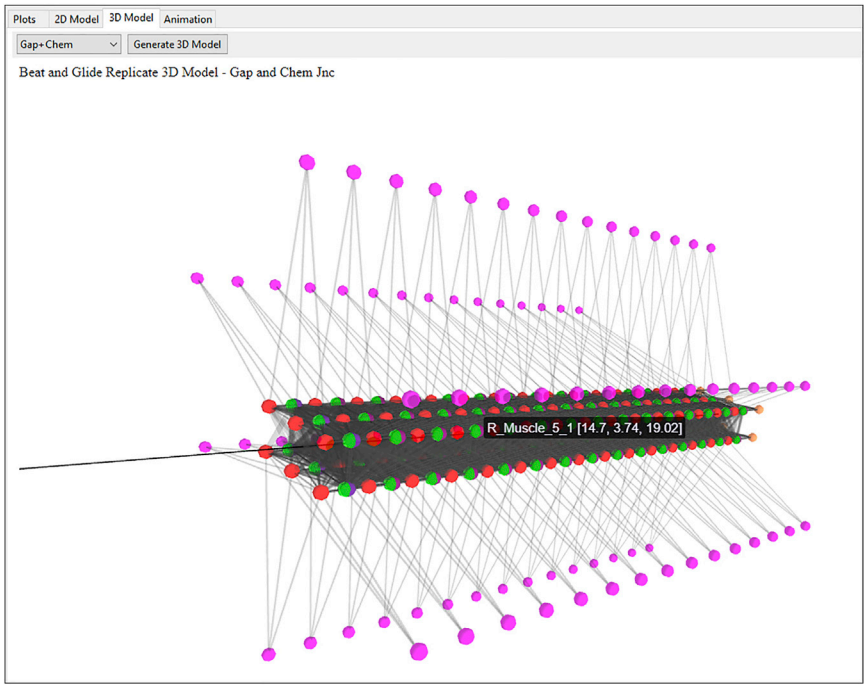


Figure 16. 3D models

The interactive 3D models display the individual cells and their connections. Generating 3D models is a good first step in testing the model's accuracy.

plots are as follows: The **tail movement** plot shows the movement of the tip of the tail relative to the midline. Tail beats can be composed of a series of tail beats followed by a resting period, called an episode. The **episode duration** plot shows the duration of each episode throughout the run time of the model. The time between two successive episodes is plotted on the **episode intervals** plot. The frequency of individual beats is plotted on the **instantaneous frequency** plot.

The tail beat frequency, calculated as the number of beats per episode duration, is plotted on the **tail beat frequency** plot. The number of tail beats per episode is displayed on the **tail beat/episode** plot.

If multiple plots are displayed, they are all synchronized based on the time axis so that you can zoom in and out of the same region for related plots.

Animation: The behavior of the model can be visualized as a damped pendulum as described in Setting Kinematic Parameters step. An example of the output animation can be seen in Methods video S1: Animation.

LIMITATIONS

*Sili*Fish is currently running only on Windows platforms. As it is not converted to a library format yet, the model creation is limited by the features implemented in the user interface.

*Sili*Fish uses several third-party tools specifically for generating visual outputs. These tools use an internet connection; however, no data is sent from the user.

We compared the speed performance of *Sili*Fish to our previous Python-based single-coiling model.[1] For example, a simulation of 10,000 ms takes about 20 min to run in our previous Python-based model, whereas the same model runs in only 10 s using *Sili*Fish on the same personal laptop computer (Intel(R) Core(TM) i7-8550U CPU, 16 GB RAM).

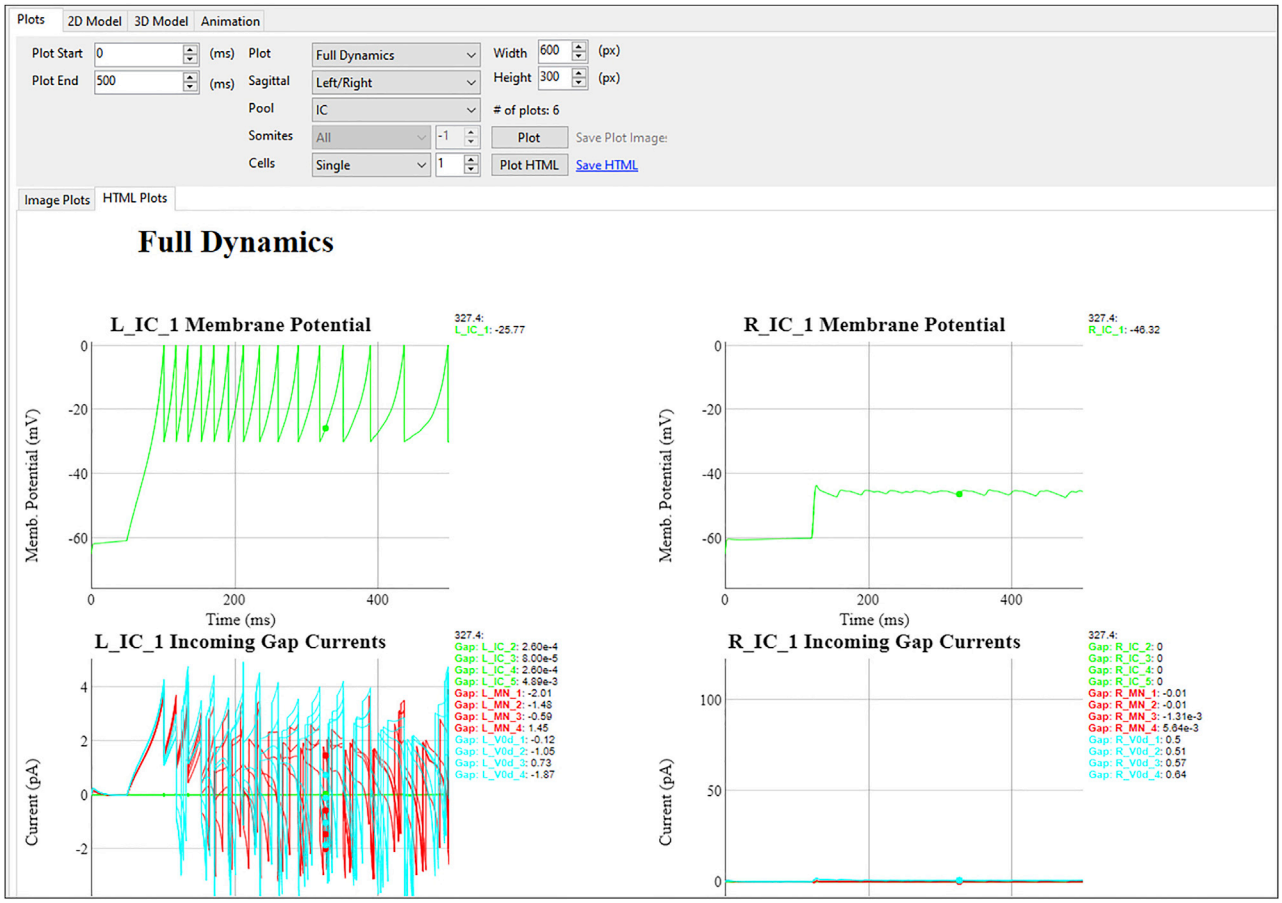


Figure 17. Plots

The interactive HTML plots can be generated to show the membrane potentials, applied stimuli, or incoming currents for one or more cells or cell groups.

Because of how the GUI is set up, *Sili*Fish 1.0 currently allows one hundred cells per somite per cell pool. However, there is no limit to how many cell pools the user can define. We tried using SiliFish with 1,200 cells for a 1,000 ms simulation for testing purposes. The simulation took about 4 min to run. The simulation took nearly 20 min to run when the cell number was raised to 2,400 cells. However, 16 GB RAM was insufficient to simulate a model with 1,200 cells for 10,000 ms or 3,600 cells for 1,000 ms. *Sili*Fish will benefit from further optimization in memory allocation in future versions.

Although we use *Sili*Fish to model larval zebrafish swimming, its use is not limited to zebrafish or the larval stages. However, there are some limitations of the software that may make them less fit for more complex animals. The most obvious one is that even though network dynamics can be studied for other locomotion behaviors, the actual conversion to locomotion other than carangiform swimming cannot be done.

The conversion of neural output to locomotion also does not consider the fluid dynamics of the environment, which may be important to consider in the swimming behaviors of adult fish.

There is also no capability for real-time neuromodulation. It is possible to create multiple versions of the neurons with different intrinsic properties to mimic behavior under the effects of neuromodulation. You can turn them on or off using the timeline feature; however, there is no feature to incorporate the real-time influences on neuromodulation on neural activity.

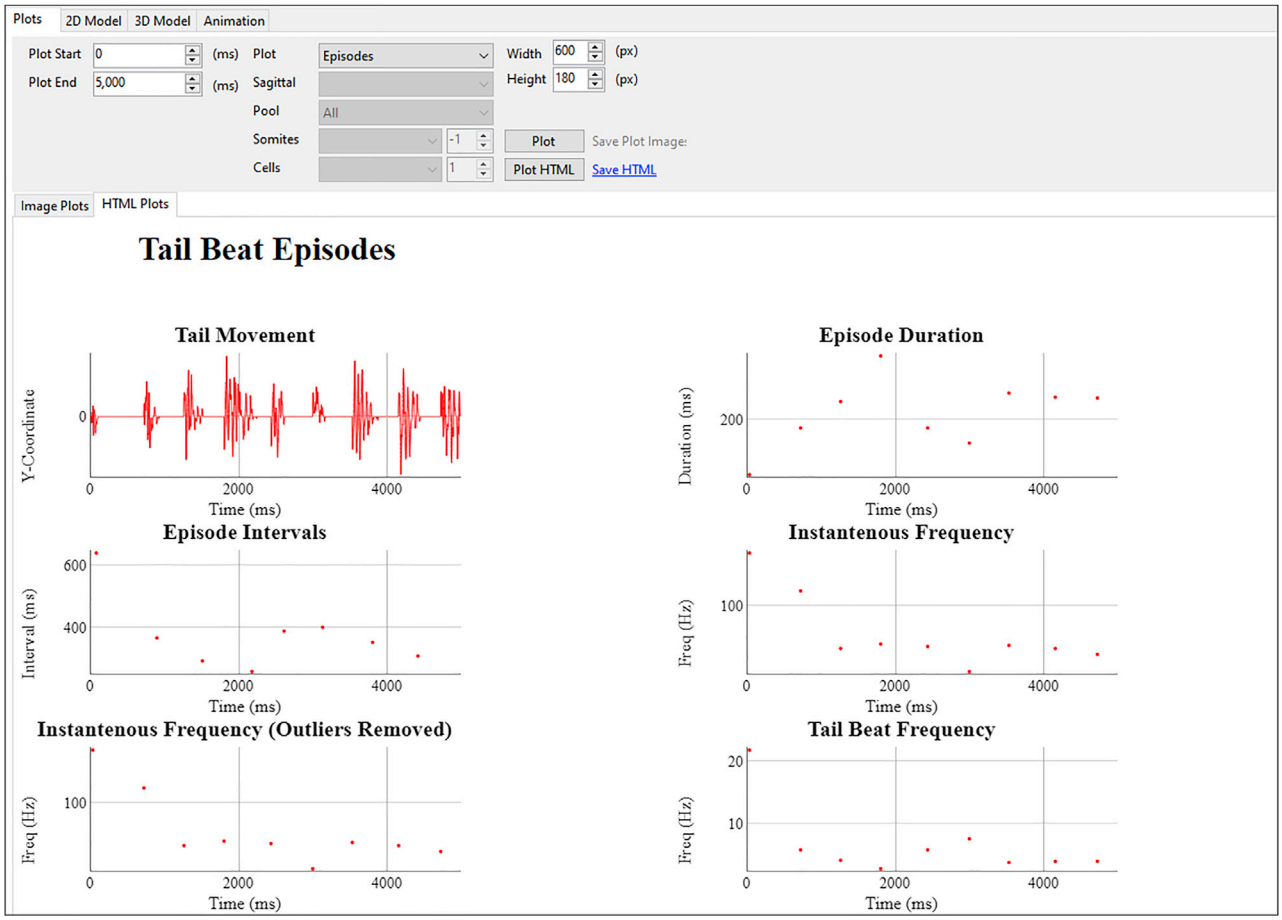


Figure 18. Plotting motion related values

The interactive HTML plots can be generated to show the tail movements and related quantifications.

TROUBLESHOOTING

You can encounter several issues while creating or running a model. Below are the most frequent ones that we have encountered.

Problem 1

The 2D and 3D models do not show the links between certain cell pools.

Potential solution

Make sure the source and target pools are properly entered at Creating Cell Pools and Connections - step 8-a, and the weight value in step 8-f is greater than zero. The thickness of the connection lines is proportional to the weights. Therefore, if the weight for this specific connection is too small, the corresponding lines in the generated 2D/3D models may be too thin. Try zooming in to be able to see them.

Problem 2

There is no locomotion in the created animation.

Potential solution

Plot the full dynamics of the muscle cells to see whether they are receiving any activation from any neurons. If there are no inputs, check whether there are any connections to muscle cells at Creating Cells Pools and Connections - step 8. Finally, check whether the timeline of any of the muscle cells or connections prevents the activity.

Problem 3

In the generated plots, muscle cells have incoming connections but no input current.

Potential solution

Plot the full dynamics of the motor neurons that are supposed to activate the muscle cells and check whether they are receiving any input. Continue to check pre-neurons' activity until you reach the main activation source. The link from the external stimuli to the muscle cells may be broken at one point, either by a missing connection or an entered timeline.

Problem 4

All the connections are properly set, and the plots show the neurons are spiking and muscle cells are triggered; however, there is still no locomotion in the created animation.

Potential solution

Check the kinematic parameters at Optional: Setting Kinematic Parameters - step 10. Try higher conversion coefficient values (δ).

Problem 5

The simulation run time is too long.

Potential solution

The run time of the model depends on the complexity of the model and the duration of the simulation. For complex models, try running shorter periods until the troubleshooting steps are completed. Running a simulation for 1,000 ms should suffice in most cases.

> *Note:* If there are many cellular pools or large numbers of cells within each pool, we suggest activating the cell pools (Creating Cell Pools and Connections - step 7-g) one by one, or starting with a lower number of cells per each pool, before running the full model for longer periods of time.

> *Note:* We recommend saving the model file before running the simulation for long periods of time to prevent data loss due to uncaught exceptions that may exist in the code.

Problem 6

Intermittently, the HTML plots cannot be generated and give error messages (Figure 19).

Potential solution

The interactive HTML plots require a high amount of memory and take longer to generate. If *Sili*Fish is unable to display the plots or models, the generated HTML files will be saved to the *Sili*Fish output folder. You can access the files outside of *Sili*Fish if that happens.

Problem 7

The HTML plots cannot be generated and give error messages (Figure 19).

Potential solution

Try creating the image plots and find out what time regions you need to have the interactive capability. Then, define the plot range to that period so that memory will be sufficient.

Problem 8

The animation HTML cannot be generated and give error messages (Figure 19).

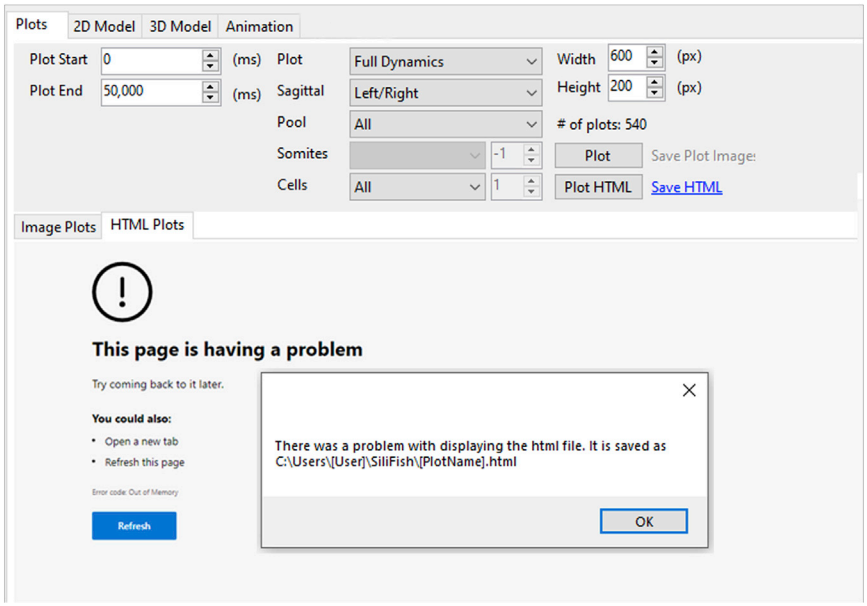


Figure 19. Example of an error message

The interactive HTML plots require high memory. The generated HTML files will be saved to the hard drive if the plots cannot be displayed due to low RAM. This image is edited to show two possible messages that the user might receive.

Potential solution

Depending on the duration of the simulation, the size of the animation HTML file can be too large for the system resources to display properly. Try recreating the animation plots using a larger time resolution (**Δt**) than the simulation (Figure 14). For example, if the Δt of the simulation is 0.1 ms, you can try 1 ms. Lowering the time resolution may cause the animation to have sharper movements but will allow the user to pinpoint a time range that would benefit from a higher time resolution. The animation start and finish time can be set accordingly (Figure 14), and the time resolution can be brought back to the original simulation Δt to generate higher-quality animations for shorter periods.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Tuan Vu Bui (tuan.bui@uottawa.ca).

Materials availability

There are no materials required.

Data and code availability

All the source code and sample data are available at https://github.com/Bui-lab/SiliFish/.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2022.101973.

ACKNOWLEDGMENTS

This research is funded by NSERC Discovery Grant RGPIN-2022-03898, NSERC Canadian Graduate Scholarship Award (NSERC 566014-2021), and Ontario Graduate Scholarship (OGS).

AUTHOR CONTRIBUTIONS

Conceptualization, E.T., Y.P., T.V.B.; Methodology, E.T., Y.P., T.V.B.; Software, E.T.; Validation, E.T., Y.P., T.V.B.; Investigation, E.T.; Writing –Original Draft, E.T.; Writing –Review & Editing, E.T., T.V.B.; Visualization, E.T.; Funding Acquisition, T.V.B.; Supervision, T.V.B.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Roussel, Y., Gaudreau, S.F., Kacer, E.R., Sengupta, M., and Bui, T.V.((2021). Modeling spinal locomotor circuits for movements in developing zebrafish. Elife *10*, e67453. https://doi.org/10.7554/elife.67453.

2. Dou, Y., Andersson-Lendahl, M., and Arner, A. (2008). Structure and function of skeletal muscle in zebrafish early larvae. J. Gen. Physiol. *131*, 445–453. https://doi.org/10.1085/jgp.200809982.

3. Microsoft. (2009). Microsoft Windows 7 (Microsoft). https://www.microsoft.com/en-ca/windows/.

4. Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., and Vrgoč, D. (2016). Foundations of JSON schema. In Proceedings of the 25th International Conference on World Wide Web, pp. 263–273.

5. Izhikevich, E.M. (2007). In Dynamical Systems in Neuroscience, T.J. Sejnowski and T.A. Poggio, eds. (The MIT Press). https://doi.org/10.7551/mitpress/2526.001.0001.

6. Asturiano, V. (2022). force-graph. https://github.com/vasturiano/force-graph.

7. Asturiano, V. (2022). 3d-force-graph. https://github.com/vasturiano/3d-force-graph.

8. Vanderkam, D. (2022). DyGraph. https://dygraphs.com/.

9. Antanas, M., Gediminas, T., Paul, A., and Alan. (2022). amCharts 5. https://www.amcharts.com/.