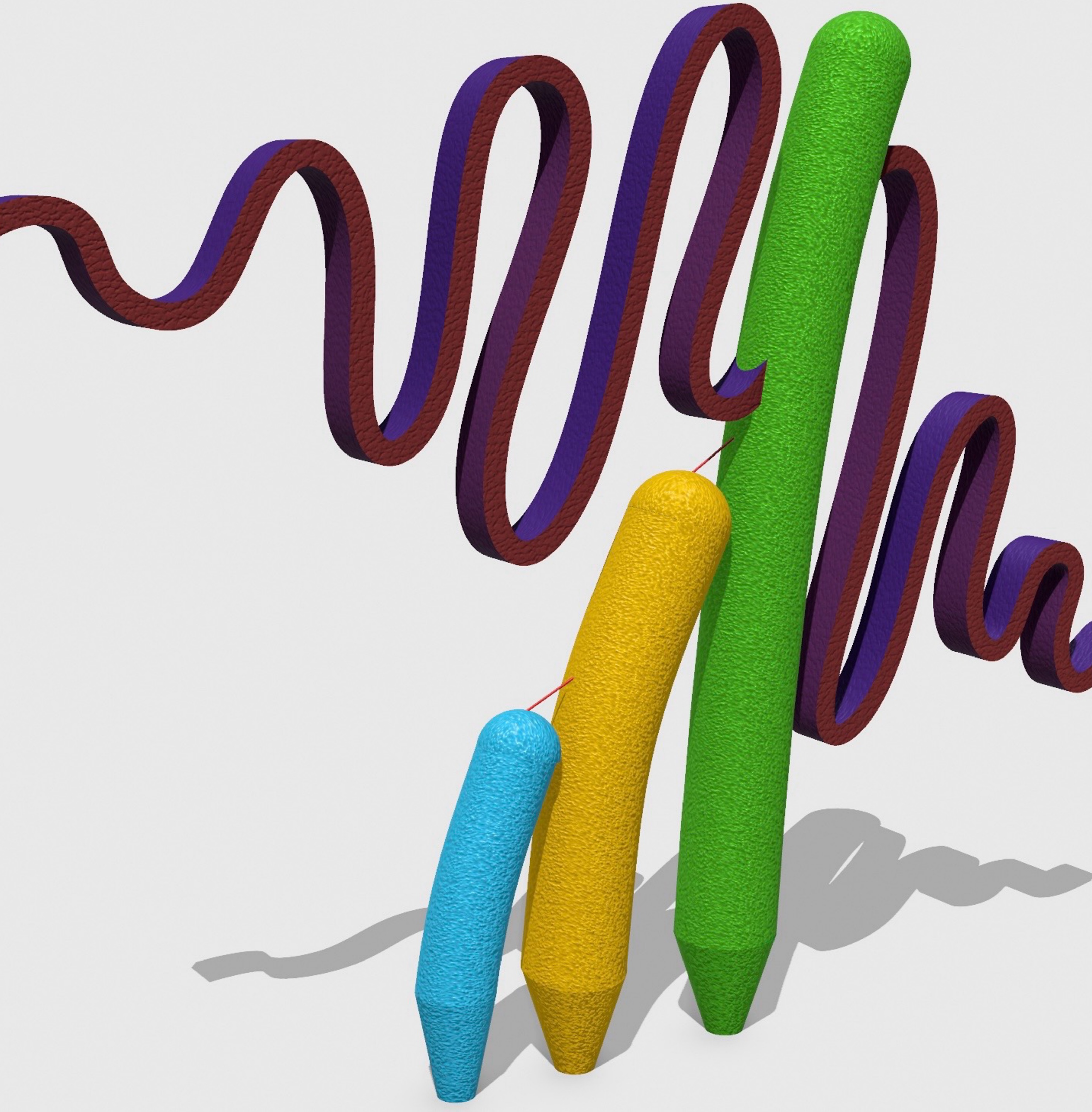
HairBundleLab

The Mammalian Cochlear Hair Bundle
Response Simulator



Varun Goyal

Department of Mechanical Engineering
University of Michigan, Ann Arbor, MI, USA, 48109

Karl Grosh

Department of Mechanical Engineering
Department of Biomedical Engineering
University of Michigan, Ann Arbor, MI, USA, 48109
Kresge Hearing Research Institute, Ann Arbor, MI, USA, 48109

Contents

Chapter 1

Introduction

Welcome to HairBundleLab, your ultimate numerical tool for exploring the complexities of mammalian cochlear hair bundle (HB) responses to step-like stimuli. Whether you're using force input mechanisms like fluid-jets or displacement inputs such as rigid probes, HairBundleLab offers a platform to understand and visualize these intricate processes.

With HairBundleLab, you are not just observing, you are in control. Craft your own HB models, tailoring them precisely by adjusting geometrical, mechanical, and electrical parameters to suit your experimental needs. The app provides you with an in-depth look at HB behavior, offering real-time visualizations of displacements and mechano-electric transducer (MET) currents in the time domain.

This comprehensive documentation serves as your go-to tutorial, guiding you step-by-step through the app's interface. You will discover what each parameter represents, how to modify them effectively and gain insights through detailed illustrations and practical examples. Whether you're a seasoned researcher or a curious learner, HairBundleLab is designed to enhance your understanding and facilitate your research in auditory science with ease and precision.

1.1. Documentation Structure

This document is designed to provide comprehensive information about the system requirements and necessary software for running HairBundleLab.

In Chapter 2, you will find detailed information about the hardware and software prerequisites necessary to run HairBundleLab efficiently.

Chapter 3 offers a thorough guide on setting up HairBundleLab, including steps for downloading, installing, and launching the application on your system.

In Chapter 4, you will find a detailed overview of the app interface, including descriptions of various sections where you can define mechanical, electrical, and geometric parameters. This chapter also contains illustrations to guide you through defining inputs that stimulate the HB. Additionally, it outlines the functionalities of the control center, where you can execute the model and export data.

Finally, Chapter 5 discusses the initial variables that populate the workspace backend, which are essential for initializing the model. It also includes an example demonstrating the use of default parameters.

This structure ensures that users have a clear, step-by-step guide to effectively utilize and navigate through HairBundleLab.

Chapter 2

System Requirements

This chapter details the essential system requirements needed to run HairBundleLab on your computer. Please ensure that your system meets the following criteria. Below is a list of the mandatory software required for the installation and operation of the application:

1. **MATLAB**: The application was developed using MATLAB 2024b [1], and requires a licensed MATLAB installation to run.
2. **MATLAB Symbolic Math Toolbox**: The application employs various symbolic functions to solve for the bundle geometry and perform nonlinear analysis, including nonlinear geometry. Therefore, the Symbolic Math Toolbox [2] must be installed in your MATLAB environment.
3. **MATLAB Parallel Computing Toolbox** (not mandatory, but recommended): This toolbox [3] is used in the background to run multiple simulations simultaneously, improving efficiency and reducing computation time. For instance, if you want to analyze responses to multiple force or displacement amplitudes, parallel computing can significantly shorten the total time required. The simulation time will vary based on the number of GPUs in your machine and the number of workers allocated for parallel computing in MATLAB. By default, the HairBundleLab has parallel computing turned ON, but this setting can be turned OFF if needed. If your MATLAB license does not include this toolbox, refer to Section 2.2 and Section 4.4 for more information.

2.1. What if I don't have MATLAB 2024b or later?

If you are using an older version of MATLAB, you can still use HairBundleLab, although some formatting may not render correctly. For instance, the symbol μ might appear as \mu, or subscripts/superscripts (e.g., $l1gs$ instead of l_1^{gs}) might not display properly. Functionally, the simulation should operate without issues.

Parallel computing has been supported in MATLAB since 2004, so any version released after 2004 can perform parallel computing. If you do not have the Parallel Computing Toolbox included with your license, you can still run the app by following the instructions briefly mentioned in Section 2.2 and detailed in Section 4.4. To check which toolboxes are included with your MATLAB installation, refer to Snippet 2.1 for the necessary code lines to input in the command window.

```
% Method 1: This will display a list of installed toolboxes. Look for
% Parallel Computing Toolbox       Version X.XX
ver -support

% Method 2: If it returns 1, the toolbox is installed.
license('test', 'Distrib_Computing_Toolbox')

% Method 3: If it lists a path, the toolbox is installed; otherwise, it will indicate "not found."
which parpool
```

Snippet 2.1: Three ways to check if the Parallel Computing Toolbox is installed. Additional methods are available but not listed here.

2.2. What if I don't have the Parallel Computing Toolbox?

If you do not have the Parallel Computing Toolbox, you can easily disable parallel computing within the application. Simply open the app and press the button marked to disable parallel computing. When the button changes color to red, as shown by , parallel computing is turned OFF. You can press the button again to re-enable parallel computing. For more detailed information on each button in the application, please refer to Section 4.4.

Chapter 3

Setting Up HairBundleLab

Once you have verified that your system meets all necessary requirements, you are ready to download and install HairBundleLab. If you are reading this document from a non-official source, make sure to go to our official lab website to download the installation file.

3.1. Downloading the Application

To download the application, visit the GitHub repository. Click on the 

button. From the drop-down menu, click on `Download ZIP` and save in your preferred folder. Once you unzip the downloaded file, open the folder named `app` and you can find the installation file named `HairBundleLab.mlappinstall` inside.

3.2. Installing the Application

Once you are inside `app`, double-click on `HairBundleLab.mlappinstall`. This will open MATLAB (if it is not already running) and prompt you to install the app under `My Apps`, as shown in Fig. 3.1. Click the `Install` button, and the installation should complete in under a minute.

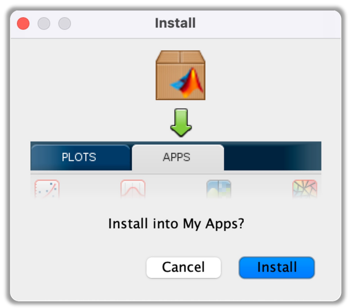


Figure 3.1: Installation window prompting the user to install HairBundleLab in the `My Apps` tab of MATLAB.

3.3. Opening the Application

Once installed, you can run HairBundleLab by following these steps, each time you wish to open the app:

1. Open MATLAB.
2. Navigate to the `APPS` tab in the MATLAB navigation bar, as highlighted by the red dotted box in Fig. 3.2.

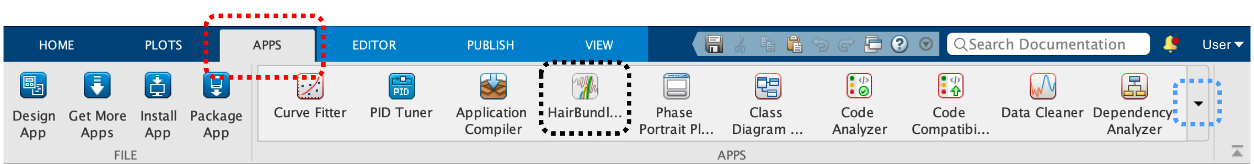


Figure 3.2: Illustration on how to execute HairBundleLab. Click on `APPS` (red dotted box) and navigate to `HairBundleLab` (black dotted box). If the app does not appear in the first row, use the drop-down arrow (blue dotted box) to expand the list.

3. Click on `HairBundleLab` to execute it, indicated by the black dotted box in Fig. 3.2. If you do not see it, expand the list of installed apps using the drop-down arrow located within the blue dotted box.

By following these instructions, you should be able to download, install, and run HairBundleLab without any issues. For further details on app functionalities and more specific instructions, please refer to the subsequent chapters of this document.

Chapter 4

Navigating through HairBundleLab

In this chapter, we will delve into the essential components of the application and provide you with a clear roadmap to efficiently navigate through its intuitive and segmented interface. Our goal is to equip you with the knowledge and tools required to simulate any HB with precision and ease. The application's interface is thoughtfully designed into three distinct columns, each serving a crucial purpose in the simulation process:

1. **Column 1**: Here, you can precisely define the mechanical and electrical properties of the HB, as shown in Fig. 4.1. Refer to Section 4.1 for details.

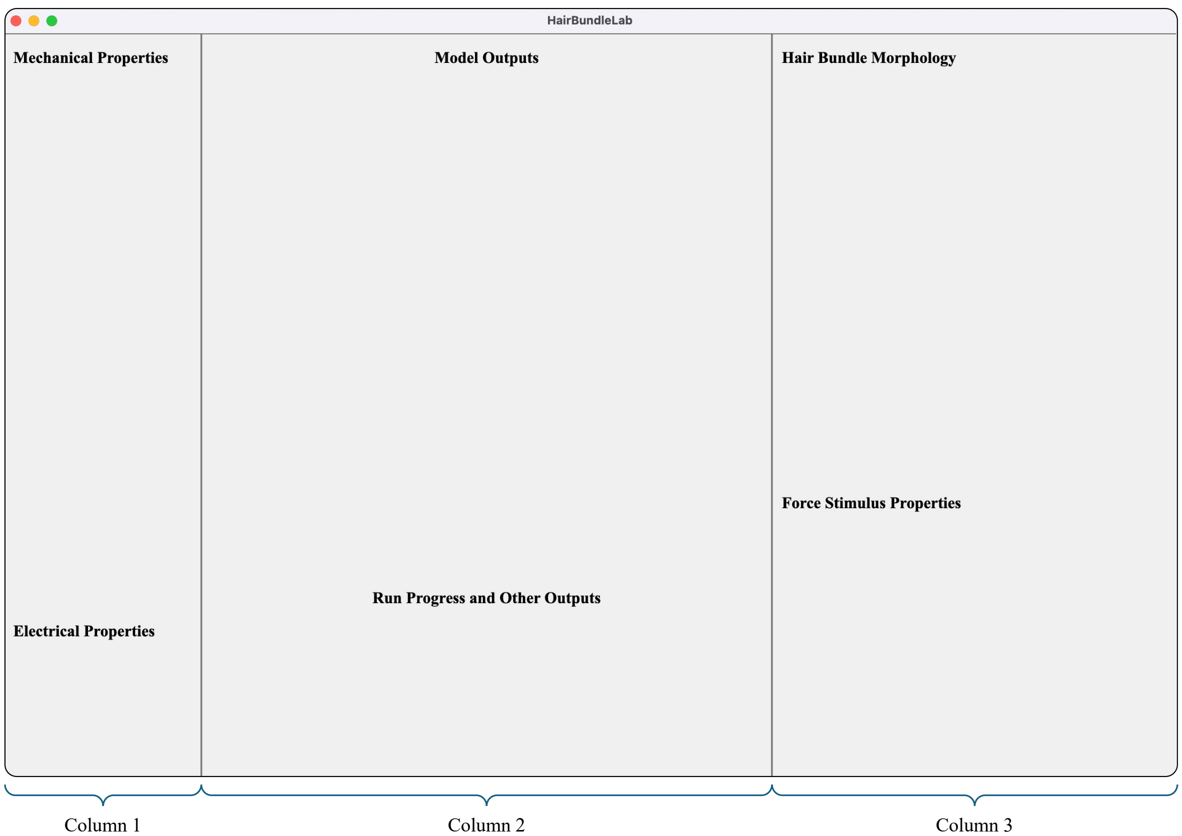


Figure 4.1: Illustration demonstrating the division of HairBundleLab into three columns, each highlighting the specific functionalities available within.

2. **Column 2**: It is the space where you can visualize the dynamic response and outputs of your model, offering insights into how a particular bundle would respond to a stimulus. You can also monitor the model's run progress in this section (see Fig. 4.1). Refer to Section 4.5 for details.
3. **Column 3**: This column, as shown in Fig. 4.1, is dedicated to defining the morphological properties of the HB. It also provides a visual representation of the HB, allowing you to see the physical structure and adjustments in real time before even running a simulation. Refer to Section 4.2 for details. Finally, this column contains the stimulus customization properties (refer to Section 4.3) along with a control center to perform various functions (refer to Section 4.4).

Each section of the application is designed to interact seamlessly, providing a holistic and interactive simulation environment. The detailed explanations and functionalities of each column, as well as guidance on effectively using them, are described in the upcoming sections of this chapter. We proceed in the following order: first column, third column, and finally the second column.

4.1. Defining Mechanical and Electrical Parameters

When you open the app, the interface displayed in Fig. 4.2 will appear, with the focus on the first (leftmost) column. This column is designed for you to define all the mechanical and electrical properties of the HB. Each parameter in this column is organized in a clear and user-friendly format that includes:

1. The parameter name.
2. A white input box for entering the parameter value.
3. The measurement unit for the parameter.

If you do not have specific values for some parameters, the app provides preset values that are displayed in the input boxes, as illustrated in Fig. 4.2.

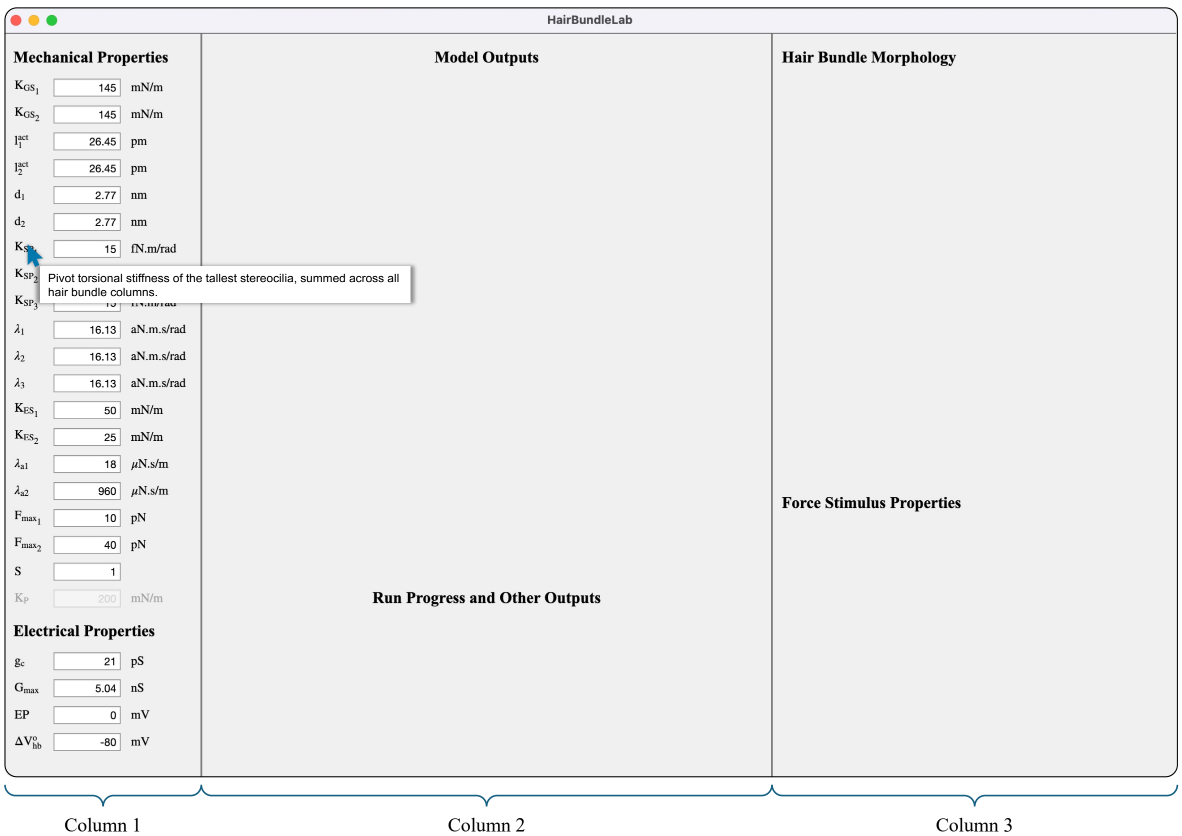


Figure 4.2: Description of the first column of the application, which displays various mechanical and electrical parameters that users can adjust to simulate their HB.

To gain a better understanding of what each parameter represents, simply hover your cursor over the parameter name. A tooltip will appear with detailed information about the parameter. For example, in Fig. 4.2, the blue cursor hovers over K_{SP_1}, and a tooltip appears with a description of K_{SP_1}. For a comprehensive list and definitions of all parameters, refer to Table 4.1.

At the bottom of the first column, you can specify the electrical properties of the HB. The three-row model uses a straightforward equation to estimate the MET current from the model outputs, requiring only a few electrical parameters to be defined. These key parameters are: single channel conductance (g_c), which represents the conductance of a single ion channel, combined channel conductance (G_{max}) which refers to the

Parameter	Description	Units
K_{SP_i}*	Combined torsional pivot stiffness of the i^{th} stereocilium across all columns	fN.m/rad
K_{GS_j}*	Stiffness of the gating spring between j^{th} and $(j+1)^{th}$ stereocilia summed across all columns in the HB	mN/m
K_{ES_1}	Stiffness of the extent spring associated with the adaptation complex in the tallest stereocilia UTLD across all HB columns	mN/m
K_{ES_2}	Stiffness of the extent spring associated with the adaptation complex in the middle stereocilia UTLD across all HB columns	mN/m
K_P	Rigid probe stiffness in the displacement-stimulated model	mN/m
λ_i*	Sum of the damping coefficients (representing viscous and shear effects) in the extracellular vicinity of the i^{th} stereocilium across all columns	aN.m.s/rad
λ_{a1}	Intracellular damping coefficient (equivalent of viscous and shear effects) in the vicinity of the tallest stereocilia adaptation complex combined for all columns	μN.s/m
λ_{a2}	Intracellular damping coefficient (equivalent of viscous and shear effects) in the vicinity of the middle stereocilia adaptation complex combined for all columns	μN.s/m
d_j*	Gating swing for the gate between j^{th} and $(j+1)^{th}$ stereocilia	nm
N_s	Total number of stereocilia in the HB	
N_{TL}	Number of tip links in the HB	
N_c	Number of transduction channels per tip link	
N	Combined transduction channels for each row pair in the HB ($= N_c N_{TL}/2$)	
k_B	Boltzmann constant	m^2.kg.s^{-2}.K^{-1}
T	Cell temperature	K
F_{max_1}	Maximum stall force of tallest stereocilia UTLD summed across all columns	pN
F_{max_2}	Maximum stall force of middle stereocilia UTLD summed across all columns	pN
l_j^{act*}	Resting extension of j^{th} gating spring	pm
S	Instantaneous feedback strength of cations on the adaptation complex	
G_{max}	Maximum conductance of the transduction channels at each LTLD	nS
ΔV_{hb}^o	Steady-state holding potential across the apical cell membrane of the OHC	mV
EP	Endo-cochlear potential at rest	mV
r_1	Radius of the tallest stereocilia (row 1)	nm
r_2	Radius of the middle stereocilia (row 2)	nm
r_3	Radius of the shortest stereocilia (row 3)	nm
l_{row_1}	Height of the tallest stereocilia (row 1)	μm
l_{row_2}	Height of the middle stereocilia (row 2)	μm
l_{row_3}	Height of the shortest stereocilia (row 3)	μm
b_{12}	Horizontal separation between rows 1 and 2	μm
b_{23}	Horizontal separation between rows 2 and 3	μm
l_1^{gs}	Height of the tip link insertion in row 1 from its pivot point when $\phi = 0$	μm
l_2^{gs}	Height of the tip link insertion in row 2 from its pivot point when $\phi = 0$	μm

Table 4.1: Model parameters and their description. * $i \in [1, 3]$ and $j \in [1, 2]$. Both i and j are integers.

maximum combined conductance of all ion channels in the HB between a pair of rows, endocochlear potential (EP), and the holding potential difference (ΔV_{hb}^o) across the bundle (between sub-tectorial space and the OHC). Note that G_{max} is related to g_c as $G_{max} = N_c N_{TL} * g_c/2$. Therefore, the value of g_c is automatically updated and the value that the model uses is G_{max}. These parameters are also listed and defined in Table 4.1 for your reference.

This intuitive setup ensures that you can easily input and understand the critical mechanical and electrical parameters needed for your simulation.

4.2. Defining HB Morphology

In this section, we will explore the top half of the third column, where you can define the geometric characteristics of the HB and visualize it when the tallest stereocilia stand vertically. As outlined in Section 4.1, the parameters follow a consistent structure: the geometric parameter name, an input box for its value, and its unit, as illustrated in Fig. 4.3. Similar to the first column, you can hover over each parameter name to reveal its definition, which is also provided in Table 4.1. For example, hovering over r_3 displays its definition, as shown in Fig. 4.3.

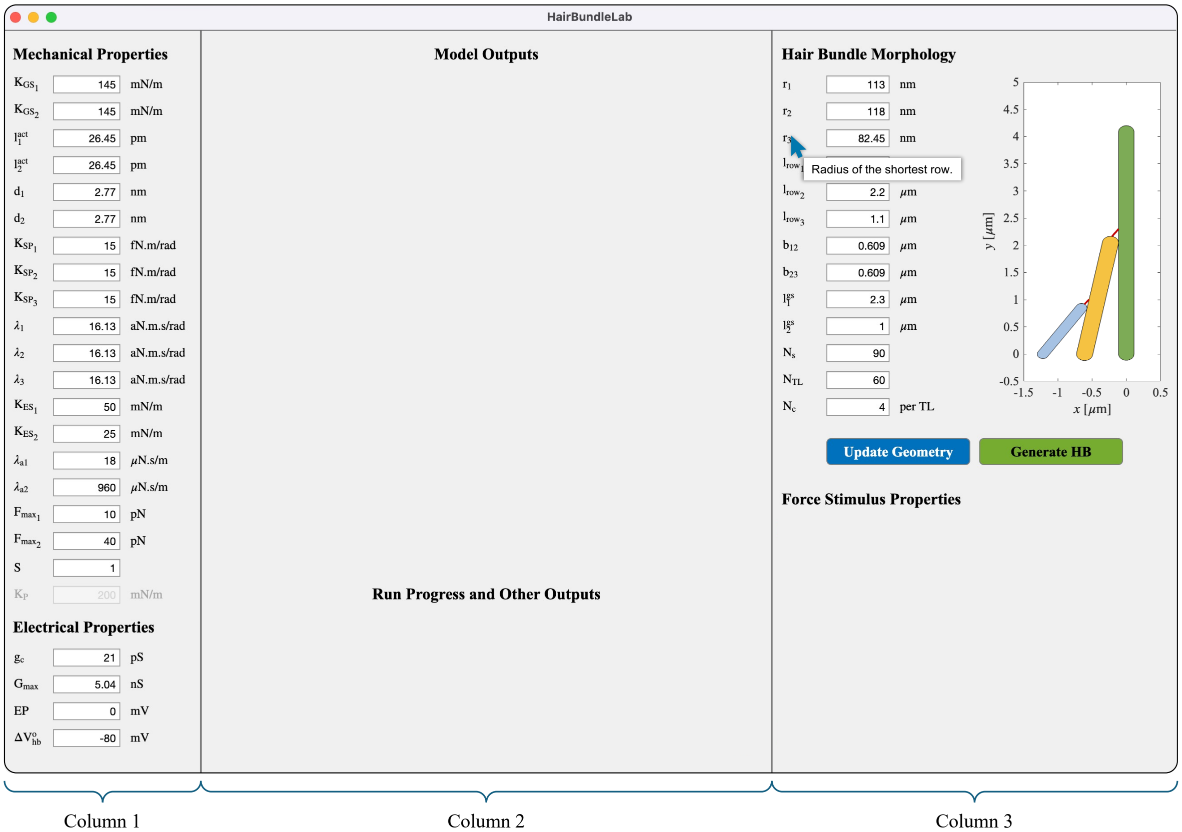


Figure 4.3: Description of the top half of the third column, where users define the geometric properties of the HB. The left side of this section allows users to specify parameters such as radii, lengths, and channel-related properties of the HB. On the right side, users can visualize the HB model based on the parameters defined on the left.

In the right half of this section, you can visualize the HB based on the geometric parameters defined on the left half. For clarity, the tallest row is depicted in green, the middle row in yellow, the shortest row in blue, and the maroon-colored links between each row-pair represent the tip links. When you first open the app, preset values populate each input box for the geometry, and the plot on the right will initially be empty. To visualize the HB with the default parameters, click on the green `Generate HB` button shown in Fig. 4.3.

> **⚠ Important Note**
>
> Due to the dimensions of the visualization window, the HB may sometimes appear skewed or flattened at the top when different parameters are defined. However, rest assured that in the actual design of the three-row model, all stereocilia are represented as rigid rods with circular tips. The visual discrepancies are purely due to space constraints within the visualization plot, and do not reflect any inaccuracies in the actual geometric parameters of the model.

You have the flexibility to design a different HB by modifying the geometric parameters on the left. For example, in Fig. 4.4, five parameters have been adjusted: r_1, r_2, r_3, l_{row_1}, and l_2^{gs}. On the left side of the figure, the bundle is generated using the default parameter values by pressing the `Generate HB` button, as indicated by the black cursor. The right side of the figure shows the bundle after altering the five aforementioned parameters. To visualize this new HB, you can either click on `Update Geometry` or `Generate HB`.

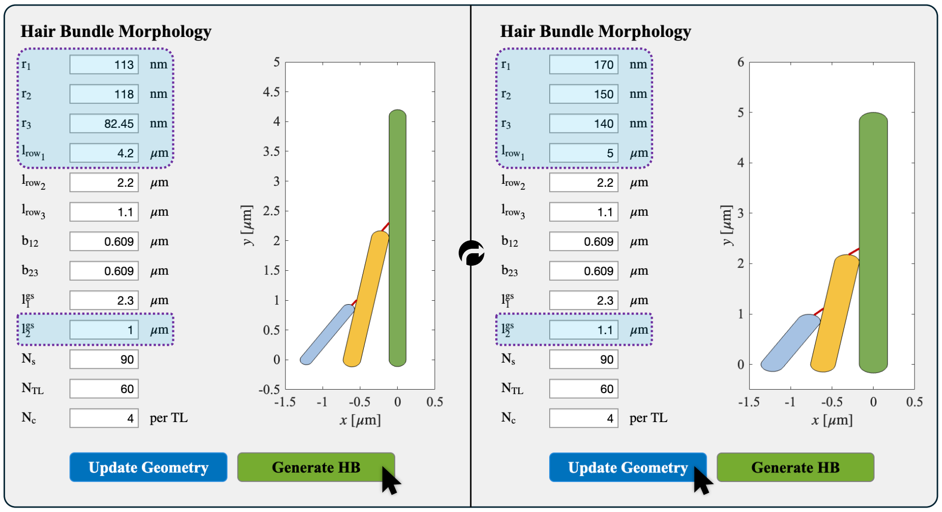


Figure 4.4: An example demonstrating how changes in geometric parameters from the first set (left) to the second set (right) affect the design of the HB. Use the left section to modify the parameters. To visualize the updated HB, click either `Update Geometry` or `Generate HB`.

It is important to note that up to this point, you have not run any simulations. This feature allows you to visualize the HB geometry before running time-domain simulations, helping to avoid infinite loops or errors due to inaccurate geometry. This pre-visualization step ensures that the geometric parameters are correctly set, providing a solid foundation for accurate simulations.

4.3. Defining the Input

This section focuses on the final aspect of user input: defining the input or stimulus to excite the HB. Here, you will learn how to specify the conditions for stimulating the HB using either a step-force or a step-displacement. A step-force, in experimental settings, is typically generated by a fluid jet. On the other hand, a step-displacement occurs when the HB is stimulated by a probe of a certain stiffness, with one end actuated by a piezoelectric device to induce displacement.

In the model, both step-force and step-displacement stimuli are characterized by a rise-time and an amplitude. The rise-time refers to the time it takes for the stimulus to rise from zero to its peak value, while the amplitude represents the peak value of the stimulus.

A screenshot of the app's interface can be seen in Fig. 4.5. The parameters for defining the input or stimulus are set up as follows: the parameter name, its unit (if applicable), an information tooltip represented by a question mark in a grey box, and a white input box for entering the value. To view detailed information about each parameter, you can hover over the grey question mark box, and a tooltip will appear, as shown in Fig. 4.5. By default, preset values are displayed in the white input boxes. These values can be easily adjusted by typing a new value into the respective input box.

By default, HairBundleLab is set to run a fluid-jet or force stimulus. Thus, the section is aptly titled `Force Stimulus Properties`. Here is detailed information about managing the force input parameters:

1. **Static Force**: You can define a static force to bias the HB either towards an excitatory resting rotation ($\phi > 0$) for a positive static force, or towards the inhibitory direction ($\phi < 0$) for a negative static force.
2. **Rise-Time**: This parameter defines the activation time of the applied force. A larger rise-time value results in a slower activation of the HB displacement, while a smaller rise-time achieves steady-state

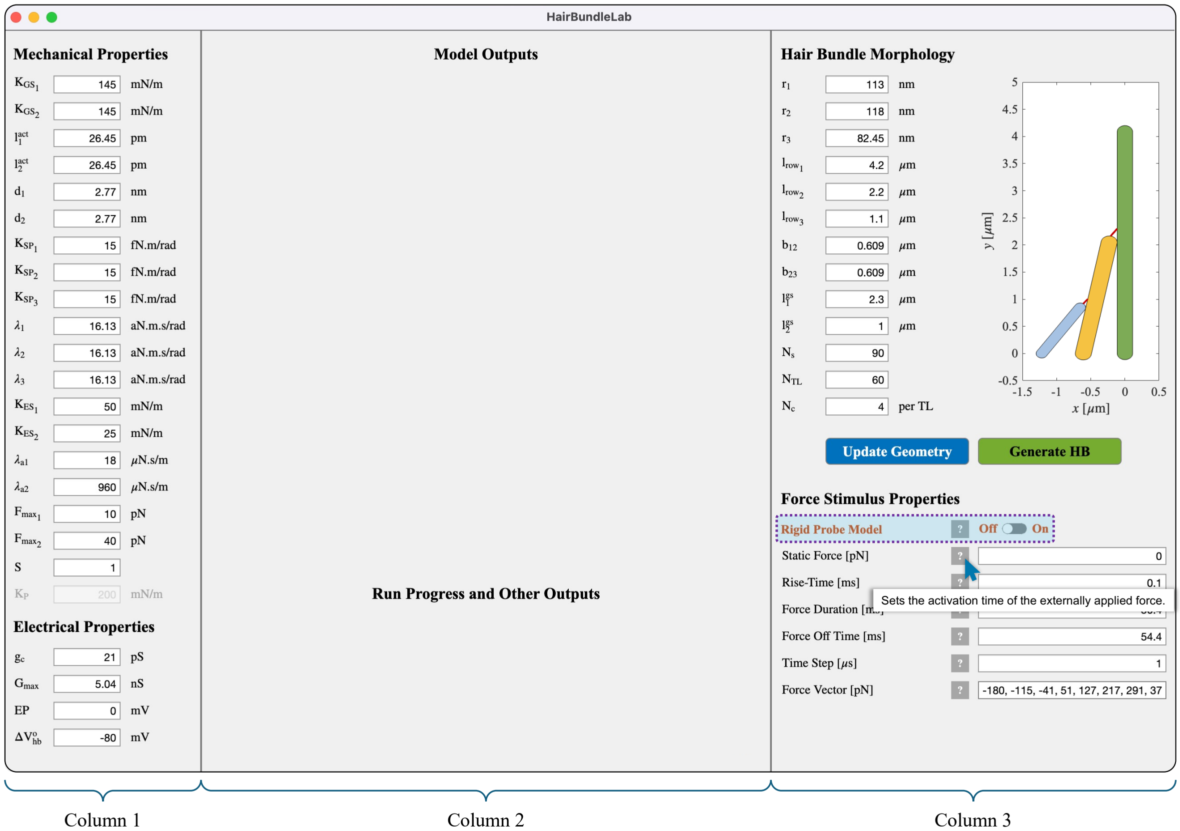


Figure 4.5: Description of the bottom half of the third column. In this section, users can define the input or stimulus properties, which are characterized by a rise time and amplitude. The other key parameters include the total response time, consisting of the initial zero-force response (55.4 ms), the user-defined duration of the applied input, and the user-defined time after the input is turned off. By default, the force input model is active. Users can switch to the nearly-rigid probe displacement input model by toggling the `Rigid Probe Model` switch to ON, as shown in the blue-shaded box.

faster.

3. **Force Duration**: Here, you specify the duration for which a non-zero force is applied to the system (not the static force, but the force defined in `Force Vector`).

4. **Force Off Time**: This parameter indicates the time the simulation will continue to run after the force has been turned off.

5. **Time Step**: This is the increment step for the simulation. It should be selected carefully in accordance with the rise-time to effectively capture the rise-time dynamics. For example, if the rise-time is 0.01 ms (10 μs), the time step should be lower than 10 μs to ensure accuracy.

6. **Force Vector**: This parameter allows you to define the force amplitude. You can either specify a single force (scalar) or a set of different force amplitudes to run multiple simulations in parallel. If you do not have the Parallel Computing Toolbox, you can still obtain responses for multiple forces from the same simulation by disabling parallel computation, as described in Section 2.2 and Section 4.4. To define multiple forces, you can separate them by commas, as demonstrated in Fig. 4.5, or input a vector in the format `start:increment:end`. For instance, with `start = -100`, `increment = -50`, and `end = 200`, the vector generated will be: `-100, -50, 0, 50, 100, 150, 200`. For comma separated, you can input the vector elements in any order, and the application will automatically sort them in ascending order.

⚠ Important Note

The total duration of the HB's response is the sum of three distinct time periods. Initially, there is a hidden time period of 55.4 ms before any force is applied. This pre-force interval allows the HB to reach a steady-state in response to the defined `Static Force`. Consequently, for each force amplitude specified in the `Force Vector`, the total response duration will be `(55.4 + Force Duration + Force Off Time)` ms. This ensures that the simulation captures the complete dynamic behavior, including the initial steady-state, the period of force application, and the post-force relaxation phase.

To run the model with excitation from a probe of a specific stiffness, simply enable this option by clicking on the switch located next to `Rigid Probe Model`. This switch is illustrated within the highlighted box with a dotted border in Fig. 4.5 and in Fig. 4.6 (top).

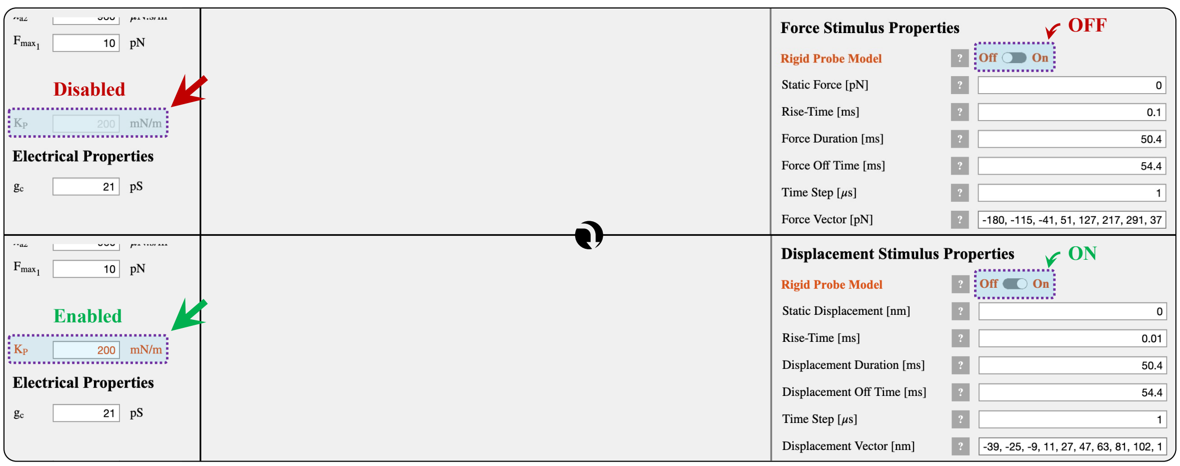


Figure 4.6: Illustration demonstrating how to switch between the force input (fluid-jet) model (top) and the displacement input (probe) model (bottom). By default, the force input model is active. To switch to the probe model, click the toggle next to `Rigid Probe Model`. This action updates all input parameter labels to replace the word "force" with "displacement" and activates an additional mechanical property in the first column: the stiffness of the probe, represented by the variable K_P (identified by a green arrow and ENABLED label). To revert to the force input model, click the toggle again to set `Rigid Probe Model` to OFF.

When the switch is toggled OFF, as shown at the top of Fig. 4.6, the probe stiffness setting (K_P) under `Mechanical Properties` in the first column is disabled. This is indicated by a blue box with a red arrowhead pointing towards it.

When you turn the switch ON (refer to the bottom of Fig. 4.6), the probe or displacement model is activated. This is signaled by the switch's ON position and the enabling of the probe stiffness setting, as marked by a green arrowhead.

Additionally, when the stiffness setting is enabled in the first column, the nomenclature for the stimulus changes to displacement-related terms. Rest assured that the meaning of each setting remains the same as previously described, except that you now enter displacement values instead of force values. Consequently, `Static Force` changes to `Static Displacement`, and `Force Vector` changes to `Displacement Vector`, which you define in the same manner as you would for a force. Like with the force stimulus, preset values for these quantities are displayed.

⚠ Important Note

The units for a force input are picoNewtons (pN), and the units for a displacement input are nanometers (nm).

4.4. Control Center

The control center is the section from which you can run the model and perform various functions, as detailed below. Fig. 4.7 illustrates these functions within a blue-colored box, located beneath the force or displacement stimulus properties section.

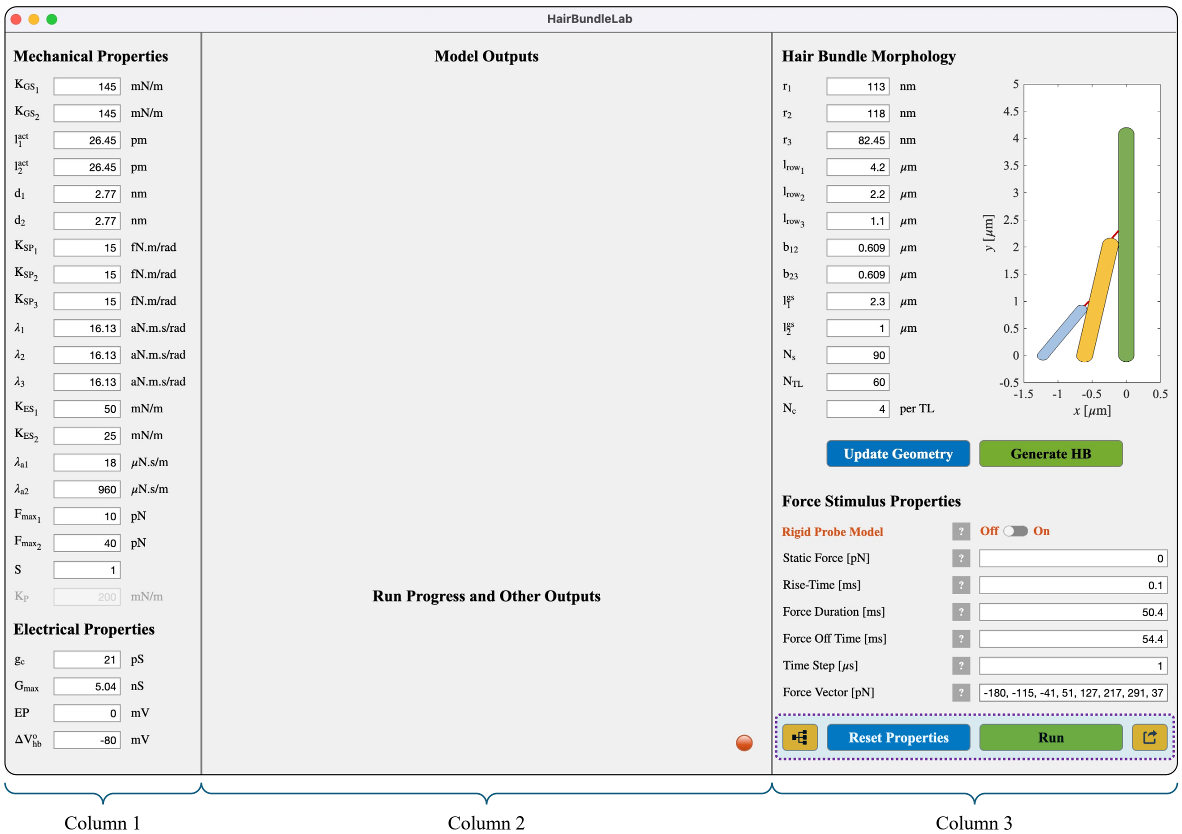


Figure 4.7: Description of the control center located below the input properties in the third column. The control center provides key functionalities for managing the simulation. The leftmost yellow button allows users to toggle between parallel and non-parallel computing modes. To reset the altered model to its default parameters, click the blue `Reset Properties` button. Note that this action also sets the `Rigid Probe Model` toggle to OFF if it is currently ON. To run the simulation and obtain the time-domain response, use the green `Run` button. Finally, the rightmost yellow button, located within the blue-shaded box, enables users to export all simulation outputs.

The following functions can be performed from the control center:

1. **Run the Model**: To run the model, press the Run button. When the model has successfully started and is running, it will be indicated by the lamp shown in Fig. 4.7 in the middle column. A red lamp signifies that nothing is running. After pressing the Run button, the lamp will turn green. Once the simulation is complete, the lamp will revert to red.

2. **Reset All Properties**: By pressing the Reset Properties button, you can restore all changes made since opening the application to their default values. This will reset all mechanical, electrical, and geometric parameters to preset values and revert the model to force-stimulated mode, turning off the rigid probe model if activated. Upon pressing the Reset Properties button, a dialog box (as shown in Fig. 4.8) will appear, asking you to confirm the reset to prevent an irreversible change if Reset Properties button were accidentally clicked.

3. **Export Data**: You can export data in `.mat` format using the button. Pressing this button will open a dialog box (shown on the left in Fig. 4.9). You can specify a file name (`filename`) without spaces, and it will save the file as `filename.mat` upon pressing `Save` in the open directory of MATLAB. Note that if no simulation has been run, there will be no data to export. In such a case, pressing the button will

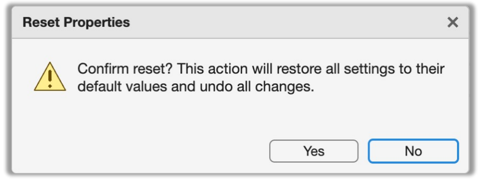


Figure 4.8: Reset properties dialog box. This dialog box appears when the `Reset Properties` button in the control center is pressed, prompting confirmation for the reset action because resetting is irreversible. It will restore all parameters to their default values, equivalent to reopening HairBundleLab for the first time. Click `Yes` to confirm the reset or `No` to cancel if no resetting is needed or if the button was clicked accidentally.

display an error message, as shown on the right in Fig. 4.9.

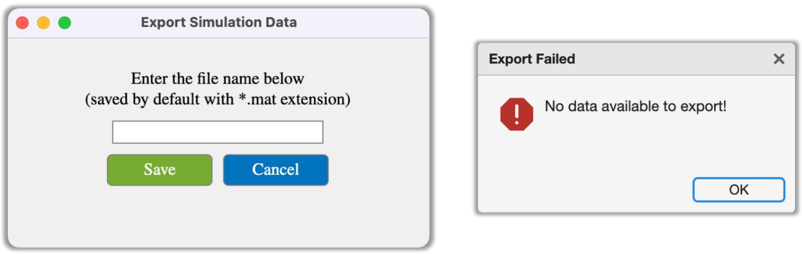


Figure 4.9: Description of the two dialog boxes that may appear when the export button is pressed. The dialog box on the left appears if a simulation has been run, indicating that data is available in the MATLAB workspace for export. In this case, you can specify a filename, which will be saved with a `*.mat` extension in the working directory upon clicking the green `Save` button. This action can be canceled by clicking the blue `Cancel` button. The dialog box on the right appears if no data is available in the MATLAB workspace, indicating that there is nothing to export.

Here is a list of information that is saved in `filename.mat`:

- **`t`**: This variable contains time increment data in seconds. It is an $m \times n$ matrix, where m represents the number of values in the `Force Vector`, and n represents the total time steps calculated as `(55.4 + Force Duration + Force Off Time)*1000/(Time Increment) + 1`.
- **`xHB`**: This matrix stores HB displacements in meters corresponding to each `t`, with dimensions $m \times n$. To retrieve the bundle rotation ϕ from `xHB`, use the following relation: $\phi = \sin^{-1}(\texttt{xHB}/l_{row_1})$, where `xHB` and l_{row_1} must have consistent units.
- **`iMet`**: This matrix stores the MET currents in Amperes corresponding to each `t`, with dimensions $m \times n$.
- **`la1`**: This matrix stores the displacements of the first adaptation motor (l_{a1}) in meters corresponding to each `t`, with dimensions $m \times n$.
- **`la2`**: This matrix stores the displacements of the second adaptation motor (l_{a2}) in meters corresponding to each `t`, with dimensions $m \times n$.
- **`xStiff`**: This vector stores the steady-state displacement of the HB in meters, computed at `(Force Duration - 1.6)` ms. It is an m-tuple vector.
- **`kStiff`**: This vector stores the stiffness (K_{HB}) of the HB corresponding to each force or displacement amplitude and `xStiff`. It is an $(m-1)$-tuple vector, calculated using the formula:

$$K_{HB} = \frac{F_o^{k+1} - F_o^k}{X_{hb}^{k+1} - X_{hb}^k} \tag{4.1}$$

for force stimulus, and

$$K_{HB} = K_P \frac{X_o^{base,k+1} - X_o^{base,k}}{X_{hb}^{k+1} - X_{hb}^k} - K_P \tag{4.2}$$

for displacement stimulus, where F_o is the amplitude defined in the Force Vector, X_o^{base} is the amplitude defined in the Displacement Vector when the Rigid Probe Model is ON, and $k \in [1, m]$. Other parameters are defined in Table 4.1.

- **xAct**: This $1 \times m$ vector stores the HB displacements in meters when the MET current peaks for each stimulus amplitude.
- **peakVal**: This $1 \times m$ vector contains the normalized absolute peak values of the MET currents for each stimulus amplitude. The peak values are derived from the maximum absolute values of the MET currents within the first 20 ms of the force onset and are normalized by the highest peak value.
- **miscOutputs**: This is a MATLAB structure that contains miscellaneous outputs, such as **hbSensitivity**, **po1Eqb**, and **po2Eqb**. **hbSensitivity** stores the HB sensitivity in nm^{-1}, calculated from the maximum slope of the activation curve. **po1Eqb** stores the resting probability of transduction channels at the tip of the middle row. **po2Eqb** stores the resting probability of transduction channels at the tip of the shortest row. To extract an output from the structure, use miscOutputs.<variableName>. For example, to access po1Eqb, type miscOutputs.po1Eqb.

4. **Parallel Computing Switch**: Some versions of MATLAB and/or certain licenses may not include the Parallel Computing Toolbox. To run HairBundleLab without this toolbox, you need to disable the default parallel computing option. To do this, locate the button in the control center, as shown in Fig. 4.7. Simply press the button to turn it OFF. When the button changes to , you can run the model without encountering any issues. To turn parallel computing back ON, press button.

4.5. Output Window

This is the final section of this chapter, where you will learn about the middle column shown in Fig. 4.10. This area is dedicated to visualizing various simulation outputs.

The upper half of this column features a 4×4 grid with descriptions for each plot:

1. **HB Displacements**: In the (1,1) block, this plot shows the time course of the HB displacements in nanometers. After the simulation finishes, you will see the displacement vs. time plot for all amplitudes in the Force Vector or Displacement Vector shown in black, superimposed on each other.
2. **MET Currents**: In the (1,2) block, this plot displays the time course of the MET currents in nanoAmperes. Similar to the HB displacements, the current vs. time plot for all amplitudes is shown in black, superimposed on each other.
3. **Activation Curve**: In the (2,1) block, you can visualize the activation curve. The peakVal (dimensionless) is plotted on the y-axis against xAct (nm) on the x-axis. For this plot to form a line, at least two elements in the Force Vector or Displacement Vector are required. With only one element, it will be displayed as a point.
4. **HB Stiffness**: In the (2,2) block, this plot visualizes the HB stiffness (K_{HB} in mN/m) from kStiff against xStiff (nm) on the x-axis. To form a line on this plot, at least three elements in the Force Vector or Displacement Vector are needed due to the way the HB stiffness is defined in Eqs. 4.1 and 4.2. With only two elements, it will be displayed as a point.

The lower half of this column is where you can monitor the run progress in real time after initiating the run. On the left side, there are five message boxes that update after each critical step is completed:

1. **Geometric relations computed**: This message appears once the model computes nonlinear symbolic equations for various geometric quantities, based on user input under Hair Bundle Morphology in the third column.
2. **Probability relations computed**: This message is displayed after computing the channel open probability functions, which depend on the nonlinear geometric relations. These symbolic equations are converted to MATLAB functions using matlabFunction in the backend.

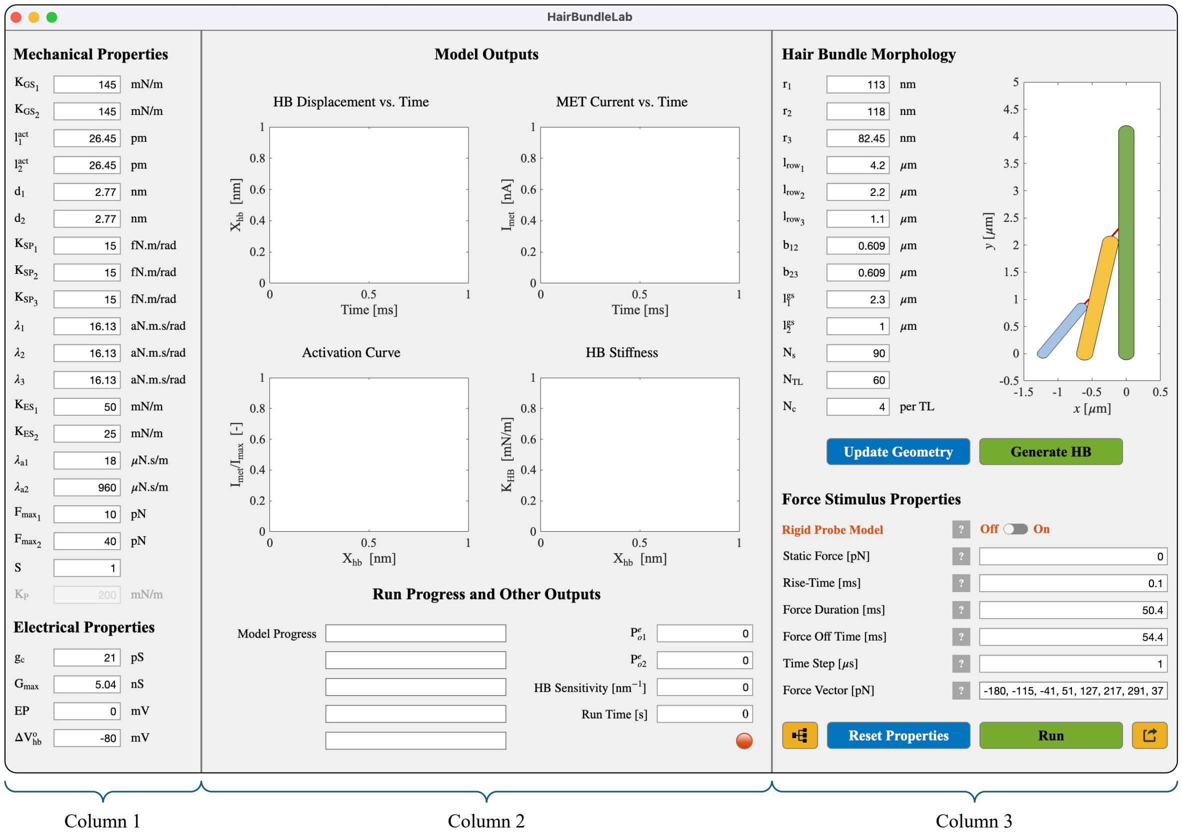


Figure 4.10: Description of the elements of the middle column. In the upper half of this section, users can visualize the model outputs across four different plots: HB displacements (first plot window - (1,1) block), MET currents (second plot window - (1,2) block), the activation curve (third plot window - (2,1) block), and HB stiffness (fourth plot window - (2,2) block). In the lower half, users can track real-time simulation progress, with five messages displayed under the `Model Progress` section, each indicating when an important step has been executed or initiated. On the right side of the bottom half, users can view miscellaneous outputs, such as the resting open probabilities of the channels, HB sensitivity, and the total runtime of the model.

3. **Initiating the Runge-Kutta solver**: This indicates that the model has begun using the fourth-order Runge-Kutta method to solve three coupled first-order differential equations.
4. **Finished numerical solution**: This message appears when the Runge-Kutta numerical solution is computed for all time steps.
5. **Plotting the results.**: This message indicates that the model is now post-processing the results for visualization in the top half of this column.

The right side of the lower half displays miscellaneous outputs discussed in Section 4.4. It includes the total run time for the simulation in seconds, shown in the white display box next to `Run Time [s]`.

> **⚠ Important Note**
>
> If you are utilizing parallel computation and the Parallel Computing Toolbox is not initially running on MATLAB, the `Run Time` will include both the time required to start the toolbox and the actual runtime of the model. For subsequent simulations, the `Run Time` will reflect the true runtime.

This concludes the tutorial for HairBundleLab. Now you know how to navigate through and use its features effectively, enabling you to simulate your own HB by defining specific mechanical, electrical, and geometric properties for two distinct types of stimulations.

Chapter 5

Running HairBundleLab

In this chapter, we will walk you through an example using the preset values in the model for a force-stimulated HB. Additionally, toward the end of the chapter, we will provide some miscellaneous information about the model to enhance your understanding.

5.1. Starter Variables

When you open the app as described in Section 3.3, it will create various preset variables in your MATLAB workspace. These variables are tabulated in Table 5.1.

Variable	Description
`duration`	Input value from the `Force Duration` or `Displacement Duration` field
`Fo`	Scalar or Vector input from the `Force Vector` or `Displacement Vector` field. For the `Displacement Vector`, `Fo` represents the force, which is calculated by multiplying the displacement values (in meters) by the probe stiffness (in N/m).
`Fst`	Input from the `Static Force` or `Static Displacement` field
`func`	MATLAB structure that contains all the symbolic functions converted to MATLAB functions using `matlabFunction`
`geom`	MATLAB structure that contains all the morphological properties, including radii and lengths
`h`	Time-step increment for the Runge-Kutta solver
`l1`	Symbolic expression for the length of the first tip link (between the tallest and the middle stereocilium)
`l2`	Symbolic expression for the length of the second tip link (between the middle and the shortest stereocilium)
`param`	MATLAB structure that contains all the mechanical and electrical parameters of the model
`shutDuration`	Duration of the simulation following the removal of the force
`tauF`	Rise-time of the stimulus

Table 5.1: Starter variables initialized in the MATLAB workspace upon opening HairBundleLab.

5.2. An Example

In this section, we will work with the preset parameters that are displayed upon opening the app. Fig. 5.1 presents the model outputs for these parameters.

The `Force Vector` was configured with 13 distinct values, resulting in 13 unique plots of HB displacements displayed in the (1,1) location of the 4×4 grid after you press the `Run` button. Similarly, 13 different MET currents are shown in the (1,2) location of the grid.

The activation curve is displayed at the (2,1) location, and the displacement-stiffness relationship is plotted at the (2,2) location, as described in Section 4.5. The lower section of the middle column shows the model progress outputs, which were detailed in Section 4.5. Additionally, you can see the resting channel open probabilities corresponding to a zero static force. Since this was the first instance of using parallel computing, the runtime is approximately 60 seconds. If parallel computing has already been initialized in MATLAB prior to running the simulation, the runtime should be reduced to 20 seconds or less for all 13 force amplitudes.

You can make interpretations from the response. For instance, for this simulation, a rise-time of 0.1 ms was used with a force input, indicating that the fast adaptation in the system is obscured. If you enable the `Rigid Probe Model`, you will observe that the MET currents exhibit a distinct peak characterized by fast

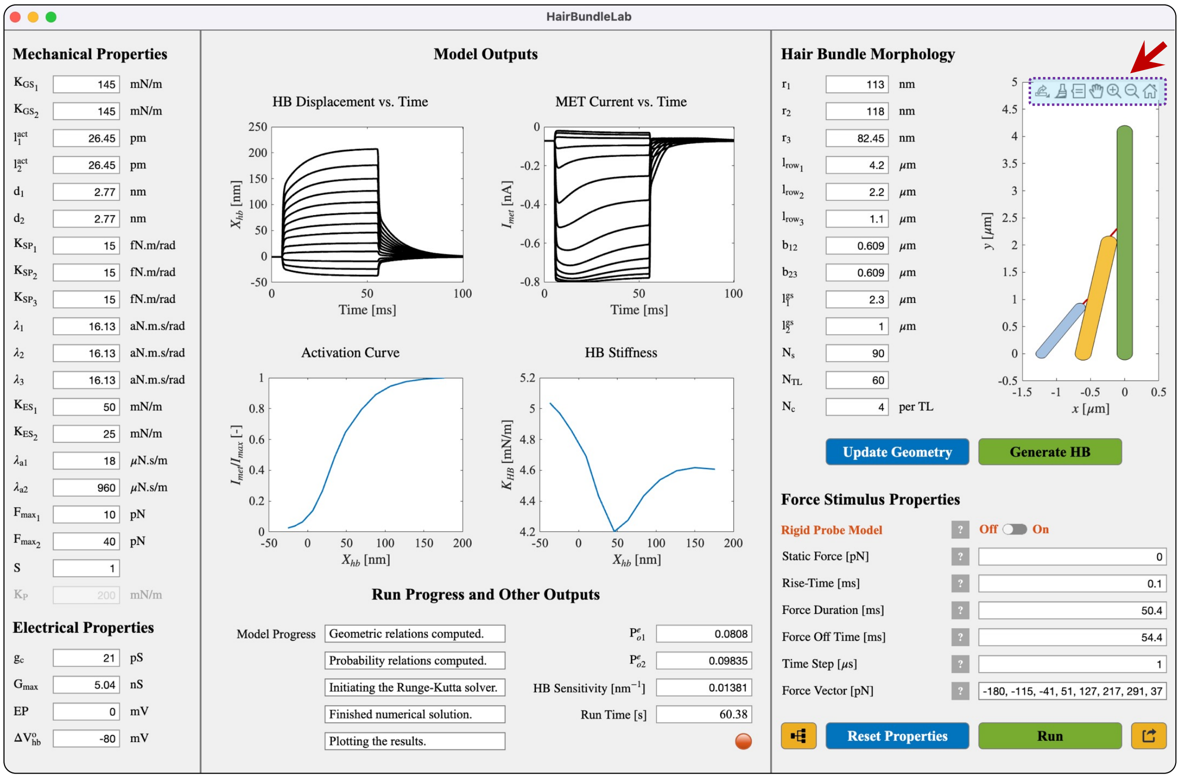


Figure 5.1: Illustration of a full working example using preset parameters. After pressing the `Run` button, the model progress updates in real-time, with the lamp turning green as the simulation progresses. Once the simulation is complete, the miscellaneous outputs appear, and the lamp changes to red. Additionally, the four output graphics are displayed to visualize the bundle's response. The blue-shaded box, indicated by a red arrowhead, shows the figure management options provided by MATLAB, offering features to copy, zoom, pan, and rotate the graphics.

adaptation, followed by a slow decay representing slow adaptation. This behavior is due to the stiffening of the HB resulting from the probe stiffness.

5.3. Miscellaneous Information

In this section, information is provided about various aspects related to running simulations on HairBundleLab.

1. **Simulation Cannot Be Stopped Midway**: Parallel computing assigns different tasks to each worker in MATLAB. For example, if MATLAB is using eight workers and you have 13 elements in your `Force Vector` or `Displacement Vector`, MATLAB assigns eight of the 13 tasks to the available workers. Once those tasks are completed, the remaining five tasks are assigned to the workers. However, once tasks have been assigned, there is no way for HairBundleLab, a MATLAB application, to communicate with the `parpool`. Consequently, there is no button to stop a running simulation midway. The only way to stop the simulation if it encounters an error not displayed in the MATLAB command window or if it is taking too long is to close the application and reopen it.

2. **Typical Runtimes**: In Fig. 5.1, the time-step was 1 μs and the total duration was 160.2 ms (comprising 55.4 ms, 50.4 ms, and 54.4 ms segments). For this computation, simulating a 1 ms response to one force input took about 10 ms on a MacBook Pro with an M1 chip, 8 GB RAM, and 8 GPU cores (equivalent to 8 MATLAB workers). In comparison, performing the same computation on the Great Lakes High Performance Computing cluster, which used 32 workers, took only 0.5 ms. Therefore, if you have more GPU cores and can assign more workers on your local MATLAB software, the simulation runtimes for a larger number of force/displacement amplitudes will be significantly faster.

3. **Additional HairBundleLab Tools**: The app provides more exportable outputs. These capabilities are

enabled by MATLAB's design structure for developing applications. As shown in Fig. 5.1, inside a blue translucent box atop the HB visual in the third column (red arrowhead), figure management options are available. You can export graphics, copy them, zoom, pan, and rotate them as needed to visualize the data in your preferred manner. All five graphics in HairBundleLab have this functionality.

With that, we conclude this documentation. With the help of this comprehensive guide, you are now equipped to effectively utilize HairBundleLab for simulating and analyzing your hair bundle models. Thank you for using HairBundleLab, and happy simulating!

Bibliography

[1] The MathWorks Inc. *MATLAB version: 24.2 (R2024b)*. Natick, Massachusetts, United States, 2024. URL: https://www.mathworks.com.

[2] The MathWorks Inc. *Symbolic Math Toolbox version: 24.2 (R2024b)*. Natick, Massachusetts, United States, 2024. URL: https://www.mathworks.com/products/symbolic.html.

[3] The MathWorks Inc. *Parallel Computing Toolbox version: 24.2 (R2024b)*. Natick, Massachusetts, United States, 2024. URL: https://www.mathworks.com/products/parallel-computing.html.