Worksheet 4: Heart Modeling - Simulink and Testing

Principles of Modeling for Cyber Physical Systems – Module 2 – Medical CPS

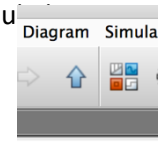
Instructor: Madhur Behl, madhur.behl@virginia.edu

Due date: 11-19-2020, by 11:59pm

In this worksheet you will create a simple heart model, such as the one we saw in class, from the components in a model library. The models will be created in Simulink/Stateflow. Simulink and Stateflow are part of the Matlab software suite.

General instructions

In all the models you create below, create a scope that shows the signals of interest. You add a scope by drag-and-drop from the Library Browser in Simu



If a scope is missing, the question will not be graded. When you create your scope, on each graph, Right-click -> `Configuration Properties` -> `Display`, and edit the name of the graph so we can tell what signal you're displaying without having to go back and forth between the scope window and the model editor. Please use expressive names, they're free.

Unless otherwise specified, the total simulation duration should be 10 seconds. See tip below on step-size.

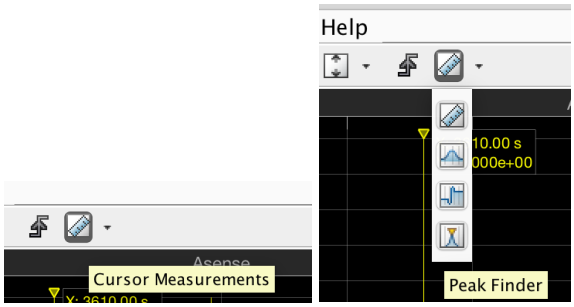
To every question below, create and submit an entirely new Simulink model. The name of the Simulink model should be the name listed after **Solution Name.** E.g. Question 1 solution should be named NPNAntegradeOnly.slx

All models will re-use the blocks in model_library.slx, but otherwise they are independent. Put all your models in a folder, compress it and upload the compressed folder, with your computing ID as the name.

Useful Tips

Remember to use the debugger and simulation stepping! (the left-pointing arrow next to the Play button). They are useful tools for seeing step-by-step progress of your model.

In the scope, you can use Measurements to get rulers that tell when events are happening. In particular, Peak Finder is helpful to quickly find all the events and their times without browsing.



Using components in model_library.slx: Start Matlab. To instantiate a component from model_library.slx, double-click on the latter – it will take a few seconds to open the first time. Then click File → New Model → Blank model. Now you use "copy-paste" to instantiate components: click on that component, Ctrl+C (or Command+C on Mac), and in your empty model, hit Ctrl+V (Command+V). It is good practice to re-name the instances so they have meaningful names. To re-name, Right-Click -> Rename.

When you try to modify a copied component, Simulink might protest and ask whether you want to "unlink" the component from the library. Do that.

Start MATLAB and open model_library.slx by double-clicking on it. model_library.slx contains several types of node and path automata, implemented as Simulink blocks (with Stateflow charts inside them).

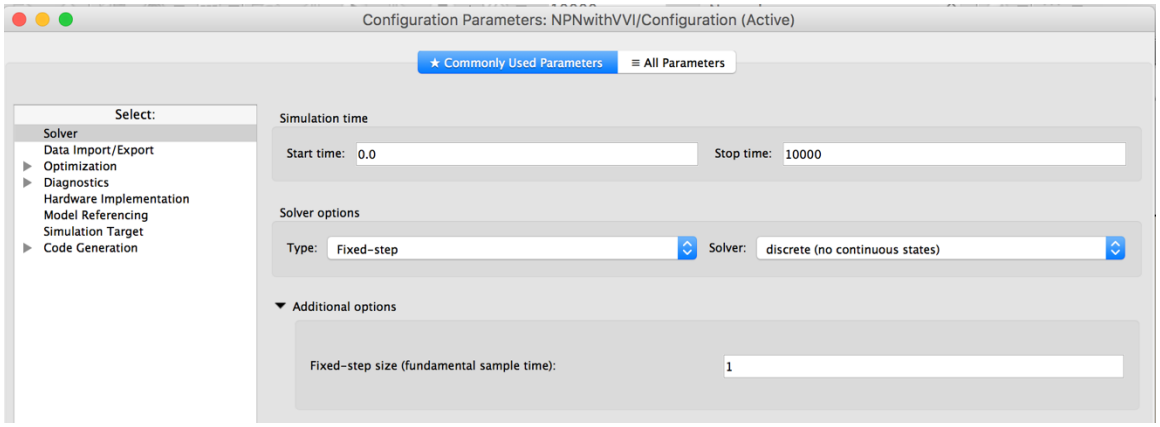
1. **Normal Sinus Rhythm (NSR) [10%]**
   Solution Name: NPNAntegradeOnly.slx
   A healthy human has a heart rate of around 60 beat per minute. During NSR, the electrical events originate from the SA node and travel to the ventricles with around 150ms delay. (Note that these numbers can vary from person to person, they are just rough averages).

   Use the blocks `NodeLongERP` and `AtoVPath` in model_library.slx to create an NPN heart model (as we saw in class), and adjust the parameters of the blocks so that the model can mimic the electrical event generation and conduction between the atria and the ventricles during NSR.

   Details: we want a heart rate of 60bpm, a conduction delay of 150ms (give or take a few ms) from atria to ventricles. Set the basic sampling time (`FixedStepDiscrete` setting) to 1. Display the atrial and ventricular events on the scope (those are `Act_path` from the two node automata).
   Make sure that all ports of `AtoVPath` are connected. Remember that a conduction path is bi-directional! So, it can be stimulated by the node "above it" or by the node "below it".

   Hint: you will connect three block instances together (obviously, since you're creating an NPN model). Remember that if we want the SA node to be the only one that fires, then its Rest period must be shorter than any other Rest period.

A note about time steps: The basic simulation step-size can be set by clicking on the blue `FixedStepDiscrete` on the bottom right of the Simulink window. (If you don't see that in your version of Matlab, click on Simulation → Model Configuration Parameters → Solver → Fixed-step size.)



Suppose you fix that to 100. This means that doing one action (like decrementing a counter) takes 100 time steps. We think of 1 time step as being 1ms. This is just a convention that we are using, it doesn't have any physical reality. The total simulation time (in the middle of the Simulink window near the top) is given in time steps. So if you want 10sec = 10,000ms = 10,000 time-steps, you set that to 10,000. Finally, the `clock' that is used in the guards of `NodeLongERP` is really just a counter `Rest_cur`, initialized to `Rest_def`. (Similarly, for `ERP_cur`, etc). So if we want `Rest_def` to be 1000ms like for NSR, then we should set `Rest_def` = desired duration/`FixedStepDiscrete` = 1000/100 = 10. Similarly, for the other counters.

2. **Sinus Bradycardia [5%]**
   Solution Name: Bradycardia.slx
   During Sinus Bradycardia, the SA node is not able to generate electrical events fast enough. Adjust the parameters of the heart model you just created to mimic the electrical behavior of Sinus Bradycardia, with a heart rate of 30bpm.

3. **Sinus Tachycardia [5%]**
   Solution Name: Tachycardia.slx
   SA node generates frequent atrial events so that the heart rate is around 120bpm

4. **AV Block [20%]**
   Solution Name: AVblock.slx

   The Atrio-Ventricular (AV) node blocks fast atrial events so that the corresponding ventricular rate remains within a normal range.
   Use blocks `AVNode`, `NodeLongERP` and `AtoVPath` to create a model which has a fast atrial rate, but such that the AV node is blocking 1 our of every 2 atrial beats, thus keeping a 2:1 ratio between atria and ventricles.

The structure of the model is N-P-N-P-N.
The total simulation duration should be 10seconds.
When we look at the scope and simulate, we expect to see 1 ventricular beat for every 2 atrial beats.
Hint: for the fast atrial rate, you can start by re-using the NPN model you created earlier to simulate tachycardia. Then add the `AVNode` block to it.
Hint: remember that a conduction path is bi-directional: it can conduct *antegrade* from atrium to ventricle, or *retrograde* from ventricle to atrium. So, it can be stimulated by the node "above it" or by the node "below it". Make sure to connect each path in your model to both nodes adjacent to it.
Hint: If a node can be stimulated from two different sources (like the AV-Node can be stimulated by either the path above it (that connects it to Atrium antegradely), or by the path below it (that connects it to the ventricle retrogradely), then you can use an OR logical block to pass these two sources of stimulation.
Hint: for the AV node to drop some beats, it must receive them faster than allowed by its ERP duration. That is, we need the `Rest_def` of the atrium to be less than the ERP of the AV node.
Hint: Keep in mind that the `AVNode` has its own rest period, and we don't want it to fire (transition to ERP location) before it receives the stimulation from the Atrium. So set the AVNode `Rest_def` to be larger than `Rest_def` of atrium+conduction delay of first path.

5. **AV delay [15%]**
Solution Name: DelayedAV.slx
A normal AV node introduces some delay in the conduction (a delay that is significant compared with the time it takes the waveform to traverse myocardial tissue from atria to ventricles). This is inherent to the heart's operation, since it allows the ventricles to fill with blood before they contract. Some patients have an AV node that causes excessive delay between atrial events and ventricular events, causing inefficient pumping of the heart.
The `AVNode` block we used in the previous question did not actually introduce any delays in the conduction: as soon as it received an `Act_node` action in location `Rest`, it transitioned to location temp and sent an `Act_path` action. So, you will now modify the `AVNode` to introduce a delay, which we will call a Dwell time.

Details: add a new location to the Stateflow chart of `AVNode`, call it Dwell. The objective of Dwell is to simply let some time pass before firing `Act_path`. Call the variable that counts down how much time to let pass `Dwell_time`, and call its default (initial) value `Dwell_def`.
- when should the chart enter the Dwell location? When should it move on to location temp?
- make `Dwell_def` an input signal, which is set by a Constant block, similar to `Rest_def`. Set it to 300.

- make sure that the `Act_path` action is issued at the correct transition…

6. **Premature Ventricular Complex [15%]**
   Solution Name: NSRwithPVC.slx
   Use the `Node_a2` block to model a PVC.
   Details:
   - you will start with the NPNPN heart model you created in above questions. If you weren't able to create the NPNPN model, then start with the NPN model in question 1. Make sure this is simulating NSR.
   - instantiate the node automaton `Node_a2` from model_library.slx and connect it to the heart model. Let's call this new node automaton instance `PVCNode`.
   - Because `PVCNode` is meant to be a source of *ventricular* events, the `Act_path` of PVCNode should be connected to `Act_path2` of the bottom path. Of course, there is already something connected there, so use an OR block.
   - a PVC is a spontaneous event, it doesn't get triggered by anything else. Should anything be connected to `Act_node` of `PVCNode`?
   - what should the `Rest_def` of `PVCNode` be set to, to guarantee that we observe a V event that propagates back up to the atria? Keep in mind the various ERPs it encounters along the way.
   We only need to observe one V event propagate retrogradely, not a cascade of them. So if we see that on your scope, we're happy. If we're happy, you're happy.

Pacemaker Model Simulink

1. **Observe VVI pacemaker [10%]**
   Load and run NPNwithVVI.slx. This connects a VVI pacemaker (which only observes and paces in the ventricles) to the NPN heart model. Do you see the PM pacing? Why, or why not? Explain how the `LRI_def` value of 1300 prevents any pacing.

2. **Debugging the VVI [10%]**
   Solution Name: NPNwithVVIBrady.slx.
   Change NPNwithVVI.slx to simulate Bradycardia, such that the atrium fires every 1800ms (after the first event). What does the PM do? why? What is a setting of LRI at which it will not pace?

3. **Maintaining the minimum heart rate [10%]**
   Solution Name: NPNwithDDD.slx
   For this, you will re-use the NPN model from question 1, so copy that first and call it NPNwithDDD.

- modify the `AtoVPath` block instance to have a conduction delay of 500.
- Add a DDD pacemaker by instantiating a `PM_DDD` stateflow chart from model_library.slx.
- Where should you connect `Asense` and `Vsense` ports?
- where should you connect Apace and `Vpace` ports, and how?
- what should you set the `LRI_def` to so a heart rate of about 60bpm is maintained?
- what should you set `AVI_def` to?
- add a scope that shows the `Asense`, `VSense`, `Apace` and `VPace`. we expect to see VPace happening, but no Apace.

4. **Getting familiar with the pacemaker [15%]**
   Solution Name: NSRExperiments.slx
   Use the NPNwithDDD.slx model you created in the previous question. Use the `LRI_def` and `AVI_def` parameter values obtained in the last question.
   In this question, you will modify the parameters of the heart model in order to reproduce 4 different sequences of events.
   From the pacemaker's perspective, every spontaneous (non-paced) event in the atrium is called an *Asense,* and every spontaneous (non-paced) event in the ventricle is called a *Vsense.* Similary, when the pacemaker paces the right atrium, it's called an *Apace,* and a ventricular pace is called a *Vpace.*
   a. Adjust the heart model parameters to obtain the following pacemaker event markers
      i. Asense – Vsense – Asense – Vsense…etc. I.e., the pacemaker does not intervene at all.
      ii. Asense – Vpace - Asense – Vpace - Asense – Vpace …etc. I.e. the atria are depolarizing normally, but for some reason, but the pacemaker finds it necessary to pace the ventricles. (Think: what could cause that?)
      iii. Apace – Vsense - Apace – Vsense -…etc. I.e. the pacemaker only intervenes to pace the atria. This pacing pulse is then normally transmitted to the ventricles, where it is sensed.
      iv. Apace – Vpace – Apace – Vpace…etc. I.e. the pacemaker finds it necessary to pace both atria and ventricles.
   b. What heart conditions could each of these 4 cases correspond to? (E.g. bradychardia, Excessive AV node delay, both, etc.)

5. **[10%]** What is the *set* of values of `LRI_def` and `AV_def` that produces each of the above observations? Hint: you will need to setup a linear program that encodes the constraints on these 2 parameters in terms of Rest period of atrium and conduction delays through the paths.