Developer's Guide To Pacemaker Development

Tutorial 4: Rate Adaptive Pacing

SFWRENG/MECHTRON 3K04
McMaster University

Michael Kehinde
kehindem@mcmaster.ca

May 5, 2021

Rate Adaptive Pacing Guide

The purpose of this document is to provide recommendations to support developing a rate adaptive pacing model.

Topics Covered

- Recommendations for implementing rate adaptive pacing

Prerequisites Review **Rate Adaptive Pacing Elaboration of Requirements**

1 Background

Please refer to Rate Adaptive Pacing Elaboration of Requirements.

2 Tutorial

Recommendations for getting started with designing and implementing your rate adaptive pacing algorithm. You may implement and/or combine some of these steps as you see fit with your design (i.e. using Simulink subsystem blocks, using Stateflow charts, or using Matlab function blocks in Simulink).

2.1 Demultiplex the raw accelerometer data.

Demultiplex the raw accelerometer data into x, y and z components.

Design decision: determine how you want to use each component realistically, i.e.

- which component/(s) to use,
- how to use them and why
- you can also think about whether to combine them

2.2 Convert raw acceleration to activity level.

The sensor output can be converted to activity level by:

1. smoothing the signal, and
2. mapping it to a scale of your choice.

Some of the points on that scale would represent different activity thresholds.

The purpose of performing signal smoothing is to smooth out short-term fluctuations while capturing long term trends in your data. Accelerometers are notorious for being sensitive and can produce really volatile data. Raw data can be unreliable to work with without signal smoothing. You may use any technique you find that works reliably. Potentially helpful resources:

1. **A few techniques in Matlab** (easy to reference the examples for Matlab function block implementations)
2. **Moving average block** (requires the DSP toolbox)
3. **Moving standard deviation block** (requires the DSP toolbox)

2.3 Obtain desired pacing rate from activity level.

Use the response factor and activity threshold parameters to implement a rate response function.

1. **Input:** activity level
2. **Parameters:** response factor, activity threshold
3. **Output:** desired pacing rate
4. **Behavior:**
   (a) for activity levels above the threshold: the higher the response factor, the higher the desired pacing race (and vice-versa)
   (b) ***Constraint***: the desired pacing rate should not exceed the MSR and should not fall short of the LRL

2.4 Obtain the current sensor controlled rate.

Obtain the current sensor controlled rate from the desired pacing rate. Use the reaction time and recovery time parameters to implement a time response function

1. **Input:** desired pacing rate, & the current pacing rate
2. **Parameters:** reaction time, recovery time
3. **Output:** the new current pacing rate
4. **Behavior:**
   (a) for desired pacing rates above the current pacing rate, increase the current pacing rate by X pulses per minute each 1-ms time step (based on the reaction time)

(b) for desired pacing rates below the current pacing rate, decrease the current pacing rate by X pulses per minute each 1-ms time step (based on the recovery time)

2.5 Implement the sensor controlled rate.

Use the sensor controlled rate in the Stateflow logic for the rate adaptive modes by implementing it within your timing transitions.

3 Best Practises

It is common to run into challenges when integrating the serial communication and rate adaptive pacing features with your pacemaker model. One potential outcome is finding that your timing if off by an integer multiple after integrating a new component. Make sure you've followed all the Simulink setup instructions from **Tutorial 1.2** closely and are also cognizant of the data types you're using for your variables (watch out for integer divisions).

If you are still running into issues with latency,

- implement the accelerometer block inside a triggered subsystem to enforce the timing of the accelerometer function to be aligned with the model.
- change the sample time in the accelerometer block to a value other than the model's fixed step size will end up changing the step size with the entire model, creating a side effect that will likely impact your serial communication procedures. Make sure the sampling time is set to "-1" (inherit) so that the sampling time of the block does not affect the sample time of the model.
- if all else fails, consider implementing the serial communication function and the pacemaker function as two parallel states to improve your model's throughput.

4 Revision History

Version	Date	Modification	Modified by
1.0	Nov. 25, 2020	Initial Document Creation	Kehinde, Michael