**Instructions for Code Used In:**

**Differentiating the contributions of Na+/K+ pump current and persistent Na+ current in simulated voltage-clamp experiments**

Jessica R. Parker1, Jan-Marino Ramirez1,2

1 Center for Integrative Brain Research, Seattle Children’s Research Institute, Seattle, Washington;

2 Departments of Neurological Surgery and Pediatrics, School of Medicine, University of Washington, Seattle, Washington

**Software Required and Installation Steps**

This code was built for a Linux or Mac computer. If it is at all possible for you to use a Linux or Mac, I absolutely recommend it. Otherwise, I would advise you to get a [virtual Linux machine](https://blog.storagecraft.com/the-dead-simple-guide-to-installing-a-linux-virtual-machine-on-windows/) instead of using just Windows. It is definitely possible to use Windows for this code, but I have never tried it.

Linux/Mac

* Open the terminal. You can find it with the launch pad.
* These next few steps are just for Macs:
  + Install [Xcode](https://developer.apple.com/xcode/), which can be found for free in the App Store, although it is often pre-installed on Mac computers.
  + Then, install the Xcode command line tools by entering the following line in the terminal

xcode-select –install

* + - A window will pop-up asking if you want to install the tools. Click “install” and follow the rest of the prompts.
* Back to both Linux and Mac, install [Homebrew](https://brew.sh/) if you don’t already have it. Homebrew is a very handy piece of software that makes installing other software very easy. To install Homebrew, enter the following line (It is supposed to be a single line) in the terminal.

/bin/bash -c "$(curl -fsSL https://raw.githubusercont

ent.com/Homebrew/install/HEAD/install.sh)"

* If you don’t have the code editor [emacs](https://www.gnu.org/software/emacs/) pre-installed or you want the most recent version, enter the following commands.

brew update

brew install emacs

sudo rm /usr/bin/emacs

sudo rm -rf /usr/share/emacs

* Install the [GSL](https://www.gnu.org/software/gsl/) scientific library with the following command:

brew install gsl

* Finally, you need to install [MATLAB](https://www.mathworks.com/products/matlab.html), which unfortunately costs money, although you can get a free trial. If you don’t have access to MATLAB, you can try Octave, which is like a generic, free version of MATLAB. Most syntax and built-in functions are the same, but there are some differences, which you would have to keep a look out for.

**Directory Contents and Organization**

Enter directory /SingleInspiratoryNeuron/ in the terminal. There are five sets of code here, each in its own subdirectory. The subdirectories are:

* /1\_FullCanonical/ - the full canonical inspiratory bursting model, used for Fig. 1A1-2 – data created takes up 28 MB
  + C-code for simulation takes 40 sec to run
* /2\_ReducedConstantNai/ - the reduced constant-[Na+]i model, used for Fig. 1B – data created takes up 1 GB
  + /2\_0\_0\_1/ - Sweeping [Na+]i from the left, increases [Na+]i parameter from 13 mM by steps of 0.25 mM. – C-code takes about 40 min to run.
  + /2\_0\_0\_2/ - Sweeping [Na+]i from the right, decreases [Na+]i parameter from 35 mM by steps of -0.25 mM. – C-code takes about 40 min to run.
* /3\_CanonicalVoltageClamp/ - the canonical voltage-clamp model, used for Fig. 3, - data created takes up 2.7 GB
  + /3\_0\_0\_0/ - Run simulation of cell at holding potential – 16 MB – C-code only takes 2 sec to run
  + /3\_0\_0\_1/ - Simulate leak current for steady-state activation measurement – 880 MB - takes about 2 min to run C-code
  + /3\_0\_0\_2/ - Measure steady-state activation – 880 MB - takes about 30 sec to run C-code
  + /3\_0\_0\_3/ - Simulate leak current for steady-state inactivation measurement – 220 MB - takes about 2 min to run C-code
  + /3\_0\_0\_4/ - Measure steady-state inactivation – 220 MB - takes about 30 sec to run C-code
  + /3\_0\_0\_5/ - Simulate leak current for time constant of inactivation measurement – 247 MB - C-code takes about 45 sec to run
  + /3\_0\_0\_6/ - Measure time constant of inactivation – 247 MB - C-code takes about 45 sec to run
* /4\_ModelA/ - Model A voltage-clamp model using 3 different values for IPumpMax (indexed by run5), see Fig. 5A – data created takes up 125 MB
  + /4\_0\_0\_1/ - Simulate leak current for tail current measurement – 62.5 MB – C-code takes about 9 sec
  + /4\_0\_0\_2/ - Depolarizing Model A to -40 mV to measure tail current – 62.5 MB – C-code takes about 9 sec
* /5\_ModelB/ - Model B voltage-clamp model using 3 different values for IPumpMax (indexed by run5), see Fig. 5B – data created takes up 125 MB
  + /5\_0\_0\_1/ - Simulate leak current for tail current measurement – 62.5 MB – C-code takes about 9 sec
  + /5\_0\_0\_2/ - Depolarizing Model B to -40 mV to measure tail current – 62.5 MB – C-code takes about 9 sec

Each of these directories contains many of the same code files.

**C Code for Integration**

Each directory contains:

* + compl – bash script file that compiles C code
  + RodentPreBotCNeuron.h – header file, also contains parameter values
  + RodentPreBotCNeuron\_main.c – main file, directs the flow of the program
  + RodentPreBotCNeuron\_integrate.c – integrates the model and saves output to /data/ subdirectory.
  + RodentPreBotCNeuron\_dy.c – defines differential equations
  + ip[..].txt – initial conditions

Some directories also contain:

* + RodentPreBotCNeuron\_integrateNW.c – integrates without saving data

To run the C code in any directory, just compile and run. Enter the following lines in the terminal

./compl

./runMouseCPG.exe

**Matlab Analysis and Plotting Code**

All directories contain:

* + readpars.m – parameter values used, corresponds to parameter values in the C code header file

For all directories, plots are saved in /plots/ subdirectory.

Directory-specific Matlab code:

* /1\_FullCanonical/
  + burstanalysis.m – Matlab function that calculates burst characteristics
  + runburstanalysisSC.m – MATLAB script that calls to burstanalysis() and saves the resulting burst characteristic data
  + plotVNai.m – Script that plots the voltage and [Na+]i traces over time, with axes instead of scale bars
  + plotVNaiIpumpInap.m – Script that plots the voltage, [Na+]i, IPump, INaP traces over time, with axes instead of scale bars
  + plotVNaiIpumpInapFormal.m – Script that plots the voltage, [Na+]i, IPump, INaP traces over time, with scale bars instead of axes. Reproduces Fig. 1A1 as is. Set SingleBurst to 1 on line 25 to reproduce Fig. 1A2.
  + Order to run code:
    - After running C-code, run runburstanalysisSC.m first. Then you can run plot[…].m files in whatever order you choose.
* /2\_ReducedConstantNai/
  + plotReducedModelBifurcation.m – Script that plots the min and max membrane potentials versus [Na+]i of the reduced constant-[Na+]i model data in 2\_0\_0\_1 and 2\_0\_0\_2, and overlays full model data from /1\_FullCanonical/. Reproduces Fig. 1B.
    - Cannot be run until after C-code and runburstanalysisSC.m have been run in /2\_0\_0\_1/ and /2\_0\_0\_2/, and in /1\_FullCanonical/.
  + /2\_0\_0\_1/ and /2\_0\_0\_2/
    - burstanalysis.m – Function that calculates burst characteristics
    - runburstanalysisSC.m – Script that calls to burstanalysis() and saves the resulting burst characteristic data
    - plotJustV.m – Script that plots the voltage trace over time, with axes instead of scale bars.
  + Order to run:
    - First make sure you have already run C-code and runburstanalysisSC.m in /1\_FullCanonical/.
    - Go to /2\_0\_0\_1/ and run C-code and then run runburstanalysisSC.m. You can then run plotJustV.m if you want.
    - Then go to /2\_0\_0\_2/ and run C-code and subsequently runburstanalysisSC.m. Then run plotJustV.m if you want.
    - Then back up to /2\_ReducedConstantNai/ and run plotReducedModelBifurcation.m.
* /3\_CanonicalVoltageClamp/
  + All subdirectories, except /3\_0\_0\_0/ contain:
    - plotItotVclamp.m – Matlab script plots the total current of the data simulated in this directory only, so it does not do leak subtraction.
  + /3\_0\_0\_2/
    - measureNaPactivation.m – Script loads the simulation data made by the C code and measures the steady-state activation level at an array of different membrane potentials, which the cell is clamped at for run6 = 2 and which are indexed by run5.
    - sweepFitBoltzmannMnap.m – Script that estimates a first order Boltzmann function of best fit for the steady-state activation levels measured in measureNaPactivation.m. Plots a figure showing part of Fig. 3A.
    - measureTailCurrent.m – Script measures and saves the tail current and tail conductance from the peak gNaP onwards during the last membrane potential step to -40 mV for run5 = 31. It is only used for run5 = 31 in order to make Fig. 3C and Fig. 5C.
    - sweepFitDecayRate.m – Script takes the tail current data measured in measureTailCurrent.m and fits it with double exponential equation. This code may take a few minutes to run. It is not the most efficient way to get the best fit, but it is effective. Plots figure equivalent to Fig. 3C2.
    - plotDecayRateFit.m – Script just plots the output from sweepFitDecayRate.m so that you don't have to rerun that sweep fit everytime you want to look at or change the plot.
    - plotTraceVariablesCurrentsFormal.m – Script plots measured INaP, normalized gNaP as measured, Vclamp, [Na+]i, ENa, the actual model INaPn, and IPump. Reproduces Fig. 5C.
    - plotInapGnapVclamp.m – Script measures and plots INaP and gNaP after subtracting leak current, but it does not write any data to files. It may seem redundant since measureNaPactivation.m and measureTailCurrent.m also make figures like these, but it is useful if you want to make changes to the figures without worrying about overwriting data files.
    - Order to run code:
      * After running C-code in both /3\_0\_0\_1/ and /3\_0\_0\_2/, plotTraceVariablesCurrentsFormal.m, plotInapGnapVclamp.m, and plotItotVclamp.m can be run at any time.
      * Run measureNaPactivation.m before running sweepFitBoltzmannMnap.m.
      * The remaining three files need to be run in this order measureTailCurrent.m -> sweepFitDecayRate.m -> plotDecayRateFit.m.
  + /3\_0\_0\_4/
    - measureNaPinactivation.m – Script loads the simulation data made by the C code and measures the steady-state inactivation level at an array of different membrane potentials, which the cell is clamped at for run6 = 1 and which are indexed by run5.
    - sweepFitBoltzmannHnap.m – Script that estimates a first order Boltzmann function of best fit for the steady-state inactivation levels measured in measureNaPinactivation.m. Plots a figure showing part of Fig. 3B.
    - plotInapGnapVclamp.m – Script does same thing as in /3\_0\_0\_2/.
    - Order to run code:
      * After running C-code in both /3\_0\_0\_3/ and /3\_0\_0\_4/, run measureNaPinactivation.m before running sweepFitBoltzmannHnap.m.
  + /3\_0\_0\_6/
    - measureTauHnap.m – Script measures the inactivation level reached after the deinactivating step (run6 = 2), which has an array of different durations (indexed by run5).
    - sweepFitTimeConstant.m – Script measures the time constant of inactivation by fitting the inactivation level reached after the deinactivating step versus the duration of the deinactivating step (indexed by run5).
    - plotInapGnapVclamp.m – Script does same thing as in /3\_0\_0\_2/ and /3\_0\_0\_4/.
    - Order to run code:
      * After running C-code in both /3\_0\_0\_5/ and /3\_0\_0\_6/, run measureTauHnap.m before running sweepFitTimeConstant.m.
* /4\_ModelA/
  + Both /4\_0\_0\_1/ and /4\_0\_0\_2/ contain:
    - plotItotVclamp.m – Matlab script plots the total current of the data simulated in this directory only, so it does not do leak subtraction.
  + /4\_0\_0\_2/
    - measureTailCurrent.m – Script measures and saves the tail current and tail conductance from the peak gNaP onwards during the last membrane potential step to -40 mV for each of the IPumpMax values (indexed by run5).
    - sweepFitDecayRate.m – Script takes the tail current data measured in measureTailCurrent.m for each IPumpMax (run5) value and fits each with a double exponential equation. This code may take a few minutes to run. It is not the most efficient way to get the best fit, but it is effective.
    - plotDecayRateFit.m – Script plots the output from sweepFitDecayRate.m for a single run5 instance so that you don't have to rerun that sweep fit everytime you want to look at or change the plot.
    - plotMultipleDecayFits.m – Script plots the output from sweepFitDecayRate.m for all run5 instances together on one figure. Produces figure roughly equivalent to Fig. 5A2.
    - plotTraceVariablesCurrentsFormal.m – Script plots measured INaP, normalized gNaP as measured, Vclamp, IPump, actual model INaPn, and measured INaP - IPump. Reproduces Fig. 5A1.
    - plotInapGnapVclamp.m – Script measures and plots INaP and gNaP after subtracting leak current, but it does not write any data to files.
    - Order to run code:
      * Run C-code in both /4\_0\_0\_1/ and /4\_0\_0\_2/ before running any Matlab code.
      * Order only matters for these files, which must be run in this order, measureTailCurrent.m -> sweepFitDecayRate.m -> plotDecayRateFit.m and/or plotMultipleDecayFits.m.
* /5\_ModelB/ (Basically the exact same as /4\_ModelA/, but I am going to repeat it anyway)
  + Both /5\_0\_0\_1/ and /5\_0\_0\_2/ contain:
    - plotItotVclamp.m – Matlab script plots the total current of the data simulated in this directory only, so it does not do leak subtraction.
  + /5\_0\_0\_2/
    - measureTailCurrent.m – Script measures and saves the tail current and tail conductance from the peak gNaP onwards during the last membrane potential step to -40 mV for each of the IPumpMax values (indexed by run5).
    - sweepFitDecayRate.m – Script takes the tail current data measured in measureTailCurrent.m for each IPumpMax (run5) value and fits each with a double exponential equation. This code may take a few minutes to run. It is not the most efficient way to get the best fit, but it is effective.
    - plotDecayRateFit.m – Script plots the output from sweepFitDecayRate.m for a single run5 instance so that you don't have to rerun that sweep fit everytime you want to look at or change the plot.
    - plotMultipleDecayFits.m – Script plots the output from sweepFitDecayRate.m for all run5 instances together on one figure. Produces figure roughly equivalent to Fig. 5B2.
    - plotTraceVariablesCurrentsFormal.m – Script plots measured INaP, normalized gNaP as measured, Vclamp, [Na+]i, ENa, and the actual model INaPn. Reproduces Fig. 5B1.
    - plotInapGnapVclamp.m – Script measures and plots INaP and gNaP after subtracting leak current, but it does not write any data to files.
    - Order to run code:
      * Run C-code in both /5\_0\_0\_1/ and /5\_0\_0\_2/ before running any Matlab code.
      * Order only matters for these files, which must be run in this order, measureTailCurrent.m -> sweepFitDecayRate.m -> plotDecayRateFit.m and/or plotMultipleDecayFits.m.