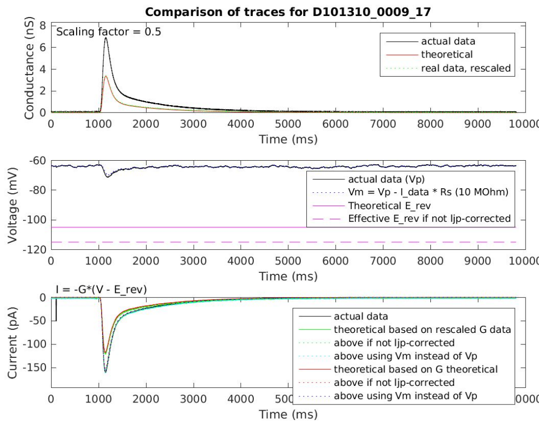
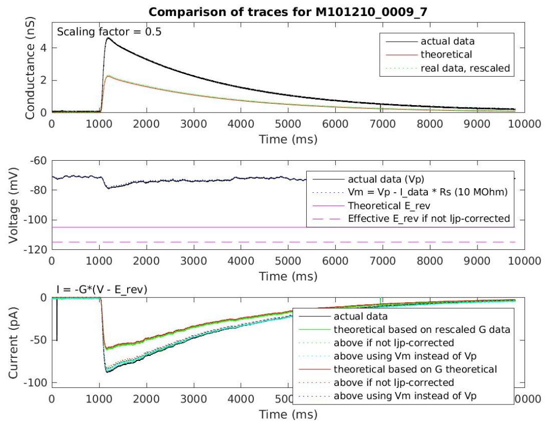
10/31/2016~11/29/2016

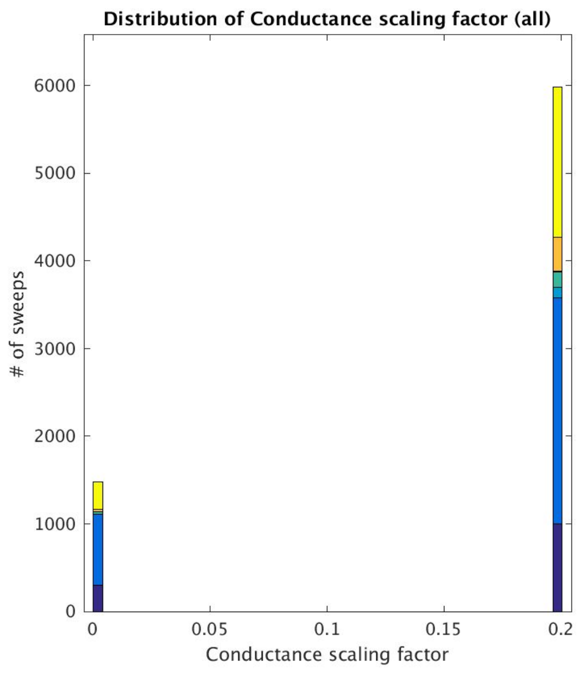
<u>Trace comparison (cont'd)</u>

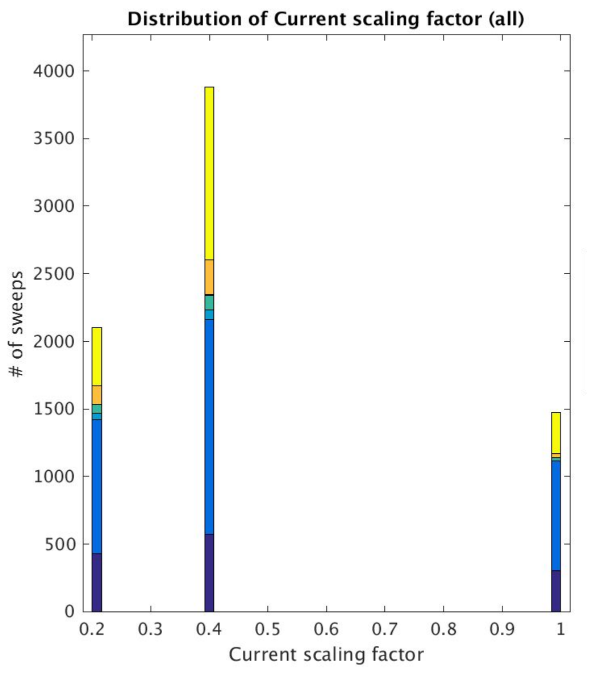
- Rescaled conductance data based on theoretical curve. The new scaling factor is saved as a variable **condscale2**.
    - Realized that conductance data were probably arbitrarily scaled by a factor different from that of the current data



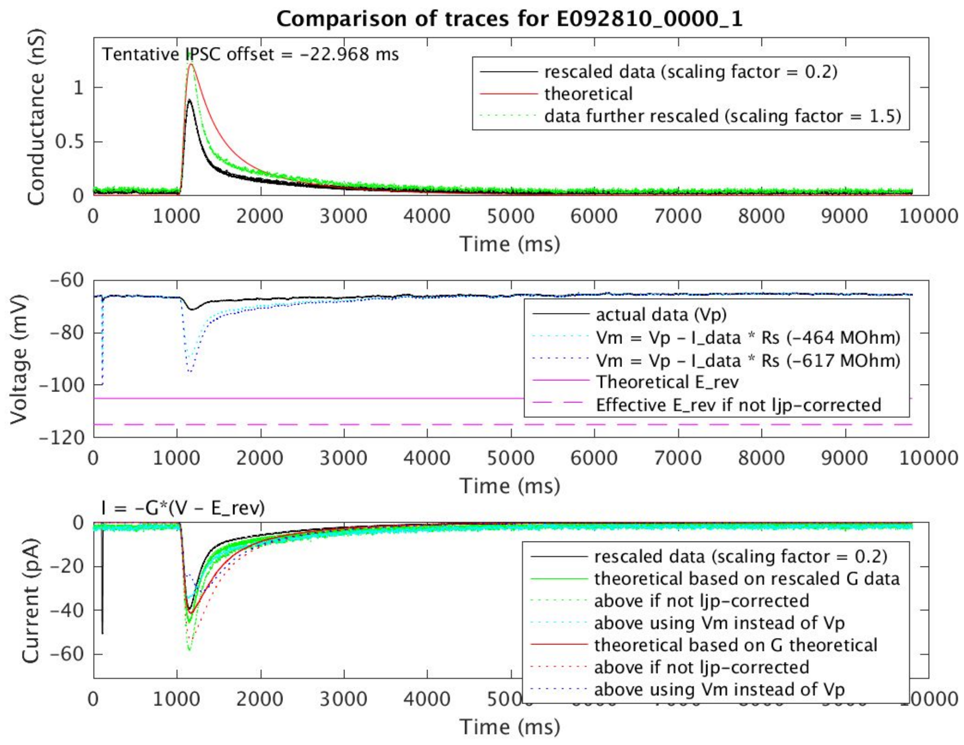


- Rescaled conductance data based on the **theoretical conductance curve** (which is in turn based on the pharm condition and G incr % associated with the set) in **ResaveSweeps.m**, which is the data reorganization part that was moved from **dclampDataExtractor.m**. The current data was still rescaled based on the fact that the current pulse should always be **-50 pA**. The resulting scaling factors are saved as '**condscale**' & '**currscale**' in **dclampdatalog_take4_resave.mat**.





- Reran trace_comparison.m with the expectation that **condscale2** should now always be **1**. This was true for all sets except **E092810_0000**, which still has aberrant conductance traces that do not shape the same either:

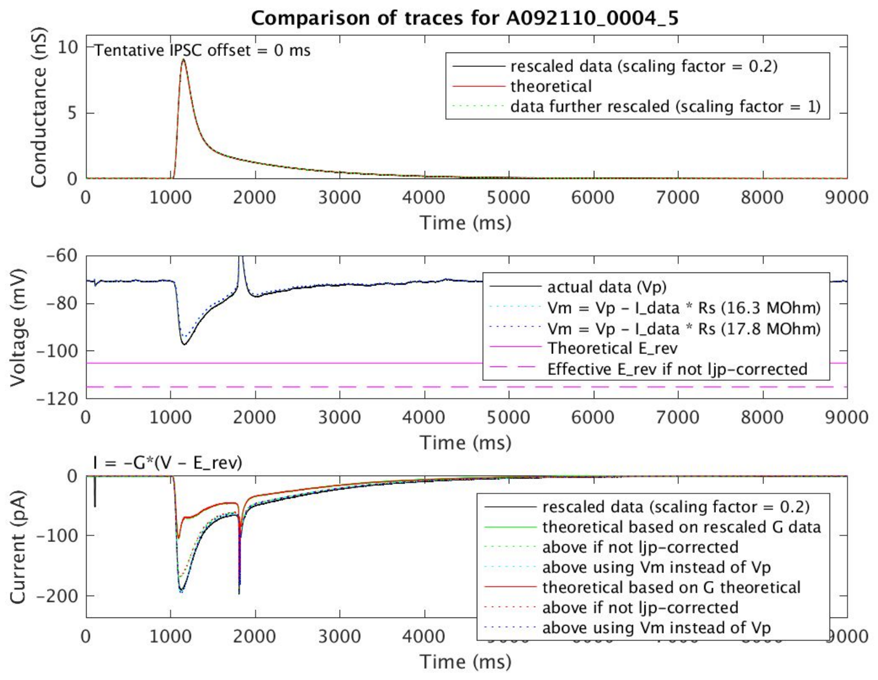


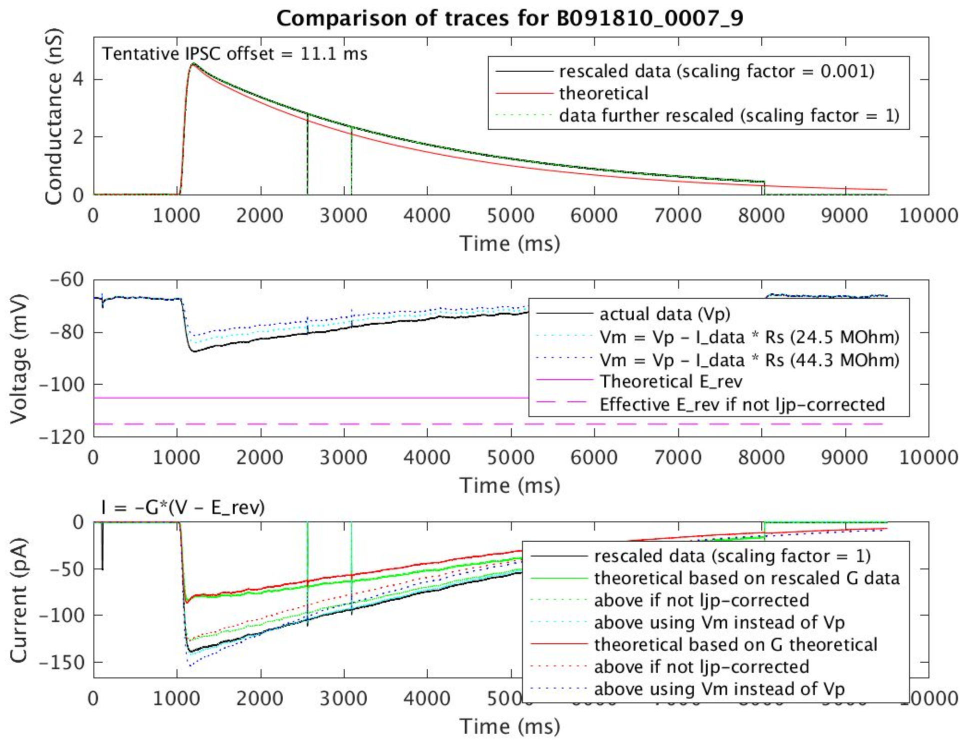
    - An examination of the corresponding dynamic clamp output file (**10928-17_58_46.PRN**) shows that a **GAT1 block curve** was applied instead of the **GAT3 block** condition associated with it.

- Modifications to **CountSweeps.m**:
    - 2016-11-30 **E092810_0000** was added to **broken_files** and taken out of all analyses (otherwise the GAT1 block might be overrepresented statistically)

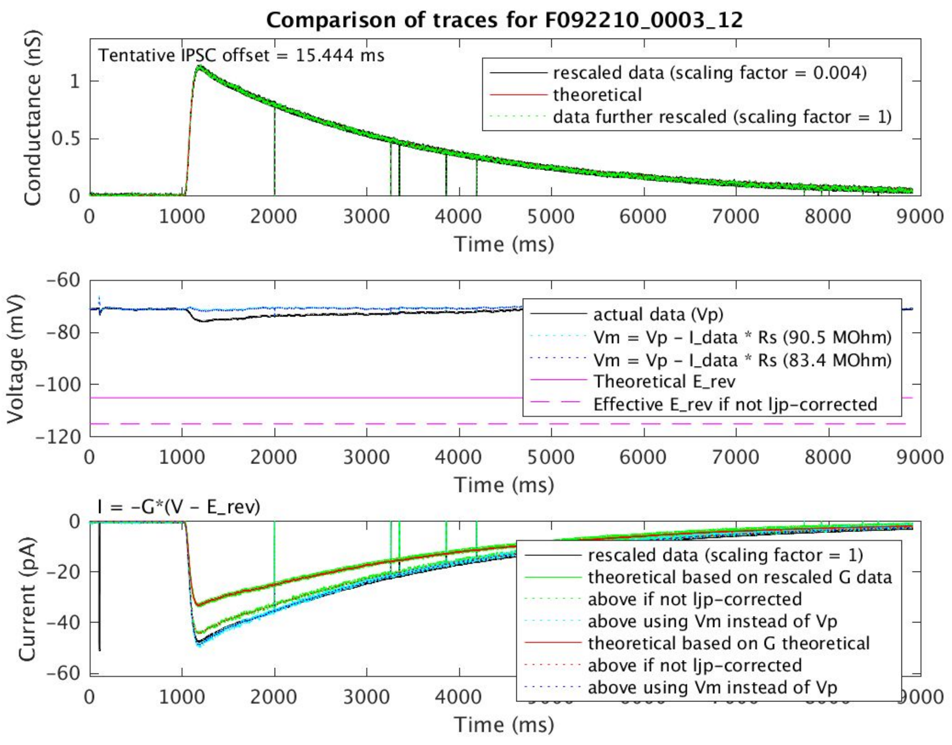
- Modifications to **dclampDataExtractor.m**:
    - 2016-11-30 **maxsets** was changed from 347 to **346**; **maxswps** was changed from 7455 to **7430**

- Assuming that the dynamic clamp computer corrected for **series resistance** (and that PClamp did not), calculated the current traces that might have been applied by the dynamic clamp computer (**I_theo1_corr** & **I_theo2_corr**)

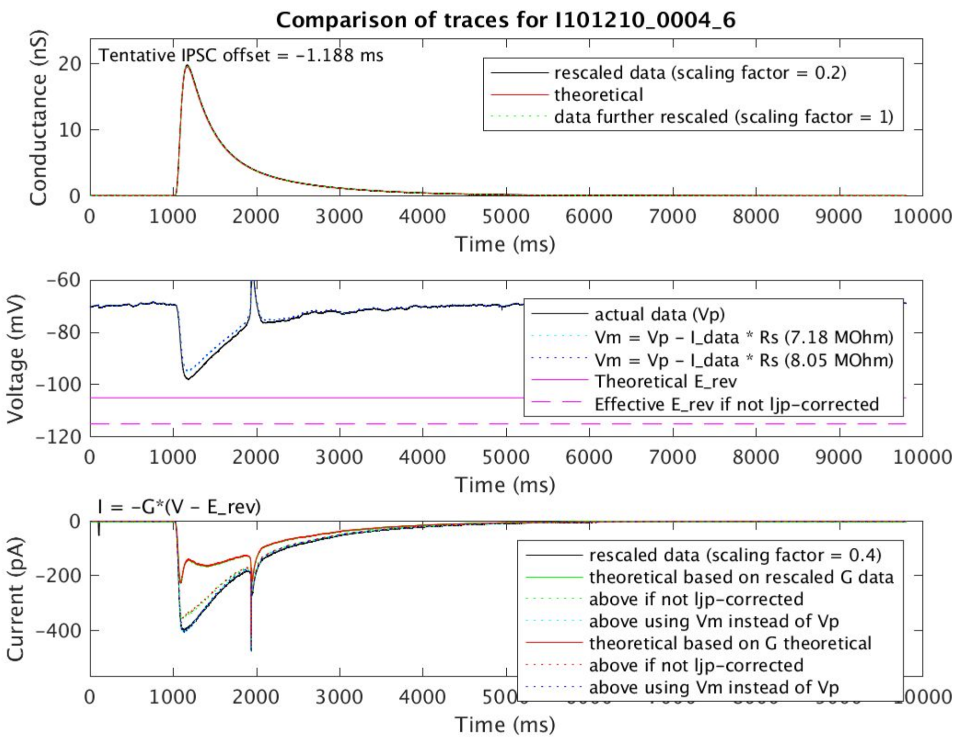


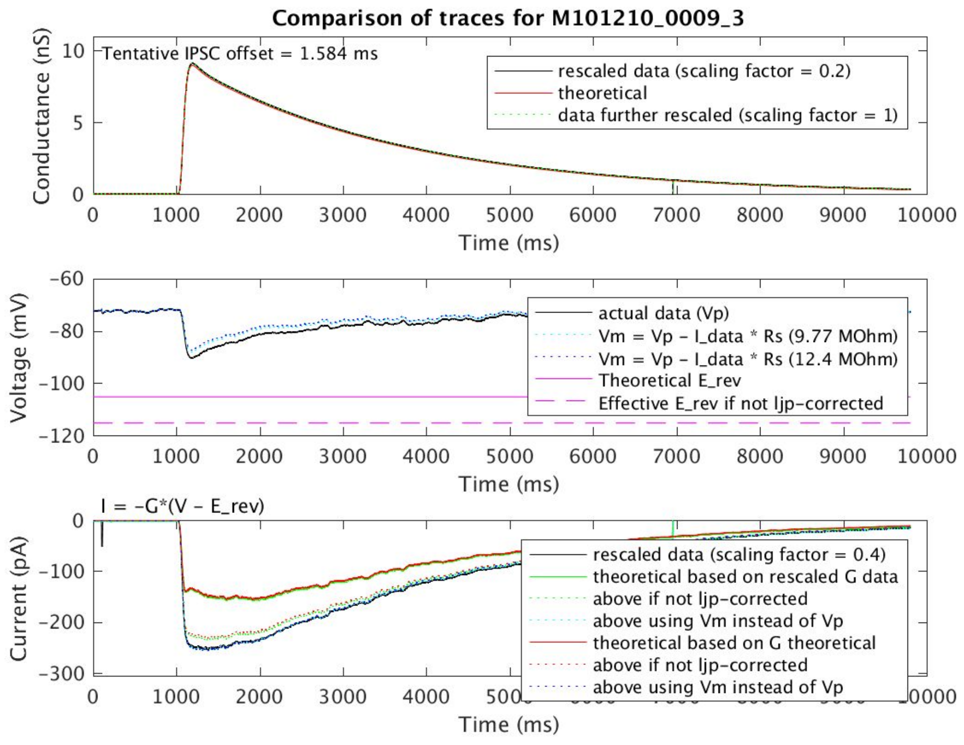


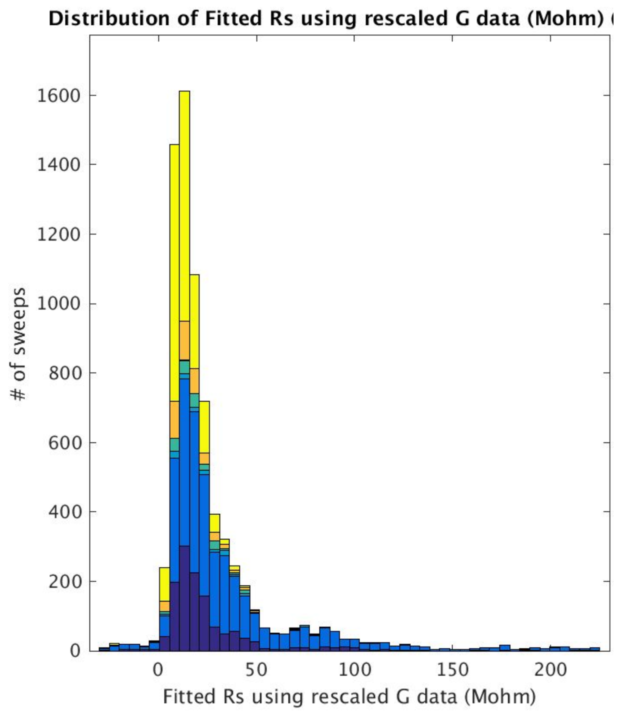
Comparison of traces for F092210_0003_12

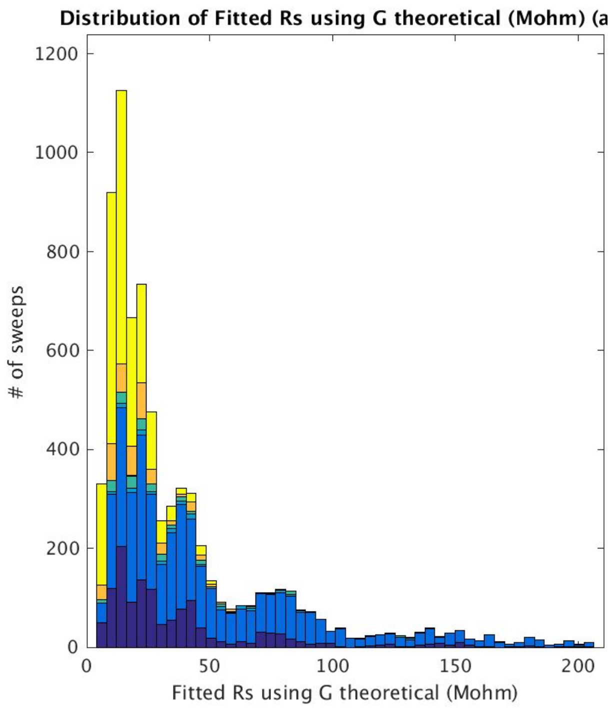


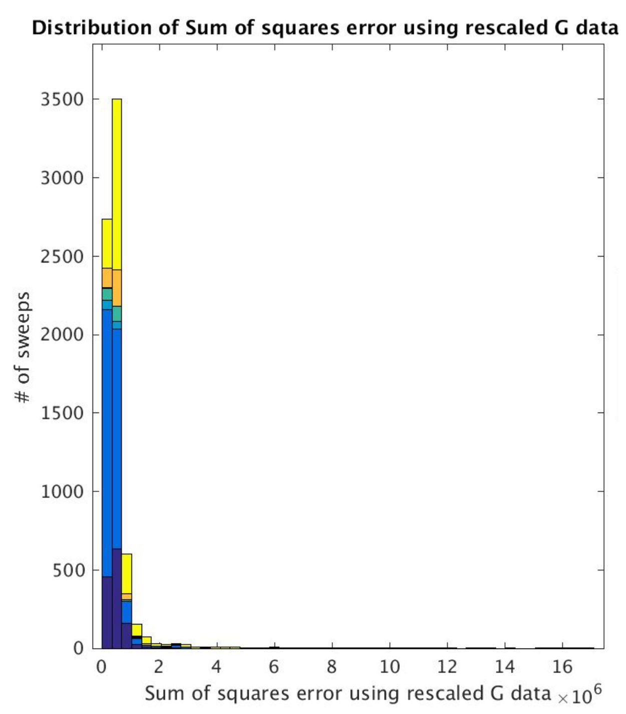
Comparison of traces for I101210_0004_6

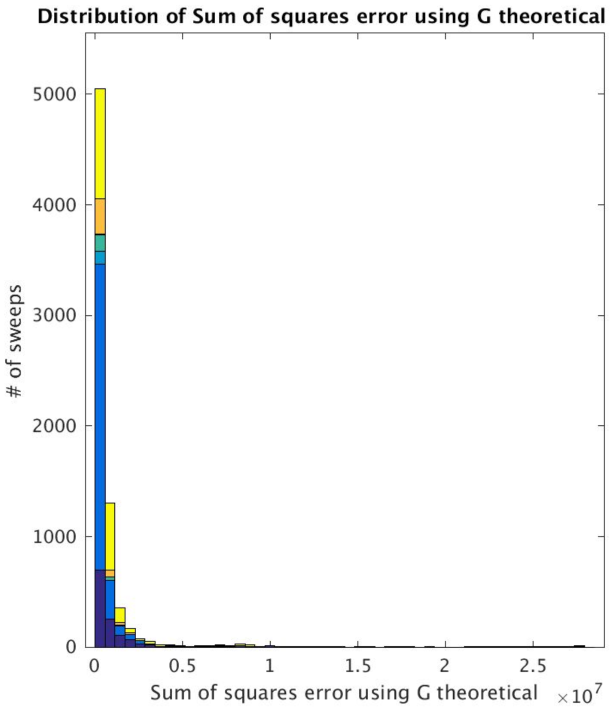



Comparison of traces for M101210_0009_3
Tentative IPSC offset = 1.584 ms
rescaled data (scaling factor = 0.2)
theoretical
data further rescaled (scaling factor = 1)
Conductance (nS)
Time (ms)
actual data (Vp)
Vm = Vp - I_data * Rs (9.77 MOhm)
Vm = Vp - I_data * Rs (12.4 MOhm)
Theoretical E_rev
Effective E_rev if not ljp-corrected
Voltage (mV)
I = -G*(V - E_rev)
rescaled data (scaling factor = 0.4)
theoretical based on rescaled G data
above if not ljp-corrected
above using Vm instead of Vp
theoretical based on G theoretical
above if not ljp-corrected
above using Vm instead of Vp
Current (pA)


Distribution of Fitted Rs using rescaled G data (Mohm) (
# of sweeps
Fitted Rs using rescaled G data (Mohm)


Distribution of Fitted Rs using G theoretical (Mohm) (a
# of sweeps
Fitted Rs using G theoretical (Mohm)





- Modifications to **trace_comparison.m**::
    - 2016-11-07 -Moved code to **compute_conductance.m**, **compute_elcurr.m**
    - 2016-11-07 -Added **scaling factor**, **G_data_rescaled**, **Rs**, **V_corr1**, **V_corr2**, **I_theo1_corr** & **I_theo2_corr**
    - 2016-11-07 **-I_theo1** now comes from **G_data_rescaled** instead of G_data
    - 2016-11-09 - Modified **Rs** so that it is fitted
    - 2016-11-09 - Added text labels
    - 2016-11-10 -Created logheader & logvariables; made variables row vectors instead of column vectors

10/31/2016~11/13/2016

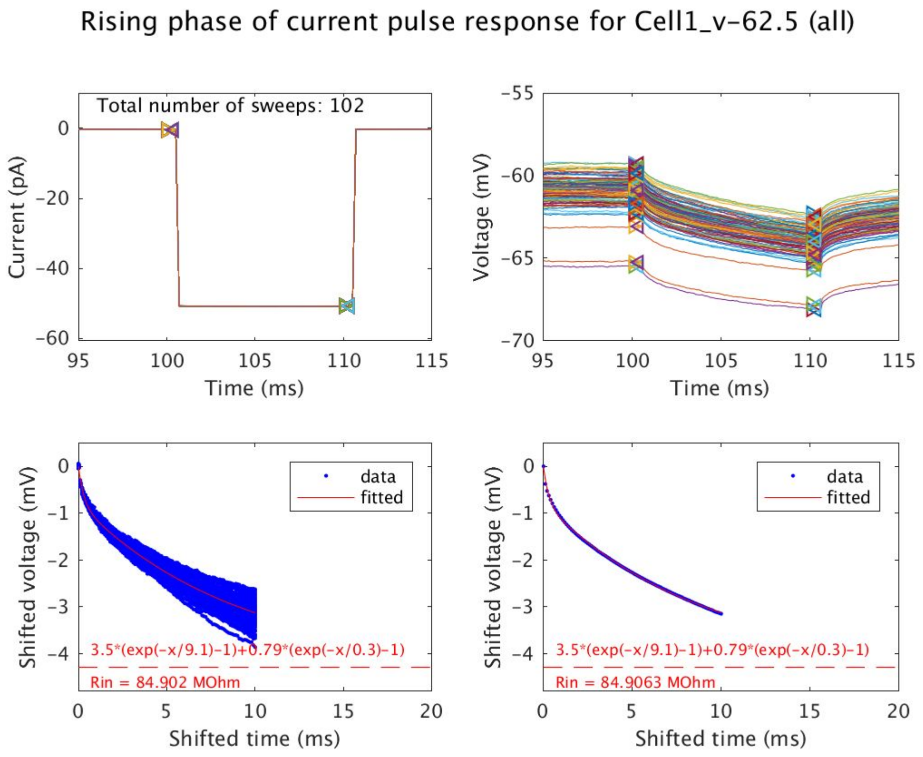
Passive fitting (cont'd)

- Instead of using Vhold, regrouped voltage traces using the the actual holding potential (**actVhold**) recorded, binning the traces into three sets for each cell:
  - (1) V > -65 mV
  - (2) -70 mV < V ≤ -65 mV
  - (3) V ≤ -70 mV
- Procedure for passive fitting of the **rising phase**:
  1. For each sweep, the current pulse were first detected from the median-filtered (window = **10 ms**) current trace by the following:
     a. The **current pulse amplitude** (**cpa**), a negative number, was computed by subtracting the average of the current between **105~105.5 ms** from the average of the current between **95~95.5 ms**
     b. The **current pulse start** (**cpstart**) was then the first index since **95 ms** that was more positive than ¼ the current pulse amplitude
     c. If the first dip point does not exist, the trace is deemed faulty and omitted
     d. Otherwise, the **current pulse end** (**cpend**) was then the last index since **105 ms** that was more negative than ¾ the current pulse amplitude
     e. The **pulse width** (**pw**) was then computed as the time interval between the first dip point and the before rise point
  2. The **recorded voltage change** (ΔV_{rec}) was then computed by the following:
     a. The **baseline voltage** (V_{base}) was computed as the average of the voltage trace over **0.5 ms** before **cpstart**
     b. The **final voltage** (V_{last}) was computed as the average of the voltage trace over **0.5 ms** before **cpend**
     c. $\Delta V_{rec} = V_{base} - V_{last}$(Should be positive)
  3. If the recorded voltage change was nonpositive, or if the current pulse amplitude was nonnegative, the trace was deemed faulty and omitted
  4. The **mean recorded voltage change** ($\Delta \bar{V}_{rec}$), the **mean current pulse amplitude** (**cpa_mean**), the **mean pulse width** (**pw_mean**) were than computed by averaging over all traces whose values respectively make sense (i.e., > 0, < 0, > 0).
  5. If the maximum of the voltage trace over **95~500 ms** was greater than **-45 mV**, the trace was deemed to have spontaneous spikes during the current pulse response window and omitted
  6. Each remaining voltage trace was then cropped between **cpstart** and **cpend**, time-shifted so that the trace begins at t = 0, and subtracted by **basev** throughout (to get ΔV)
  7. The voltage traces were then either **pooled together** or **averaged** before fitting with the following equation:

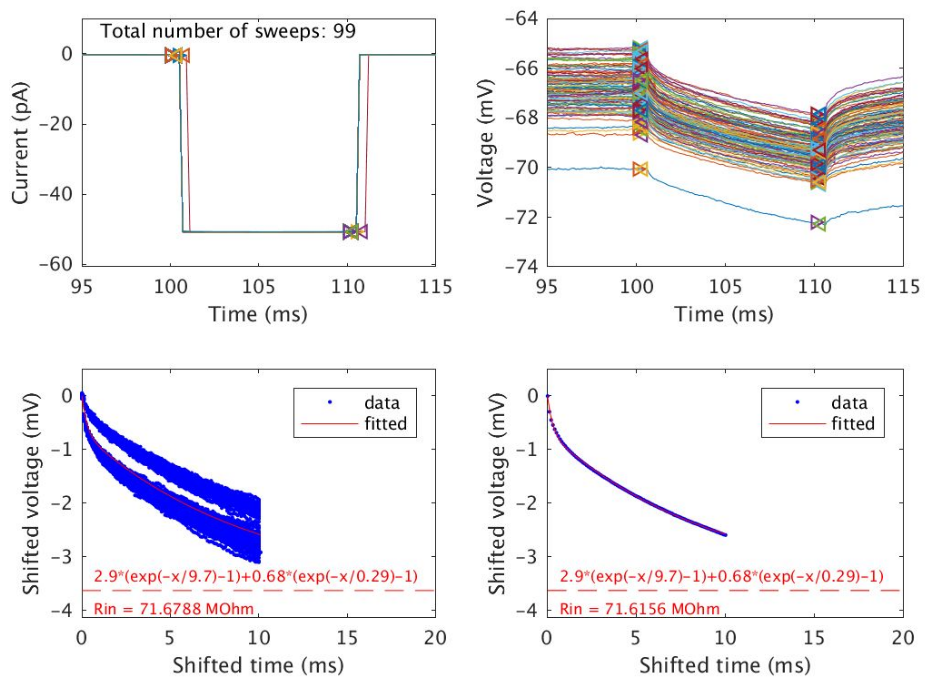
$$\Delta V = C_{L0}(e^{-t/\tau_0} - 1) - C_{L1}(e^{-t/\tau_1} - 1)$$

using the following initial conditions and bounds:

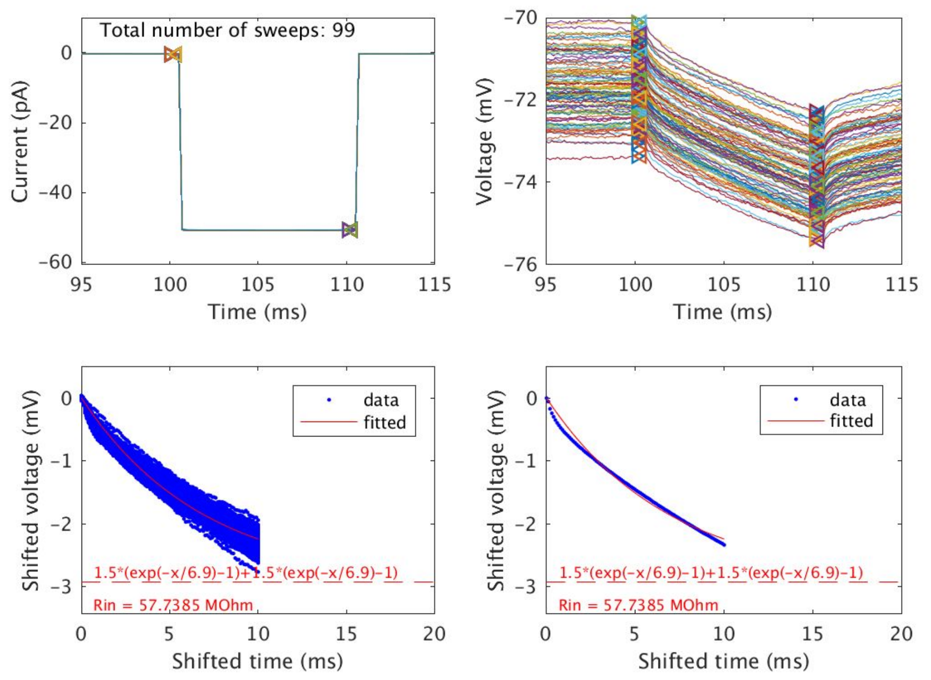
	Initial condition	Upper bound	Lower bound
$C_{L0},\ C_{L1}$	$\Delta\bar{V}_{rec}$	$10\Delta\bar{V}_{rec}$	**0 mV**
$\tau_0,\ \tau_1$	**2 ms**	**200 ms**	**0 ms**



Rising phase of current pulse response for Cell1_v-67.5 (all)

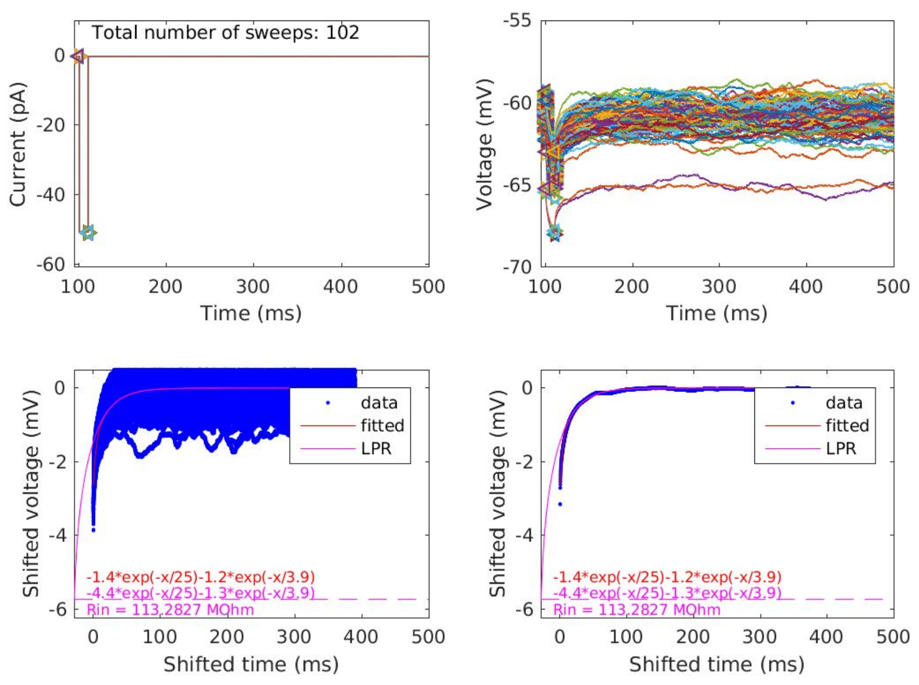


Rising phase of current pulse response for Cell1_v-72.5 (all)

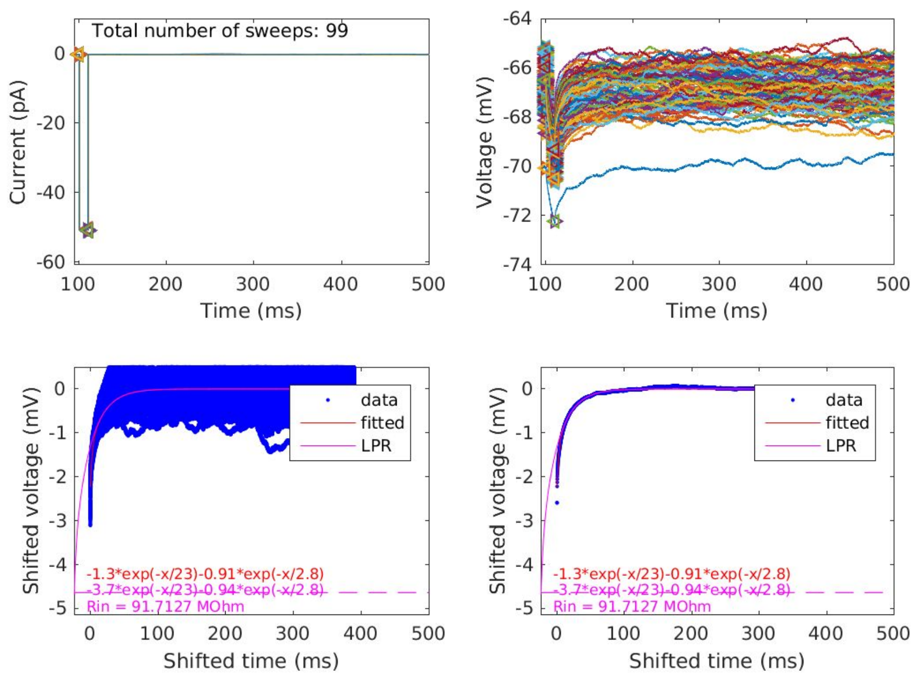


- The **falling phase** (**110 ms** to **500 ms**) of the current pulse response (the voltage trace was subtracted from the same **basev** as extracted above) was fitted by the following equation to obtain the **short-pulse coefficients**:
  $$\Delta V = -C_{S0}e^{-t/\tau_0} - C_{S1}e^{-t/\tau_1}$$
  The corresponding **long-pulse coefficients** were then extrapolated from the following equations (Johnston & Wu, p. 95):
  $$C_{S0} = C_{L0} * (1 - e^{-w/\tau_0})$$
  $$\Rightarrow C_{L0} = C_{S0}/(1 - e^{-w/\tau_0})$$
  $$C_{S1} = C_{L1} * (1 - e^{-w/\tau_1})$$
  $$\Rightarrow C_{L1} = C_{S1}/(1 - e^{-w/\tau_1})$$
  where w was the pulse width (~10 ms)

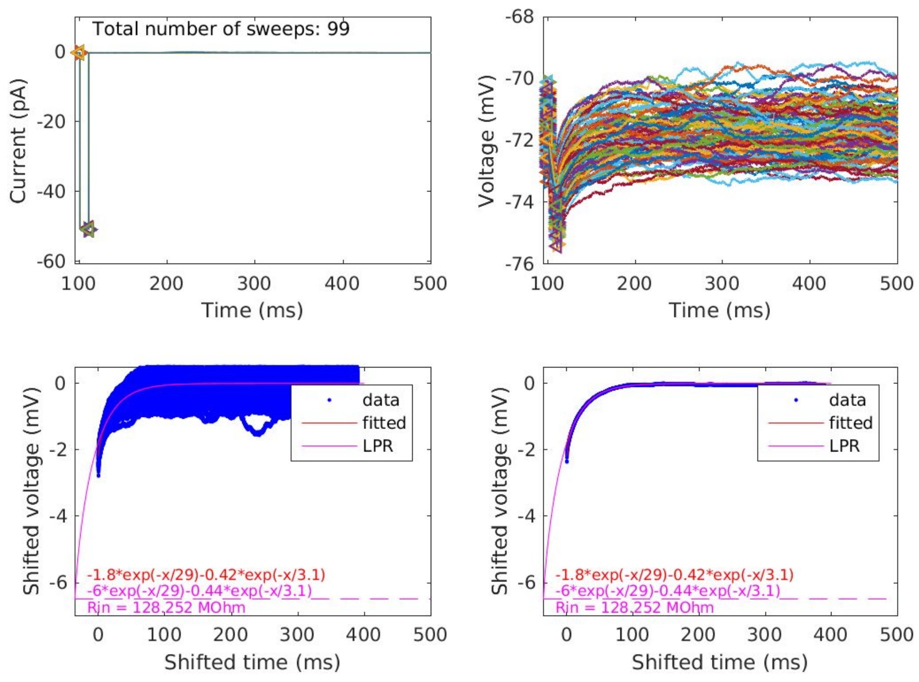
Falling phase of current pulse response for Cell1_v-62.5 (all)



Falling phase of current pulse response for Cell1_v-67.5 (all)

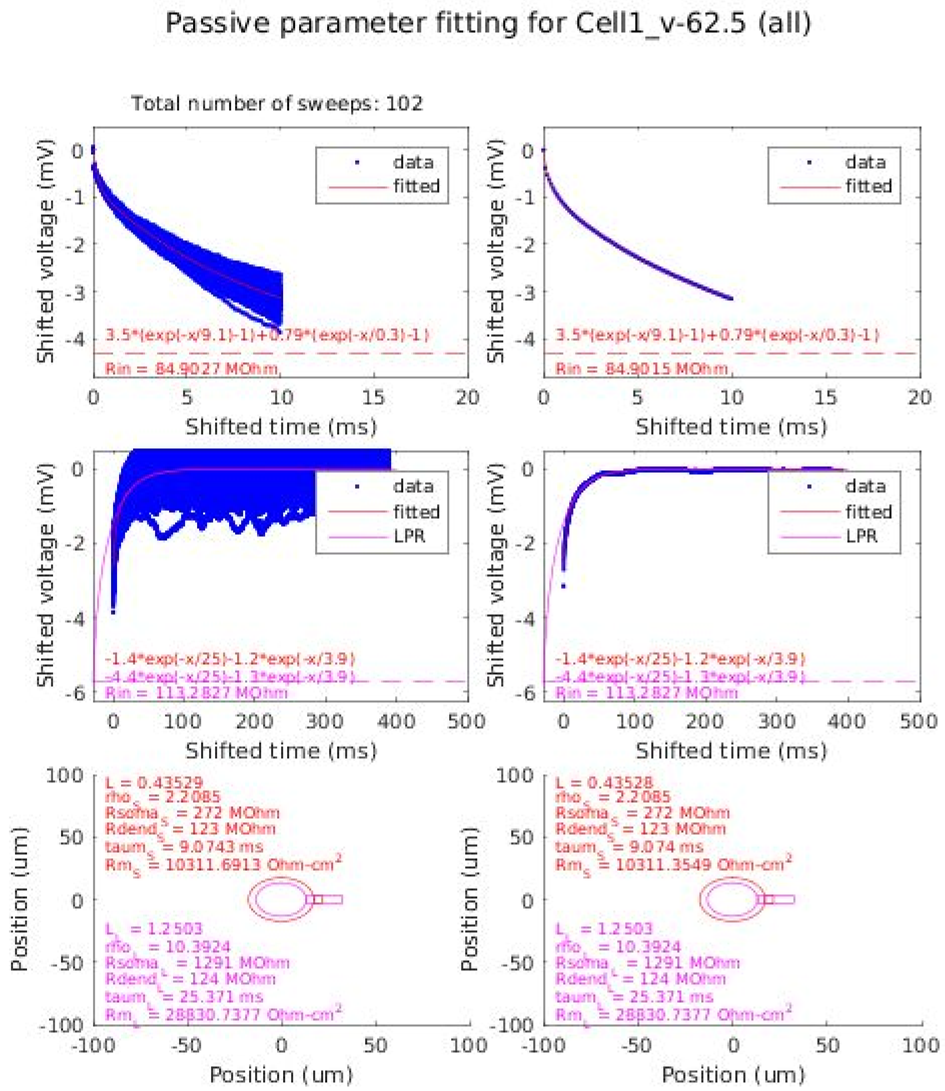


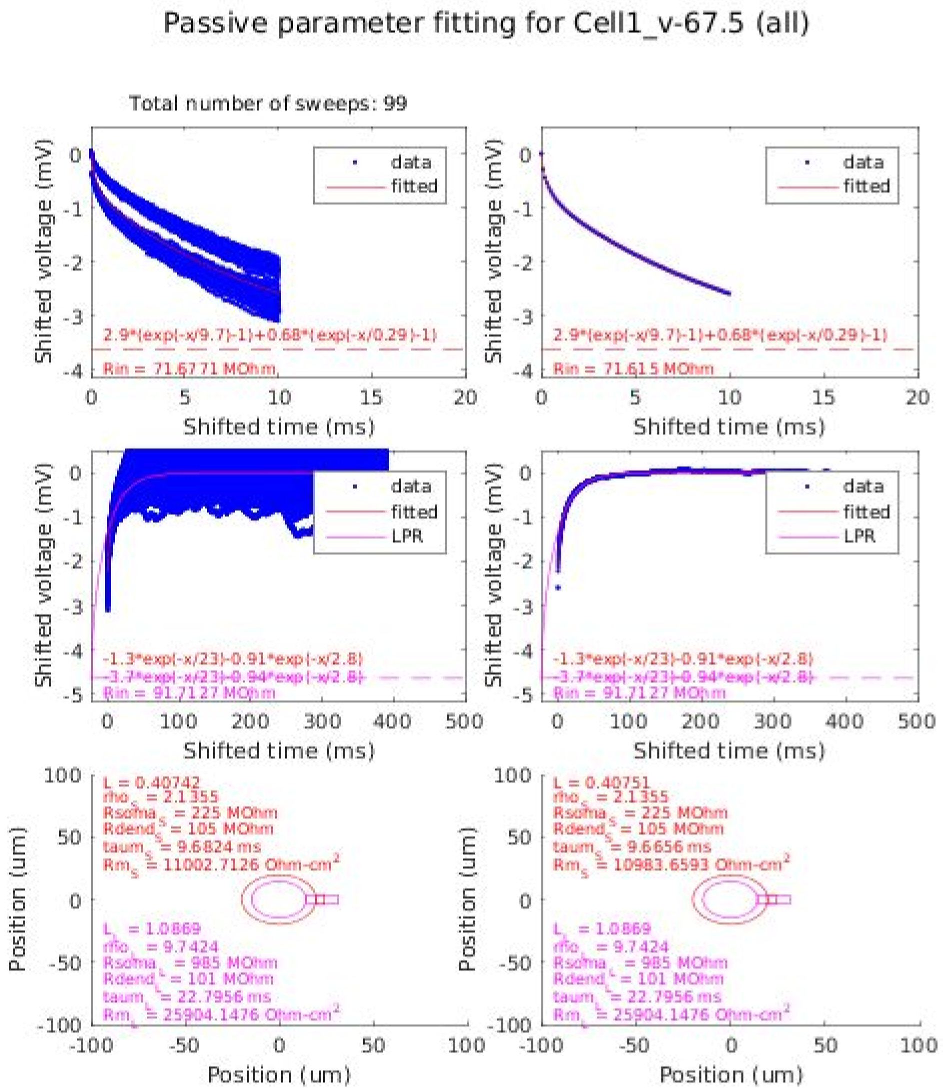
Falling phase of current pulse response for Cell1_v-72.5 (all)

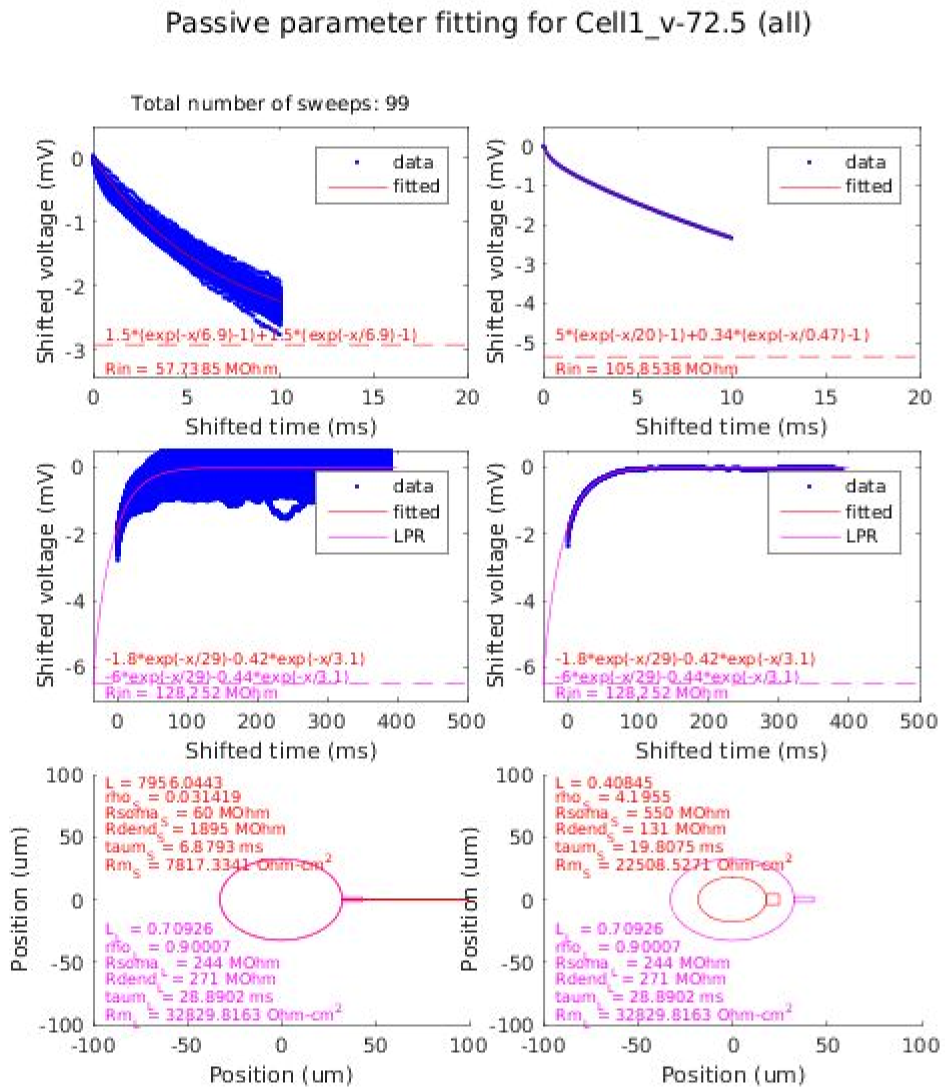


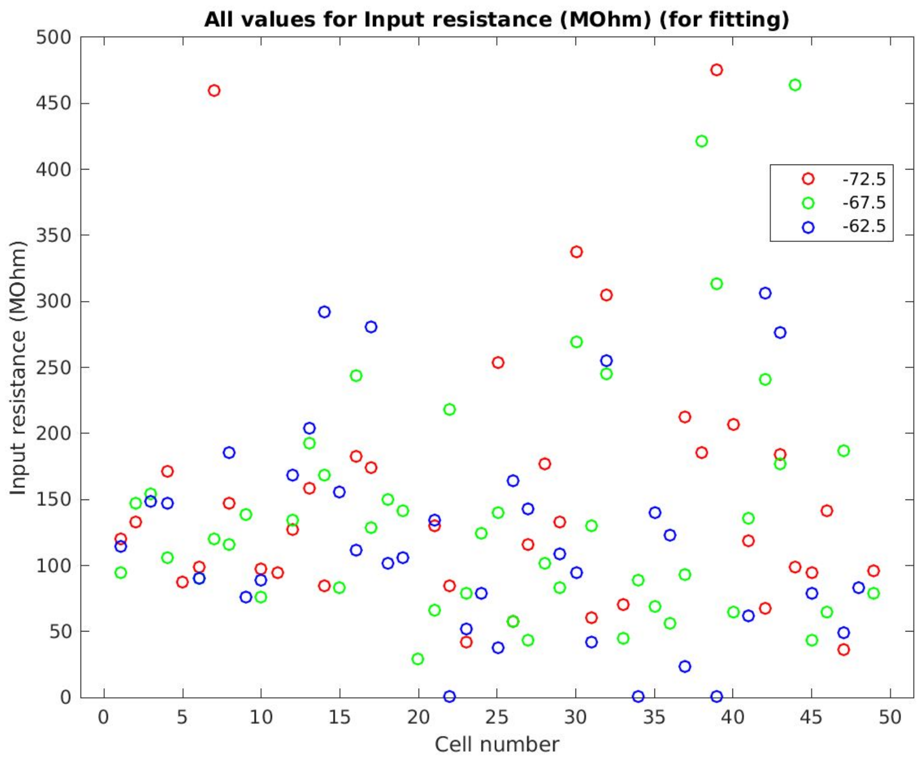
- Estimated passive parameters from the fitted coefficients and time constants:
  - From the coefficients of the two exponential terms, the **steady-state voltage change (ΔV_{ss}, [mV])** is given by:
    $$\Delta V_{ss} = -C_{L0} - C_{L1}$$
  - Thus the **input resistance (R_{in}, [MΩ])** is given by (with proper unit correction **10^3 MΩ/GΩ**)
    $$R_{in} = (10^3\ \mathrm{M\Omega/G\Omega}) \frac{\Delta V_{ss}}{\mathrm{cpa}_{mean}}$$
  - Now from Johnston & Wu, p. 96, the coefficients and time constants of the two exponential terms can yield an estimate of the electrotonic length L and the dendritic-to-somatic conductance ratio ρ through the following equations:
    $$\alpha_1 = \sqrt{\frac{\tau_0}{\tau_1} - 1}$$
    The **electrotonic length** (**L, [1]**) is solved from the equation:
    $$\left| \frac{C_{L1}}{2C_{L0}\frac{\tau_1}{\tau_0} - C_{L1}} \right| = \cot(\alpha_1 L)(\cot(\alpha_1 L) - \frac{1}{\alpha_1 L}) \tag{1}$$
    using **vpasolve** with the initial condition
    $$L_{\text{init}} = \frac{\pi}{\alpha_1}$$
    The **dendritic-to-somatic conductance ratio (ρ, [1])** is then found by
    $$\rho = -\frac{\alpha_1 \cot(\alpha_1 L)}{\coth(L)}$$
    If either L or ρ is negative, Eq(1) is solved again with L_init = **rand(1)** (a random number between 0 and 1)
  - The **input conductance (G_{in}, [μS])** is then found by:
    $$G_{in} = \frac{1}{R_{in}}$$
  - By definition (Johnston & Wu, p. 90), we have
    $$\rho = \frac{G_D}{G_S}$$
    The **somatic** and **dendritic conductances (G_S, G_D, [μS])** are then found by:
    $$G_{in} = G_S + G_D = G_S(1 + \rho)$$
    $$\Rightarrow G_S = \frac{G_{in}}{1+\rho}, G_D = \rho G_S$$
  - The **somatic** and **dendritic resistances (R_S, R_D, [MΩ])** are then found by:
    $$R_S = \frac{1}{G_S}, R_D = \frac{1}{G_D}$$
  - The **membrane time constant (τ_m, [ms])** is approximately the same as the first time constant of double exponential fit:
    $$\tau_m = \tau_0$$

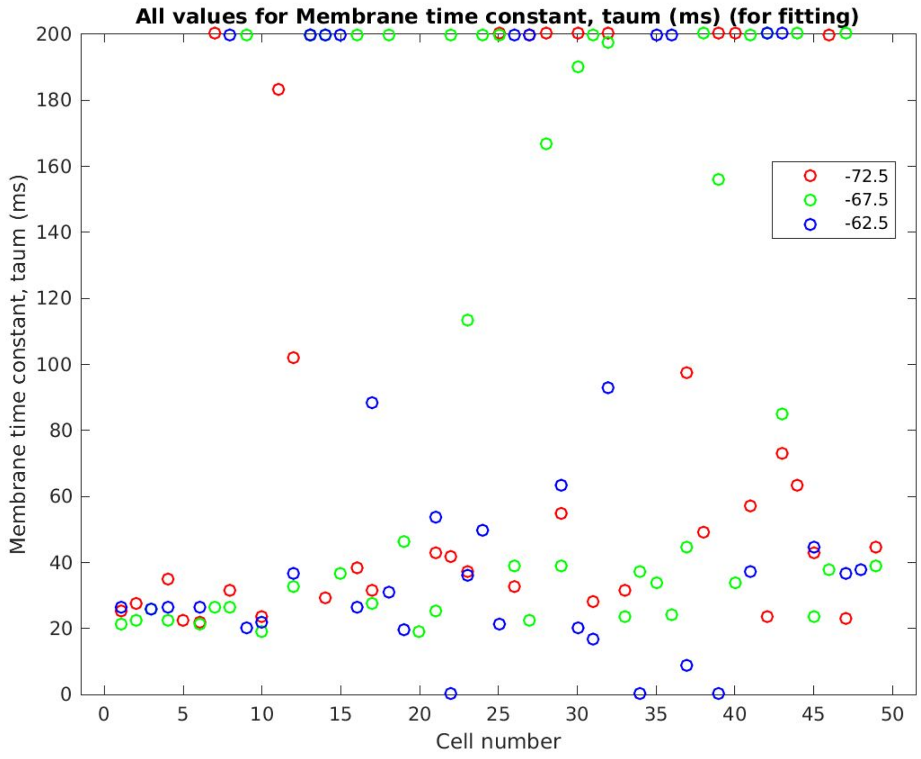
- The **specific membrane resistivity (R_m, [Ω·cm^2])** is then found by
  $$R_m = (10^3 \mathrm{ms/s})\frac{\tau_m}{C_m},$$
  where $C_m = 0.88\ \mu\mathrm{F/cm}^2$ is the **specific membrane capacitance** that's fixed across all cells.
- Modeling the soma as a **sphere**, the membrane resistance of the soma is given by (Johnston & Wu, p. 62):
  $$R_S = \frac{R_m}{4\pi a_S^2}$$
  Thus the **radius of soma (a_S, [μm])** is can be found by:
  $$a_S = (10^4\ \mu\mathrm{m/cm})\sqrt{\frac{R_m}{4\pi R_S (10^6\ \Omega/\mathrm{M}\Omega)}}$$
- Modeling all dendrites as a single **finite cylinder**, the membrane resistance of the dendrite is given by (Johnston & Wu, p. 89):
  $$R_D = \frac{2}{\pi}\sqrt{R_m R_i}(d_D)^{-3/2}\coth(L),$$
  where $R_i = 173\,\Omega\cdot\mathrm{cm}$ is the **axial resistivity** that's fixed across all cells. Thus the diameter of dendrite (**d_D, [μm])** can be found by:
  $$d_D = (10^4\ \mu\mathrm{m/cm})(\frac{2}{\pi}\frac{\sqrt{R_m R_i}\coth(L)}{R_D(10^6\ \Omega/\mathrm{M}\Omega)})^{2/3}$$
  And the **radius of dendrite (a_D, [μm])** can be found by:
  $$a_D = \frac{d_D}{2}$$
- The **space constant (λ, [μm])** is computed by (Johnston & Wu, p. 64):
  $$\lambda = \sqrt{\frac{a_D R_m}{2R_i}}$$
  Thus the **length of dendrite (l_D, [μm])** can be found by:
  $$l_D = \lambda L$$

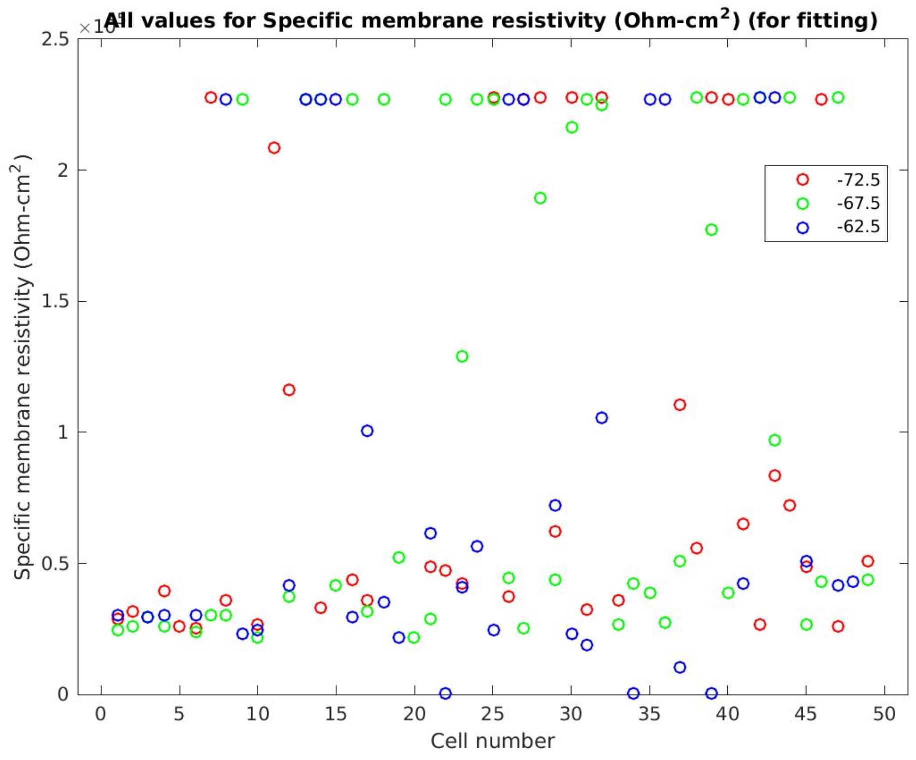

Passive parameter fitting for Cell1_v-62.5 (all)
Total number of sweeps: 102
data
fitted
LPR
Shifted voltage (mV)
Shifted time (ms)
3.5*(exp(-x/9.1)-1)+0.79*(exp(-x/0.3)-1)
Rin = 84.9027 MOhm,
Rin = 84.9015 MOhm,
-1.4*exp(-x/25)-1.2*exp(-x/3.9)
-4.4*exp(-x/25)-1.3*exp(-x/3.9)
Rin = 113.2827 MOhm
Position (um)
L = 0.43529
L = 0.43528
rho_S = 2.2085
Rsoma_S = 272 MOhm
Rdend_S = 123 MOhm
taum_S = 9.0743 ms
taum_S = 9.074 ms
Rm_S = 10311.6913 Ohm-cm^2
Rm_S = 10311.3549 Ohm-cm^2
L_L = 1.2503
rho_L = 10.3924
Rsoma_L = 1291 MOhm
Rdend_L = 124 MOhm
taum_L = 25.371 ms
Rm_L = 28830.7377 Ohm-cm^2

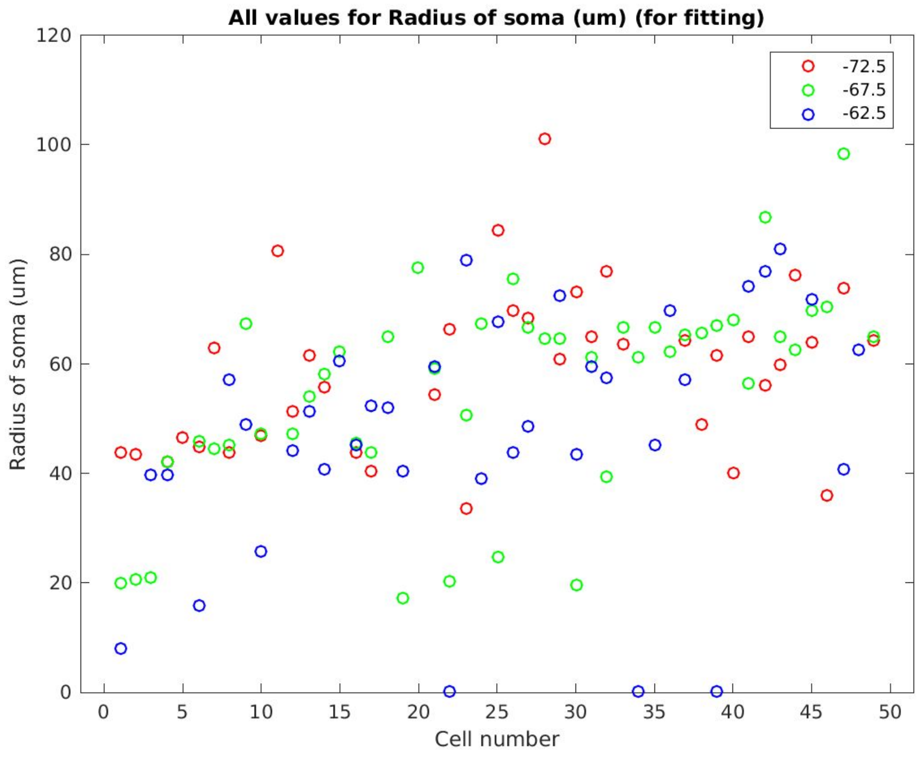

Passive parameter fitting for Cell1_v-67.5 (all)
Total number of sweeps: 99
data
fitted
LPR
Shifted voltage (mV)
Shifted time (ms)
2.9*(exp(-x/9.7)-1)+0.68*(exp(-x/0.29)-1)
Rin = 71.6771 MOhm
Rin = 71.615 MOhm
-1.3*exp(-x/23)-0.91*exp(-x/2.8)
-3.7*exp(-x/23)-0.94*exp(-x/2.8)
Rin = 91.7127 MOhm
Position (um)
L = 0.40742
rho_S = 2.1355
Rsoma_S = 225 MOhm
Rdend_S = 105 MOhm
taum_S = 9.6824 ms
Rm_S = 11002.7126 Ohm-cm^2
L_L = 1.0869
rho_L = 9.7424
Rsoma_L = 985 MOhm
Rdend_L = 101 MOhm
taum_L = 22.7956 ms
Rm_L = 25904.1476 Ohm-cm^2
L = 0.40751
rho_S = 2.1355
Rsoma_S = 225 MOhm
Rdend_S = 105 MOhm
taum_S = 9.6656 ms
Rm_S = 10983.6593 Ohm-cm^2

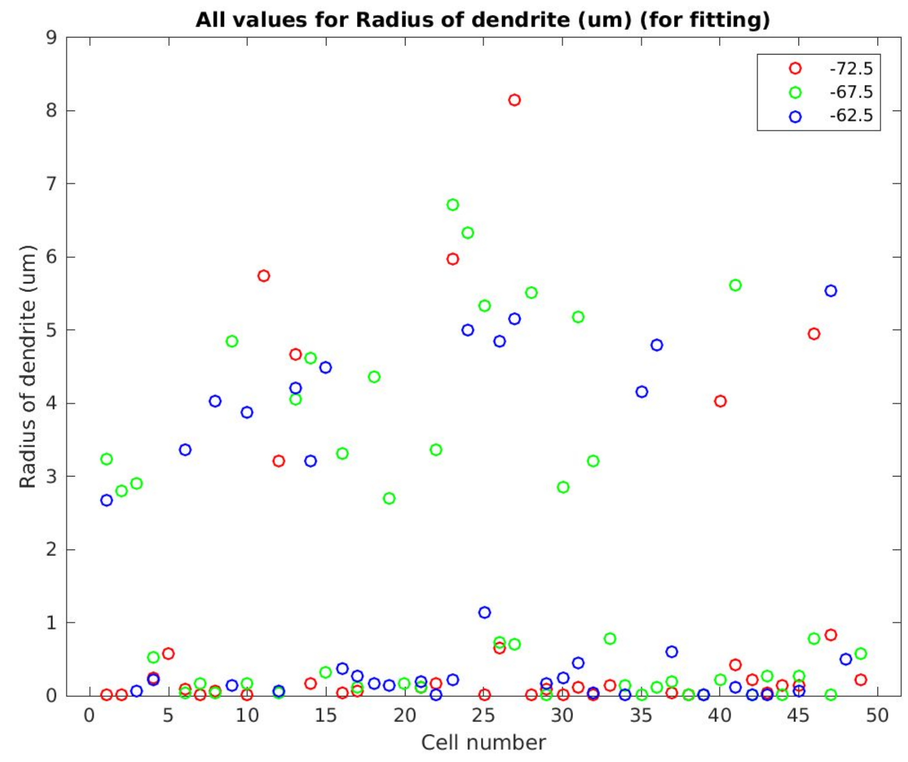

Passive parameter fitting for Cell1_v-72.5 (all)
Total number of sweeps: 99
data
fitted
LPR
Shifted voltage (mV)
Shifted time (ms)
Position (um)
1.5*(exp(-x/6.9)-1)+1.5*(exp(-x/6.9)-1)
Rin = 57.7385 MOhm
5*(exp(-x/20)-1)+0.34*(exp(-x/0.47)-1)
Rin = 105.8538 MOhm
-1.8*exp(-x/29)-0.42*exp(-x/3.1)
-6*exp(-x/29)-0.44*exp(-x/3.1)
Rin = 128.252 MOhm
L = 7956.0443
rho_S = 0.031419
Rsoma_S = 60 MOhm
Rdend_S = 1895 MOhm
taum_S = 6.8793 ms
Rm_S = 7817.3341 Ohm-cm^2
L = 0.40845
rho_S = 4.1955
Rsoma_S = 550 MOhm
Rdend_S = 131 MOhm
taum_S = 19.8075 ms
Rm_S = 22508.5271 Ohm-cm^2
L_L = 0.70926
rho_L = 0.90007
Rsoma_L = 244 MOhm
Rdend_L = 271 MOhm
taum_L = 28.8902 ms
Rm_L = 32829.8163 Ohm-cm^2

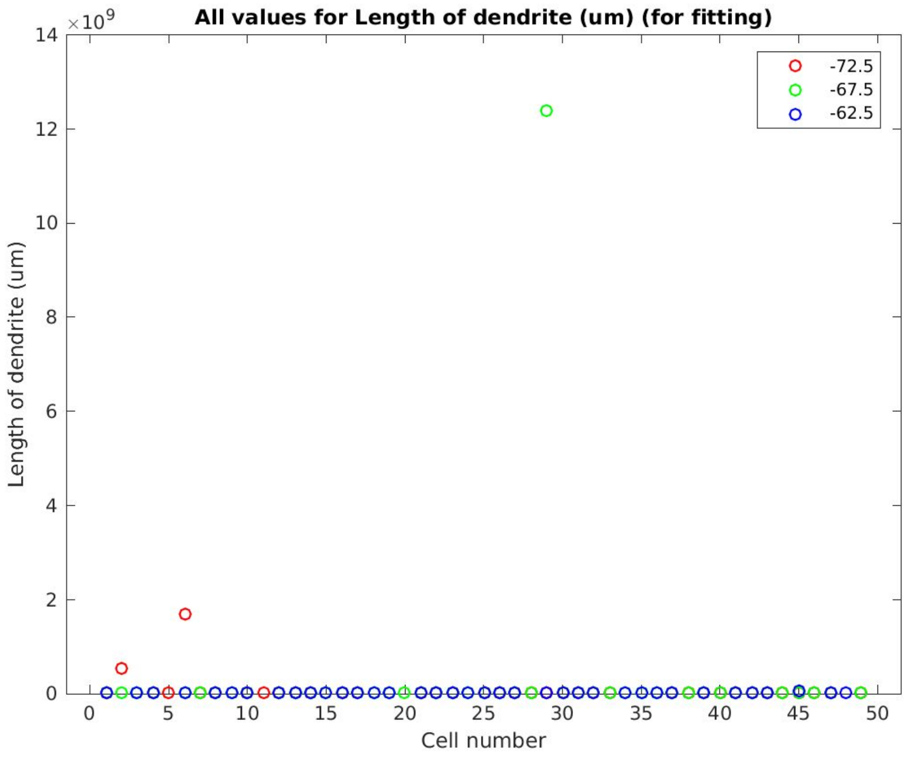

All values for Input resistance (MOhm) (for fitting)
Input resistance (MOhm)
Cell number
-72.5
-67.5
-62.5
0
50
100
150
200
250
300
350
400
450
500
0
5
10
15
20
25
30
35
40
45
50


All values for Membrane time constant, taum (ms) (for fitting)
Membrane time constant, taum (ms)
Cell number
-72.5
-67.5
-62.5
0
20
40
60
80
100
120
140
160
180
200
0
5
10
15
20
25
30
35
40
45
50


All values for Specific membrane resistivity (Ohm-cm^2) (for fitting)
×10^4
2.5
2
1.5
1
0.5
0
Specific membrane resistivity (Ohm-cm^2)
0
5
10
15
20
25
30
35
40
45
50
Cell number
-72.5
-67.5
-62.5


All values for Radius of soma (um) (for fitting)
120
100
80
60
40
20
0
Radius of soma (um)
0
5
10
15
20
25
30
35
40
45
50
Cell number
-72.5
-67.5
-62.5


All values for Radius of dendrite (um) (for fitting)
Radius of dendrite (um)
Cell number
-72.5
-67.5
-62.5


All values for Length of dendrite (um) (for fitting)
×10^9
Length of dendrite (um)
Cell number
-72.5
-67.5
-62.5

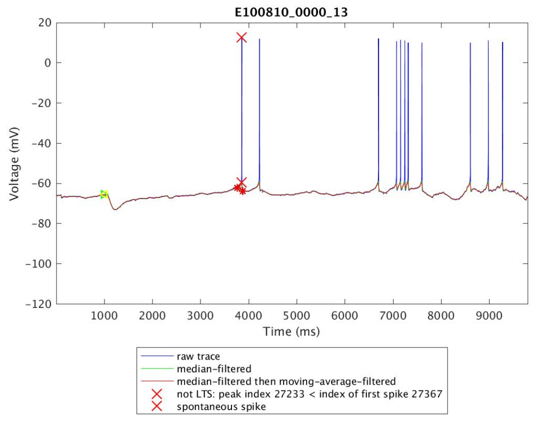
- Created **dclampPassiveFitter.m**:
    - 2016-11-08 Changed from using Vhold to using **actVhold** for binning
    - 2016-11-10 Nows saves set info linearly in **_set** and 2-dimensionally in **_cv**
    - 2016-11-10 Added fn_set & fn_cv

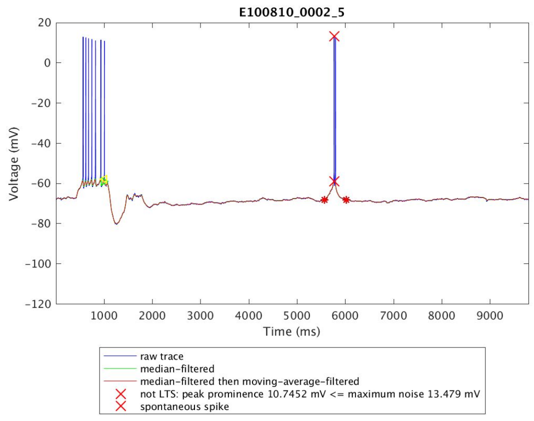
- Modifications to **find_passive_params.m**:
    - 2016-10-31 Changed equation form of ft to 'a*(exp(-x/b)-1)+c*(exp(-x/d)-1)' from 'a*exp(-x/b)+c*exp(-x/d)+e'
    - 2016-10-31 Removed '- round(0.5/sims)' from the definition of base_ind
    - 2016-10-31 Made sure tau0 >= tau1
    - 2016-11-01 Added **fitmode** so that the titles and filenames can change
    - 2016-11-01 Exclude faulty traces, including those with spontaneous spikes, from the fitting
    - 2016-11-01 Added pulse width
    - 2016-11-01 Added long pulse response
    - 2016-11-01 Added L, rho, taum
    - 2016-11-01 Plot ivec1s only (ivec0s is not directly used in the analyses)
    - 2016-11-02 Moved check directories to **check_directories.m**
    - 2016-11-02 Added the falling phase of the current pulse response
    - 2016-11-12 Now returns estimates from pooled data if those from averaged data don't exist
    - 2016-11-12 Now outputs all estimates

10/24/2016~11/13/2016

<u>Data Analysis (cont'd)</u>

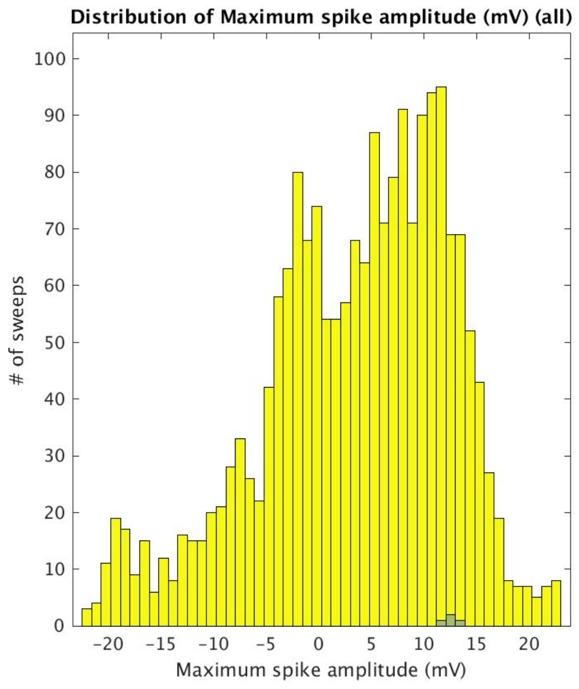
- Modifications to **fit_gaussians_and_refine_threshold.m**:
    - 2016-11-01 Replaced ProbDistUnivParam with **makedist**, also from the Statistics and Machine Learning Toolbox
- Modifications to **PlotHistogramsRefineThreshold.m**:
    - 2016-10-31 - Placed suffix into **SpecsForFitmode.m**
    - 2016-11-10 - Expanded to use multiple files, added **passive params**
- Created **find_special_cases.m** that looks for special cases and put traces in corresponding folder
    - For instance, all traces with spontaneous spikes but not LTSs

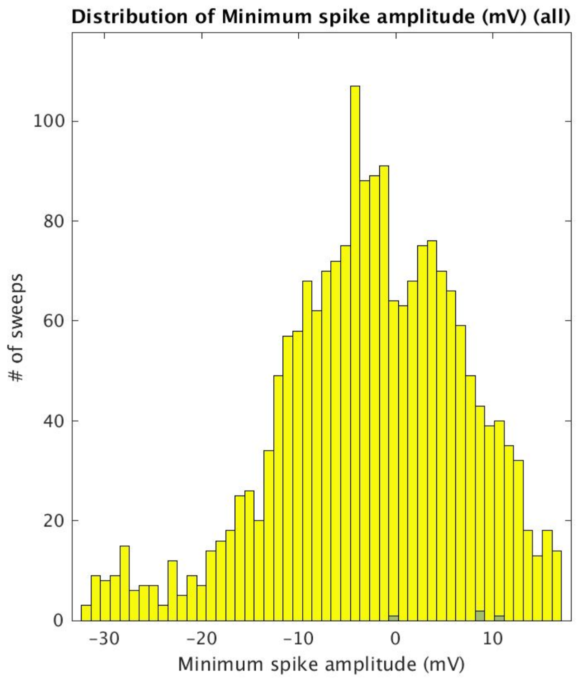


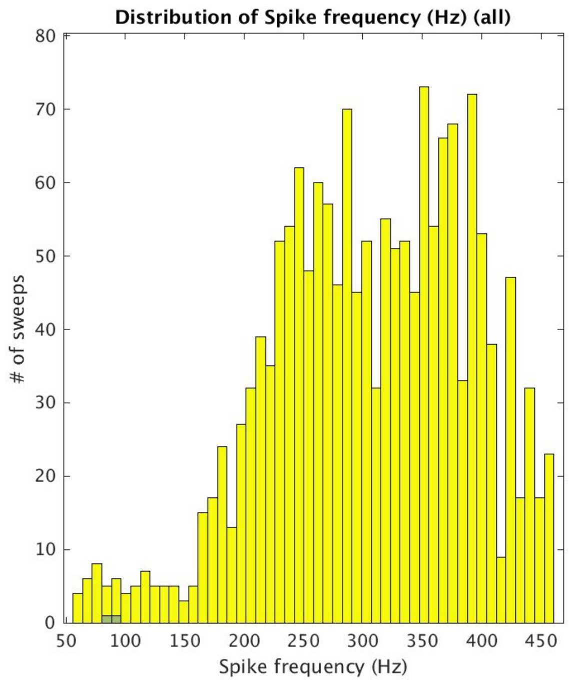


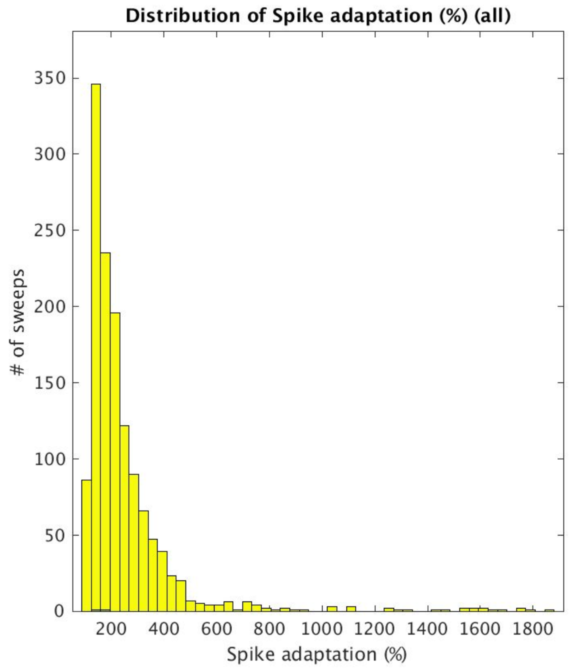
- Modifications to **dclampDataExtractor.m**:
    - 2016-10-27 BT - Added '**maxspikeamp**', '**minspikeamp**', '**spikefrequency**', '**spikeadaptation**'

        maxspikeamp ("Maximum action potential amplitude (mV)",
        set to NaN if not a burst)
        minspikeamp ("Minimum action potential amplitude (mV)",
        set to NaN if not a burst)
        spikefrequency ("Spike frequency (Hz)", set to NaN if not a burst)
        spikeadaptation ("Spike adaptation (%)", ratio of last ISI to first ISI,
        set to NaN if not a burst)

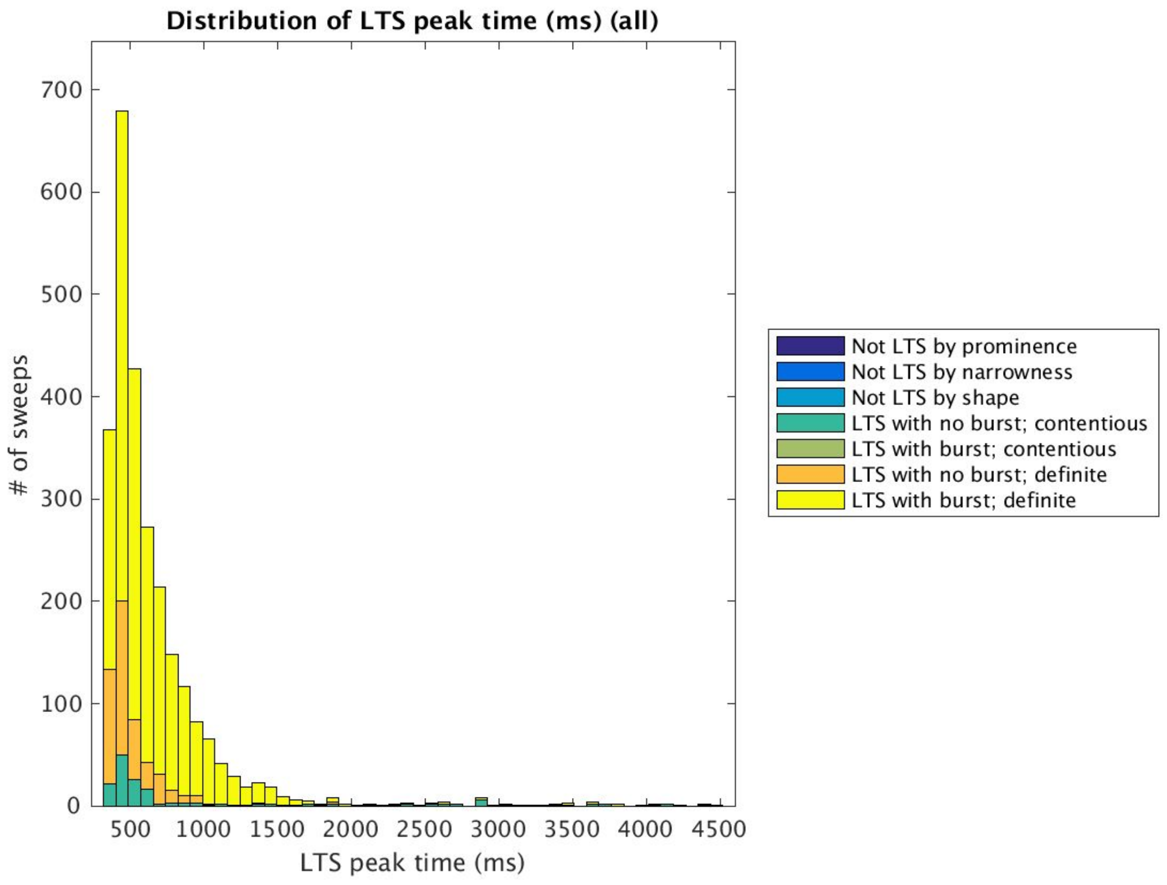


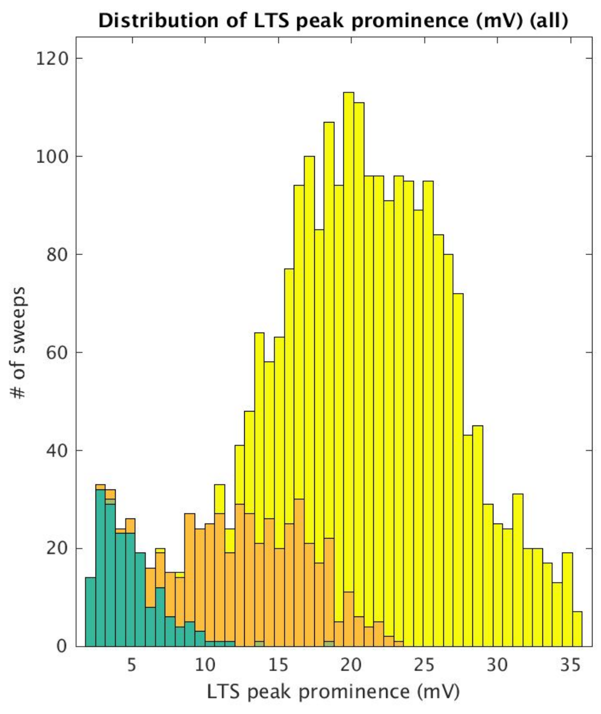


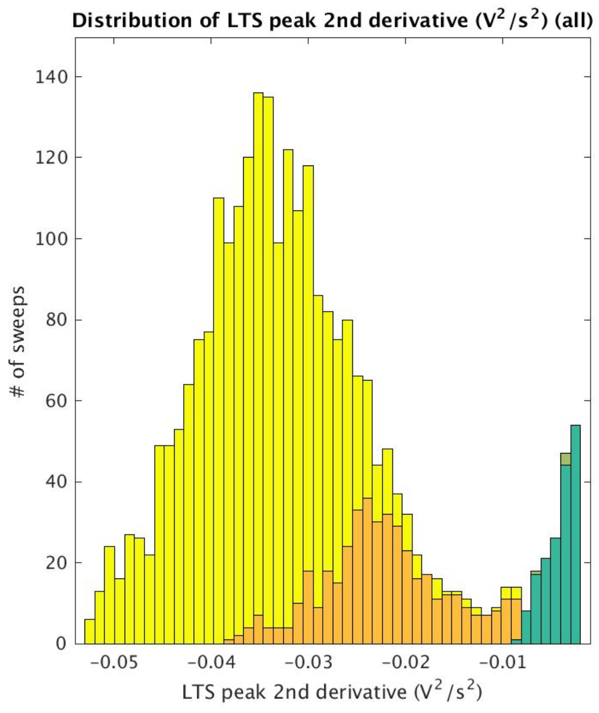


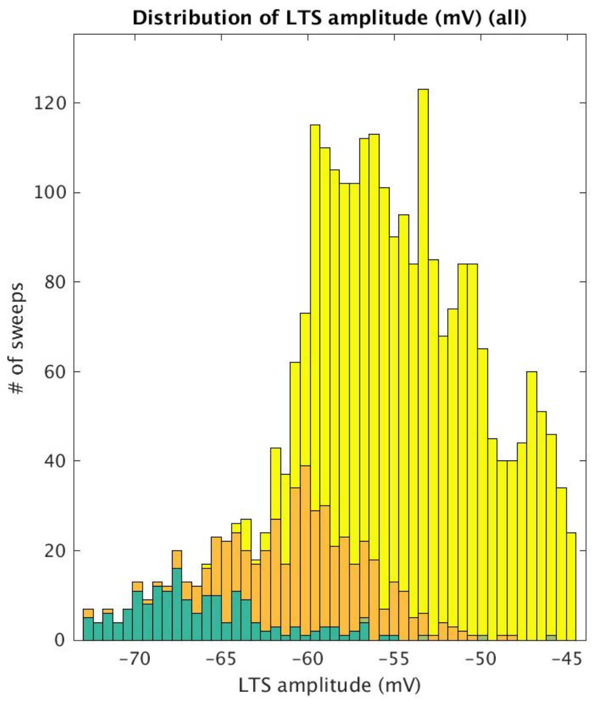


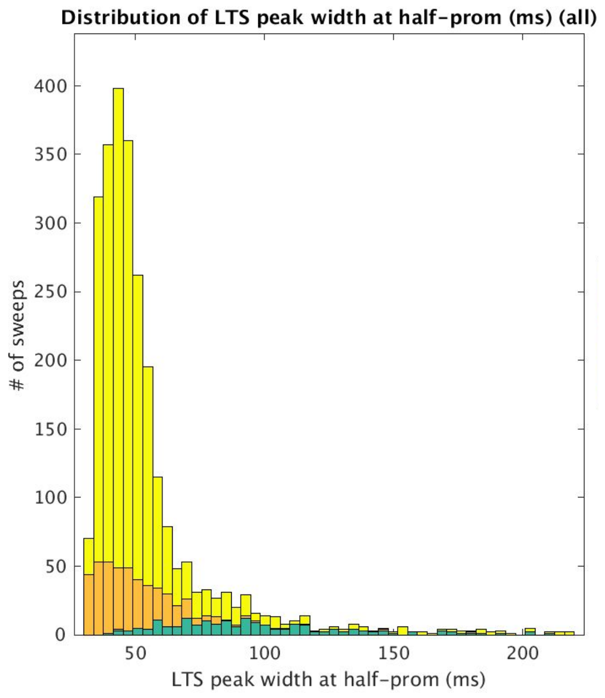
- 2016-10-27 Combined all the Rinput detection to **find_passive_params.m** under /home/Matlab/Adams_Functions
- 2016-10-31 Renamed narrowpeaktime & narrowpeak2ndder as **peaktime** & **peak2ndder**
- 2016-10-31 Added **GenerateLTSInfo.m**, which generates vectors of peak features restricted to those with LTS


Distribution of LTS peak time (ms) (all)
# of sweeps
LTS peak time (ms)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite


Distribution of LTS peak prominence (mV) (all)
# of sweeps
LTS peak prominence (mV)


Distribution of LTS peak 2nd derivative (V^2/s^2) (all)
# of sweeps
LTS peak 2nd derivative (V^2/s^2)

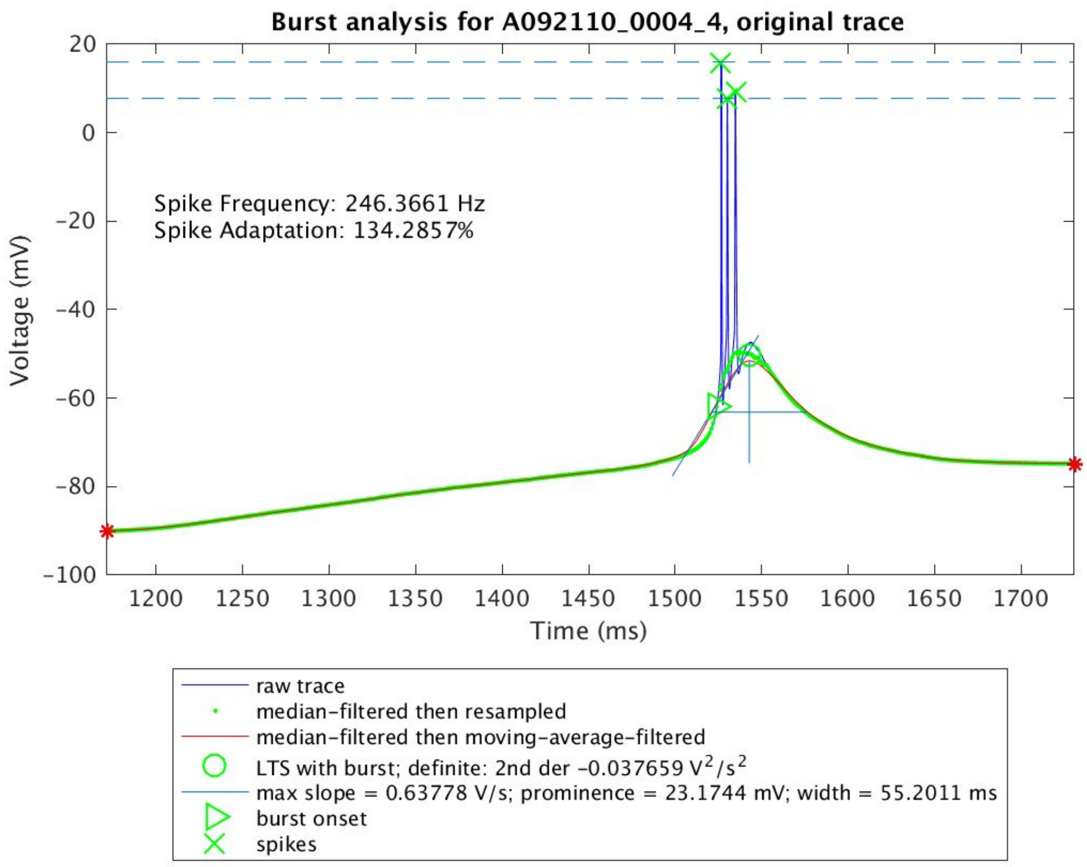




- 2016-11-01 Added **compute flags**
- 2016-11-01 Added **plotpassive2flag** (for **dclampPassiveFitter.m**)
- 2016-11-07 Added **cpa_ap**, **g_sc**, **i_sc**
- 2016-11-07 Reorganized code to make more efficient; created **CountSweeps.m** & **ResaveSweeps.m**

- Created **find_special_cases.m**:
  - Copy traces from /**vtraces_scaled**/ only
  - The traces corresponding to each set of special cases is not only copied to a distinct folder, but also checked against those already in the **CONTESTED_*** folders and **OVERRIDE_*** folders -- any nonexistent trace is copied into **unclassified**

- Modifications to **find_LTS.m**:
  - 2016-10-24 BT - Added **lines** for **minimum and maximum spike amplitudes**
  - 2016-10-27 BT - Changed spike frequency to be computed from **peak to peak**
  - 2016-10-27 Replace each directory with directories{k}
  - 2016-10-31 BT - Added **line for maxslope**
  - 2016-11-01 AL - Fixed **peakwidth** so that it's in **ms**
  - 2016-11-01 BT - Added **lines** for **peakprom** and **peakwidth**
  - 2016-11-02 AL - Moved check directories to **check_directories.m**
  - 2016-11-02 AL - Changed maxslope_label to **peakfeature_label**



- Brian helped create **PlotCorrelations.m**:
    - 2016-10-13 - Created by **BT**, adapted from **PlotHistogramsRefineThreshold.m**
    - 2016-10-17 - BT - Changed scatterplot to color points by peak class and added legend.
    - 2016-10-18 - BT - Changed color scheme of existing figures.
    - 2016-10-20 - BT - Changed importing of vectors to all vectors within mat file
    - 2016-10-20 - BT - Made correlations combinatorial and to generate in parfor
    - 2016-10-20 - BT - Added optional parameters for infolder and outfolder
    - 2016-10-20 - AL - Made variables consistent with PlotHistogramsRefineThreshold.m
    - 2016-10-20 - AL - Fixed error when ioffset_old was not skipped during plotting
    - 2016-10-20 - AL - Preallocate memory
    - 2016-10-21 - AL - Added **cl_exists** to account for the case when peakclass does not include all classes
    - 2016-10-31 - AL - Placed suffix & title_mod into SpecsForFitmode.m
    - 2016-11-03 - BT - Added label for **correlation coefficient**
    - 2016-11-08 - BT - Copy correlation plots with |correlation coefficient| above threshold (**corr_thr**) in a subdirectory called “**interesting**”
    - 2016-11-09 - AL - Now saves correlation matrix to a **mat file** in outfolder

11/15/2016~12/2/2016

<u>Meeting with John and later discussion with Mark</u>

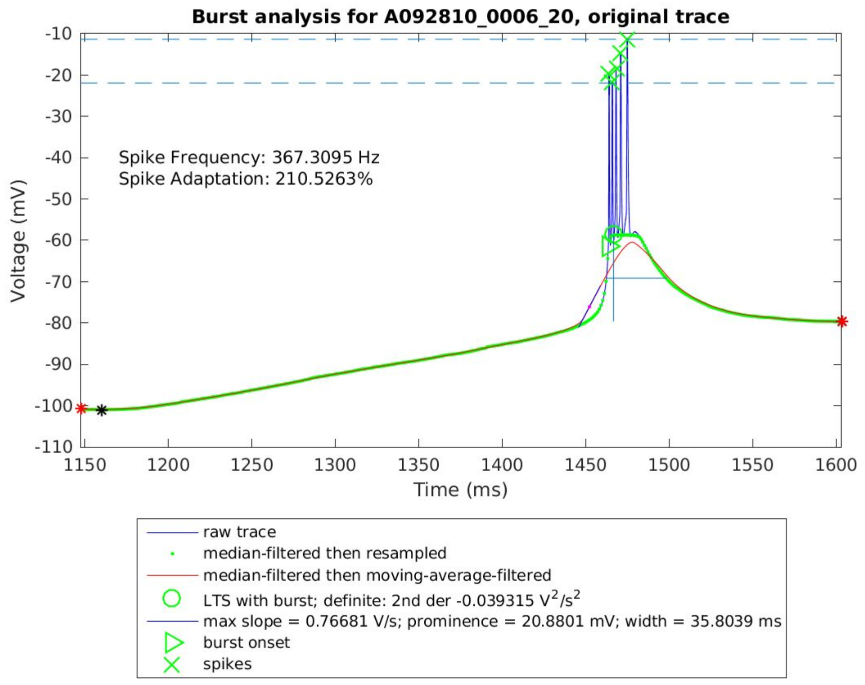
- Trace comparison
    - How should we account for series resistance? Could it explain the discrepancy between expected and recorded currents?
        - John: The discrepancy probably did not result from a series resistance correction, but from a **fixed voltage offset per cell**. Series resistance can be fixed at **10 MΩ.**
- Data analysis
    - LTS peak features
        - John: **peaktime**, **maxslope**, **peakprom**, **peakwidth** should be computed relative to the median-filtered trace, not to the moving-average-filtered trace.
- Passive fitting
    - The grouping by holding potential didn't seem to show a pattern in any of the passive parameters estimated. Should we just pool all traces of the same cell together?
        - John: Ok
    - Do we average the traces, then calculate sum of squares error, or calculate sum of squares error over all traces?
        - John: Averaging the traces is fine
    - If we make **Ra** a constant across cells, should we vary the length **L** of each compartment instead?
        - John: Yes
    - How to deal with noisy traces?
        - John: Use a goodness-of-fit threshold to take out certain traces, or **weight** the traces by some goodness-of-fit measure
        - Adam: A **sum-of-squares error** should be computed relative the fit for each individual trace.
    - How to deal with **systematic error** (possibly caused by synaptic events)?
        - John: The falling phase shouldn't be that long as steady state is reached early on
        - Mark: Fitting a period of **150 ms** should suffice
    - Some curves are fitted with a single exponential only
        - John: Force the two time constants to lie within **different ranges**
        - Mark: Find two **fixed bounds**
        - Adam: Plot a **histogram** of current double exponential fits to determine the ranges needed

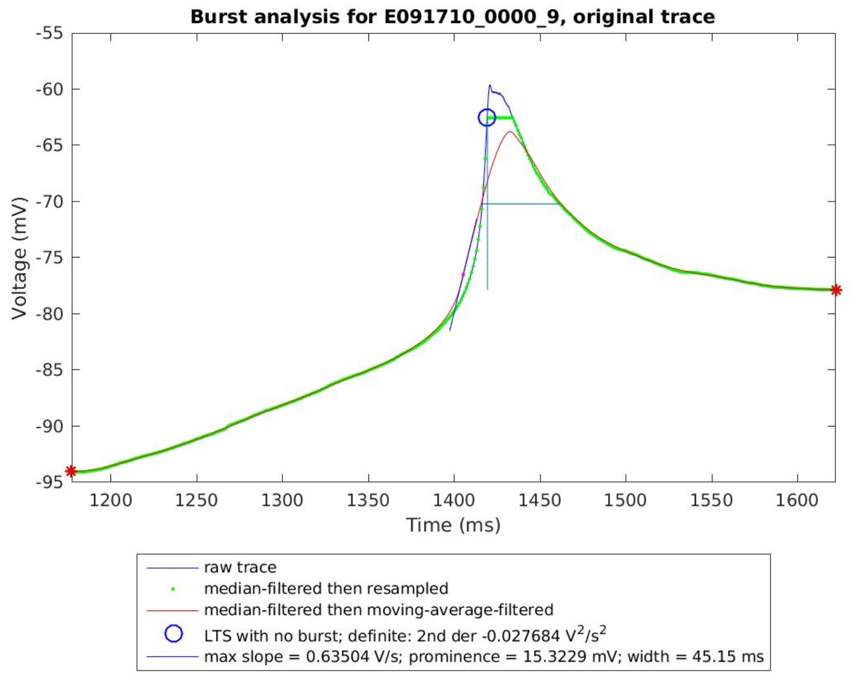
11/17/2016~12/2/2016

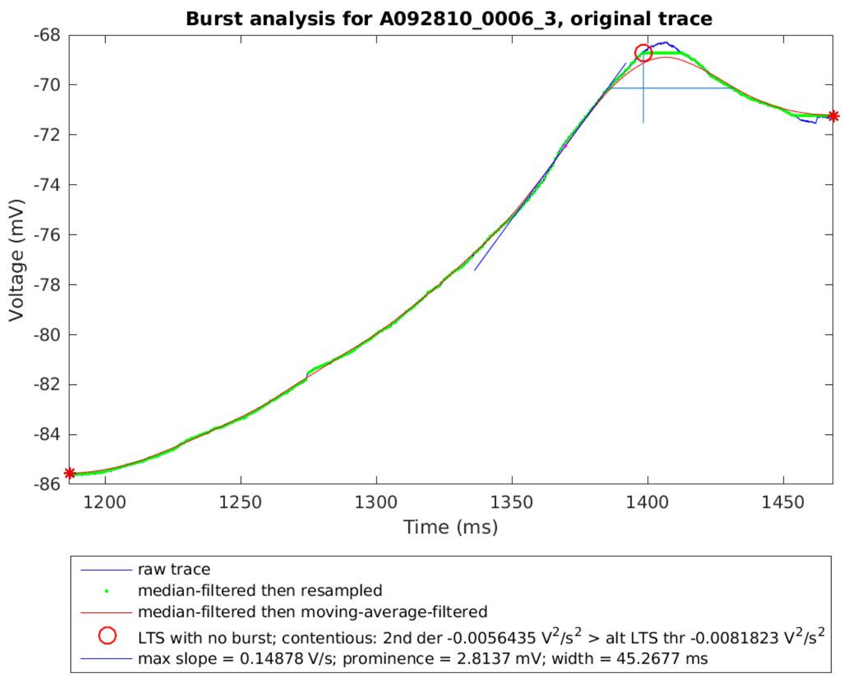
<u>**Data Analysis (cont'd)**</u>

- Modifications to **find_LTS.m**:
    - 2016-11-17 BT & AL - Changed peaktime, peakprom, peakwidth to be computed from the **median-filtered trace** instead of the moving-average-filtered trace
    - 2016-11-17 BT - Changed the plots of peaktime, peakprom, peakwidth accordingly
    - 2016-11-21 BT - Changed maxslope to be computed from the **median-filtered trace** instead of the moving-average-filtered trace
    - 2016-11-29 BT - Added **spacing_size**; Changed maxslope to be computed over **1 ms** instead of over 0.1 ms
    - 2016-11-29 AL - Added **slope_spacing**; Changed maxslope to be computed with **vector manipulation** for performance
    - 2016-11-29 AL - Made sure **maxslope**, **peakprom**, **peakwidth** was plotted on LTSs without bursts as well
    - 2016-11-29 AL - Changed the limits of the maxslope line segment so it would show up at the right place
    - 2016-11-29 AL - Added **mafw3**, **vvec4** & **v4** for finding maximum slope
    - 2016-11-30 AL - Added **slopesegyhalf** for plotting maximum slope
    - 2016-11-30 AL - Added **mafw3_dv**, changed mafw3, v4 to depend on the maximum slope found from v3

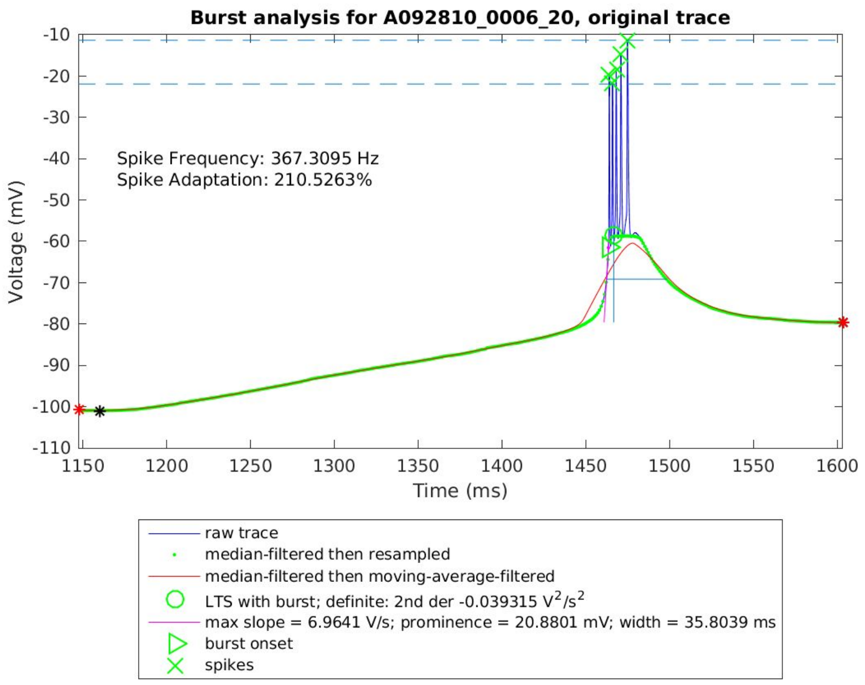
- Different ways of finding **maxslope**, all using a **secant line spacing** of **1 ms**:
    - Using a **moving-average-filtered** voltage trace with a filter width of **30 ms** (original):

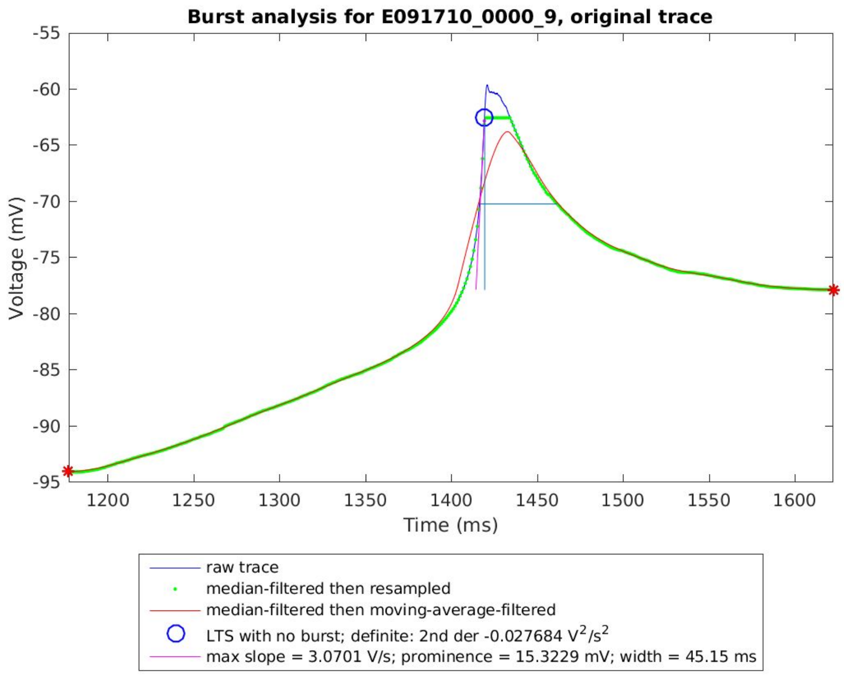


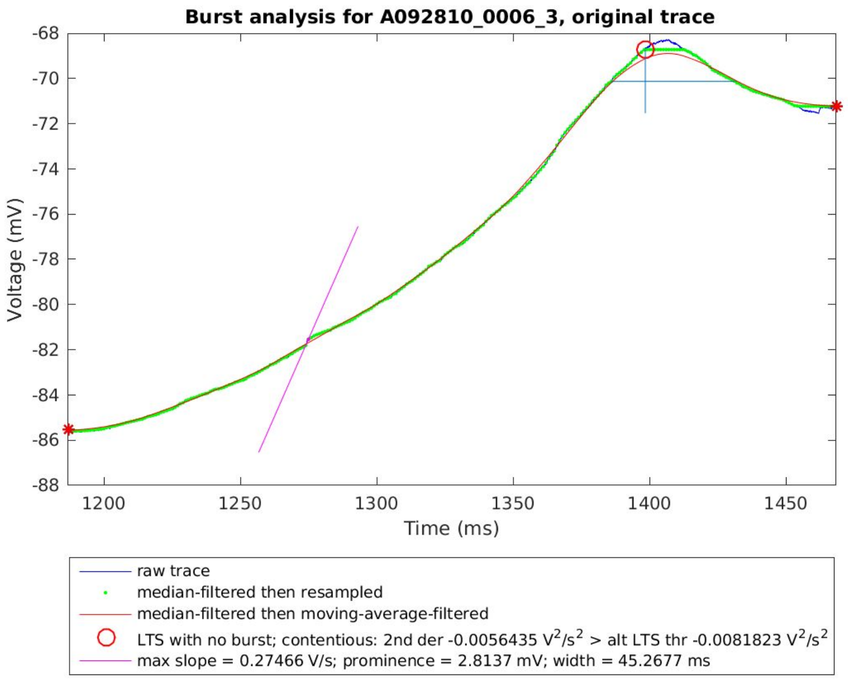




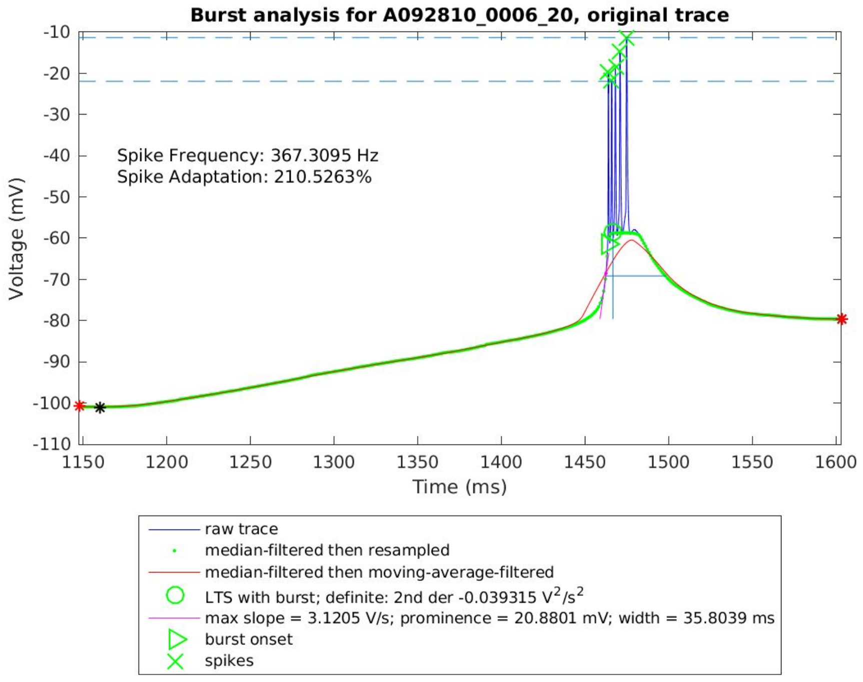
- Using the **median-filtered** voltage trace (filter width **30 ms**):

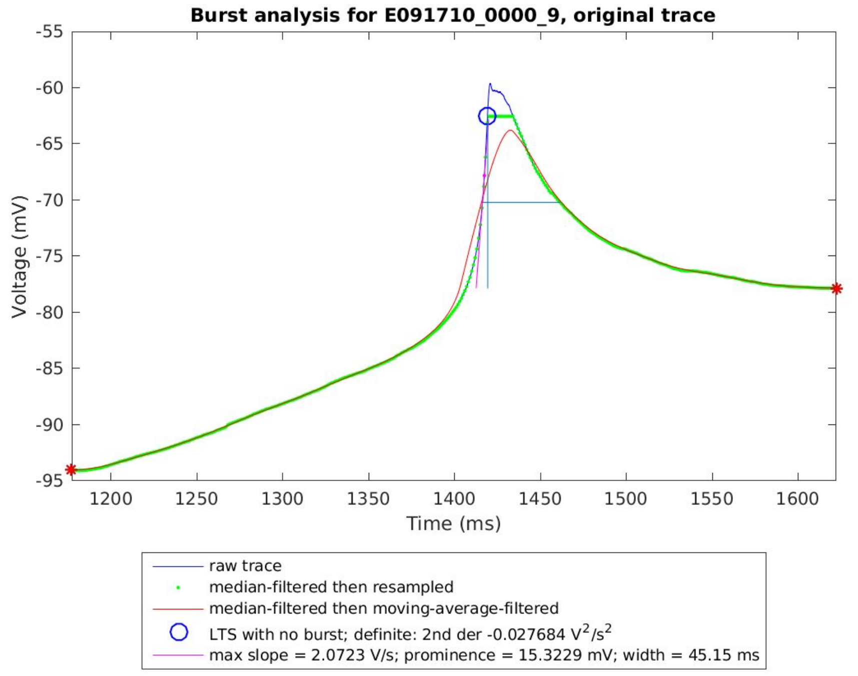


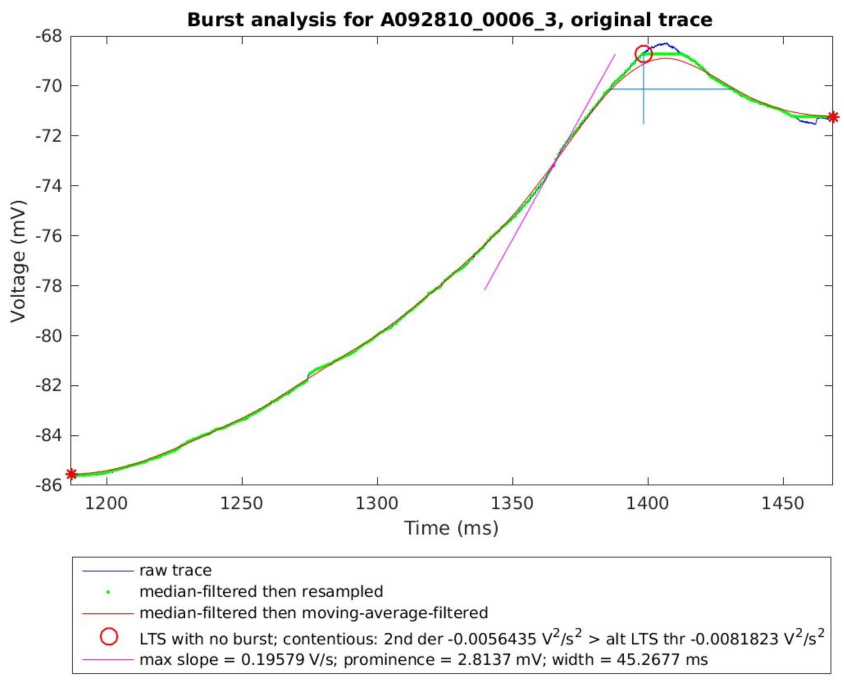

Burst analysis for E091710_0000_9, original trace
Voltage (mV)
Time (ms)
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; definite: 2nd der -0.027684 V²/s²
max slope = 3.0701 V/s; prominence = 15.3229 mV; width = 45.15 ms


Burst analysis for A092810_0006_3, original trace
Voltage (mV)
Time (ms)
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; contentious: 2nd der -0.0056435 V²/s² > alt LTS thr -0.0081823 V²/s²
max slope = 0.27466 V/s; prominence = 2.8137 mV; width = 45.2677 ms

- Using a **moving-average-filtered** voltage trace with a filter width of **5 ms**:



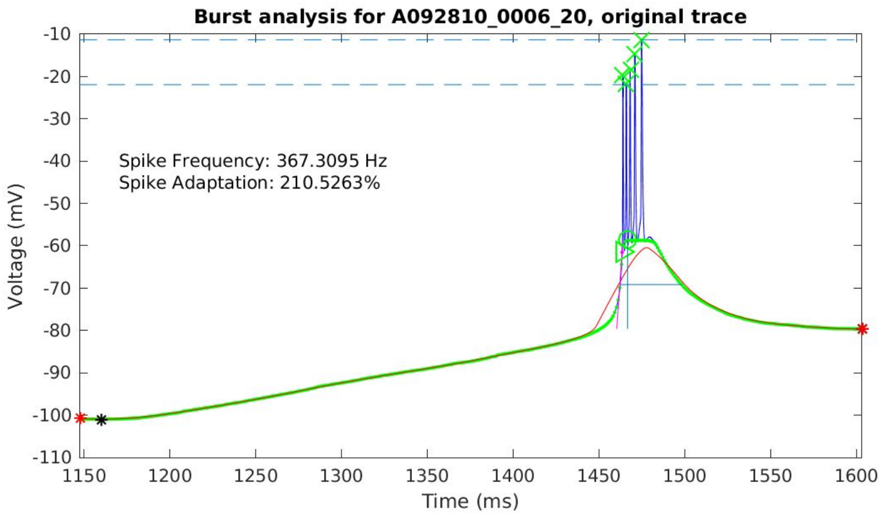




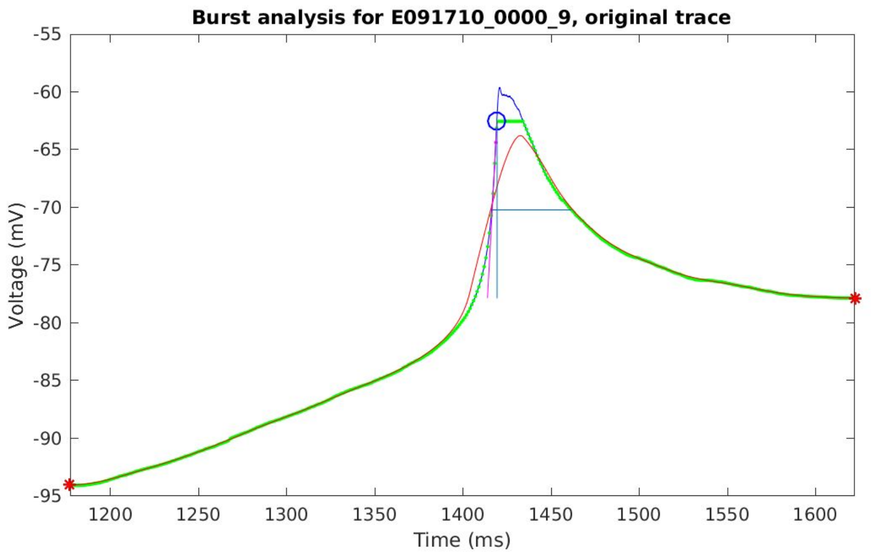
- Can it get any better?
    - Use an **average of slopes**?
        - The slopes are discontinuous when bursts occur, so averaging out would underestimate the maximum slope by too much
    - Increase **filter width**?
        - This will make the slope be less sensitive to noise for LTSs without bursts; however, it will underestimate the maximum slope for LTSs with bursts
    - Increase **slope_spacing**?
        - Again, this will make the slope be less sensitive to noise for LTSs without bursts; however, it will underestimate the maximum slope for LTSs with bursts. Furthermore, it cannot capture the **instantaneous slope** regardless of the steepness.
- Current solution: use a **moving-average-filtered** voltage trace with a filter width (**mafw3**) that changes for each trace and is dependent on a previous estimate of the maximum slope (**maxslopeval_appr**) based on the **moving-average-filtered** voltage trace with filter width **30 ms** (**v3**) and a fixed parameter **mafw3_dv** by:

    mafw3 = mafw3_dv / maxslopeval_appr

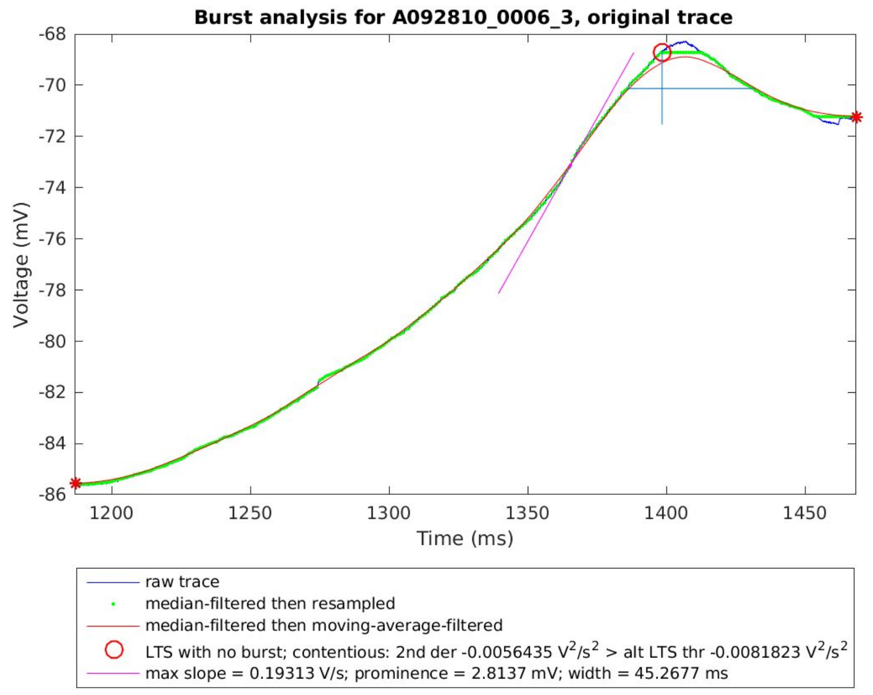
- mafw3_dv == **1 mV**



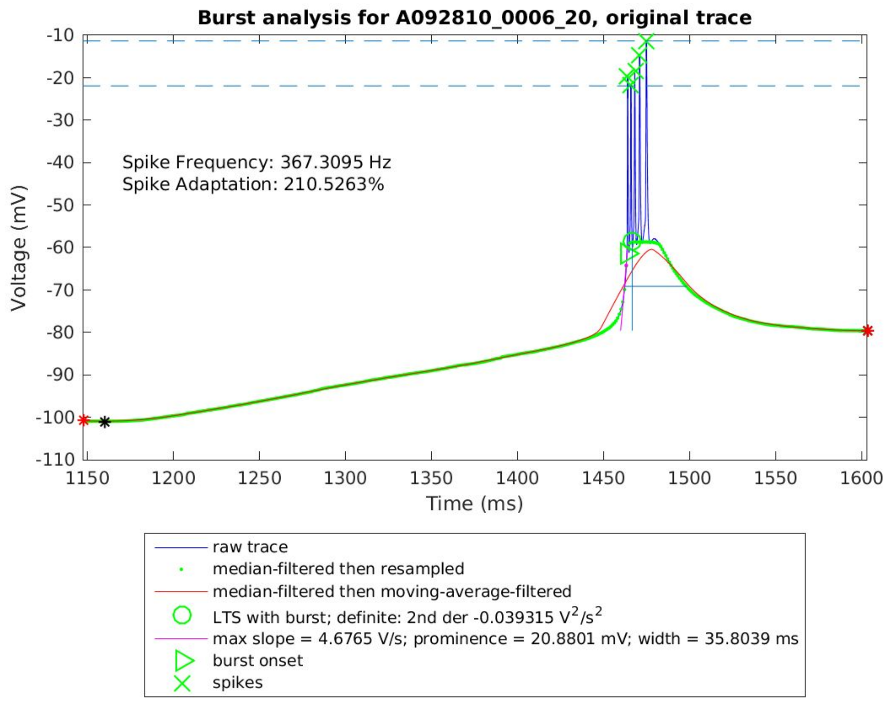
—	raw trace
•	median-filtered then resampled
—	median-filtered then moving-average-filtered
○	LTS with burst; definite: 2nd der -0.039315 V^2/s^2
—	max slope = 5.811 V/s; prominence = 20.8801 mV; width = 35.8039 ms
▷	burst onset
×	spikes

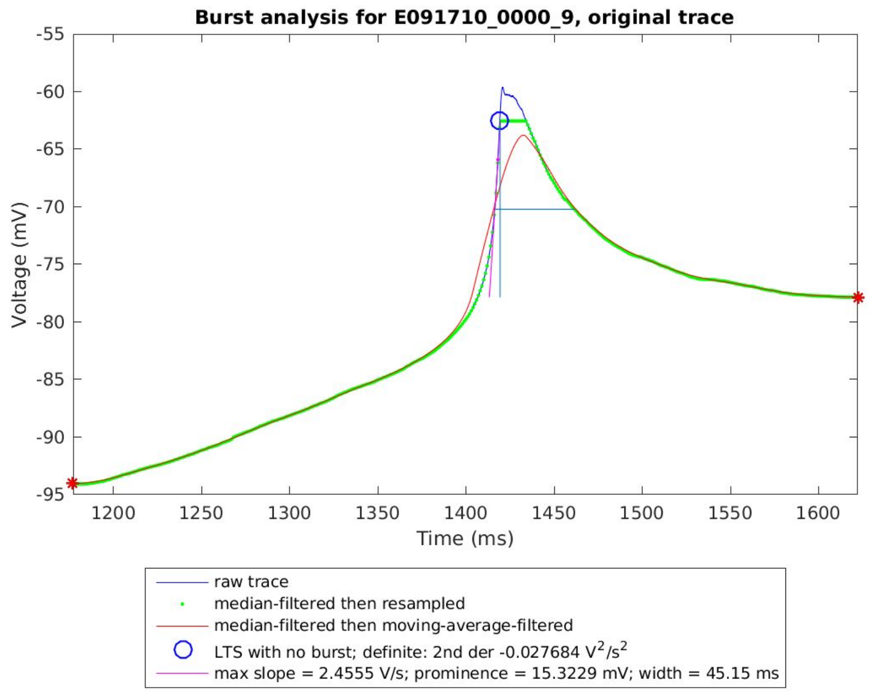


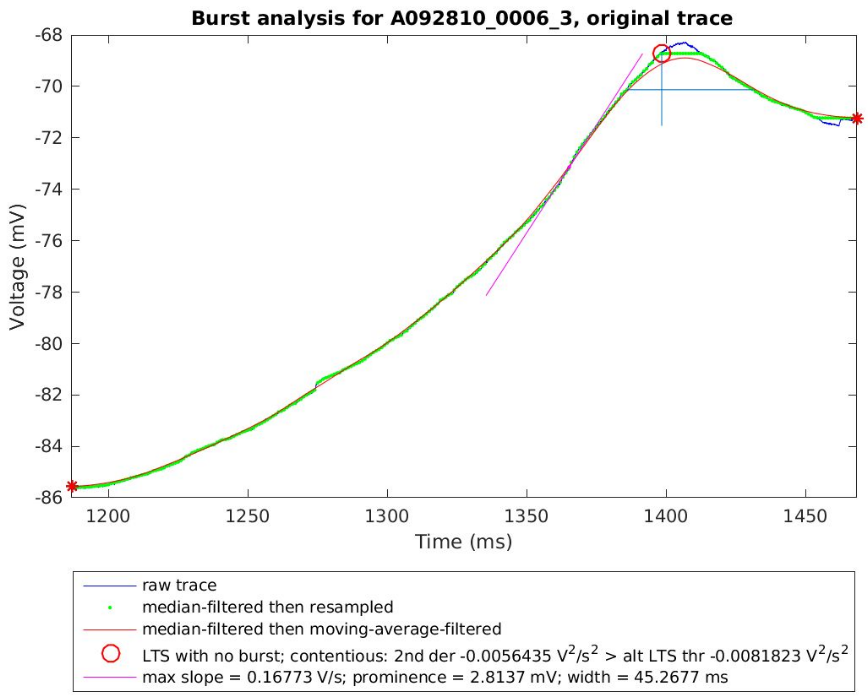
—	raw trace
•	median-filtered then resampled
—	median-filtered then moving-average-filtered
○	LTS with no burst; definite: 2nd der -0.027684 V^2/s^2
—	max slope = 2.8524 V/s; prominence = 15.3229 mV; width = 45.15 ms


Burst analysis for A092810_0006_3, original trace
Voltage (mV)
Time (ms)
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; contentious: 2nd der -0.0056435 V²/s² > alt LTS thr -0.0081823 V²/s²
max slope = 0.19313 V/s; prominence = 2.8137 mV; width = 45.2677 ms

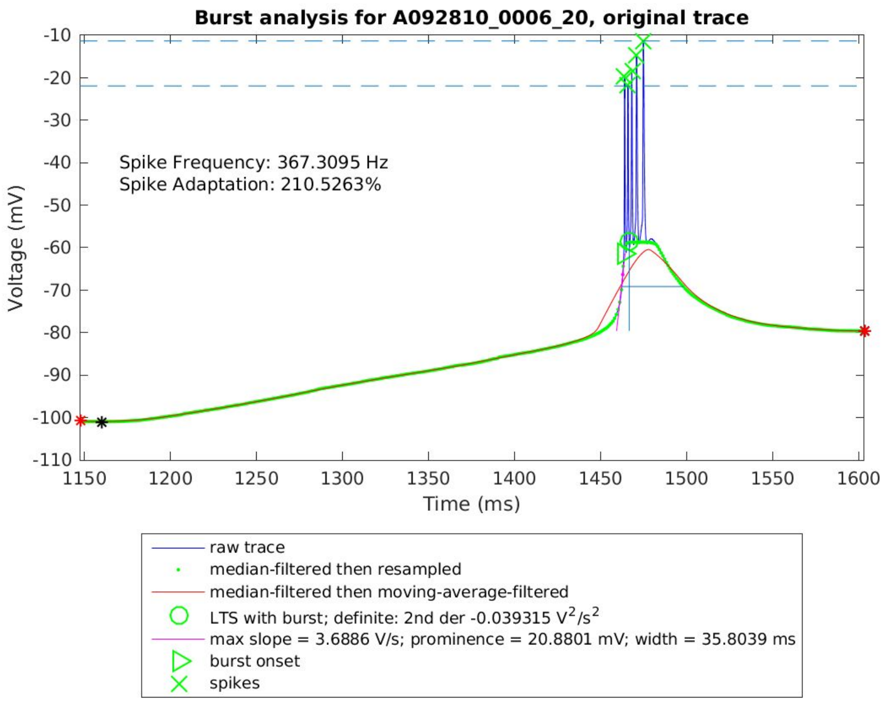
- mafw3_dv == **2 mV**

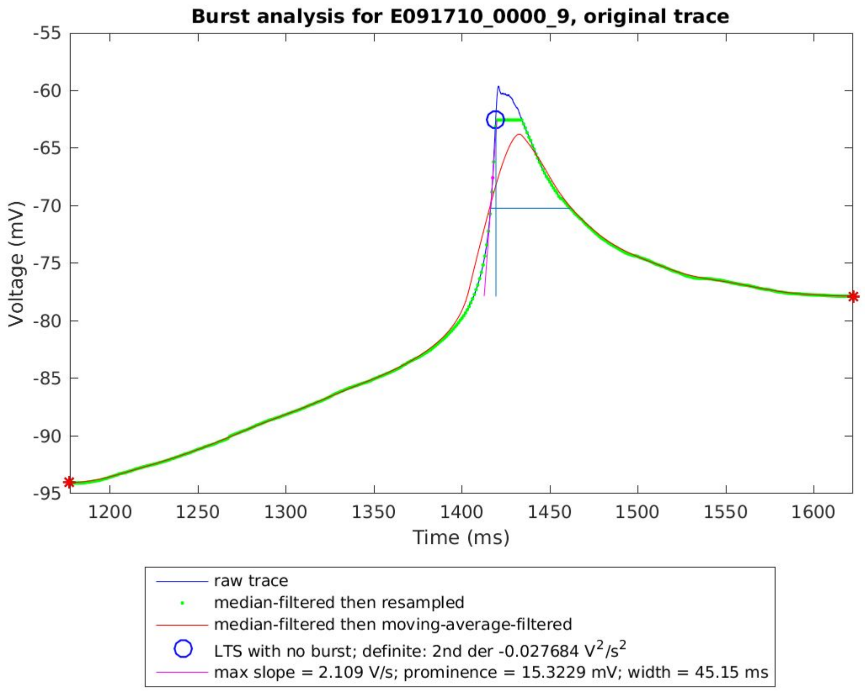

Burst analysis for A092810_0006_20, original trace
Spike Frequency: 367.3095 Hz
Spike Adaptation: 210.5263%
Voltage (mV)
Time (ms)
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with burst; definite: 2nd der -0.039315 V²/s²
max slope = 4.6765 V/s; prominence = 20.8801 mV; width = 35.8039 ms
burst onset
spikes

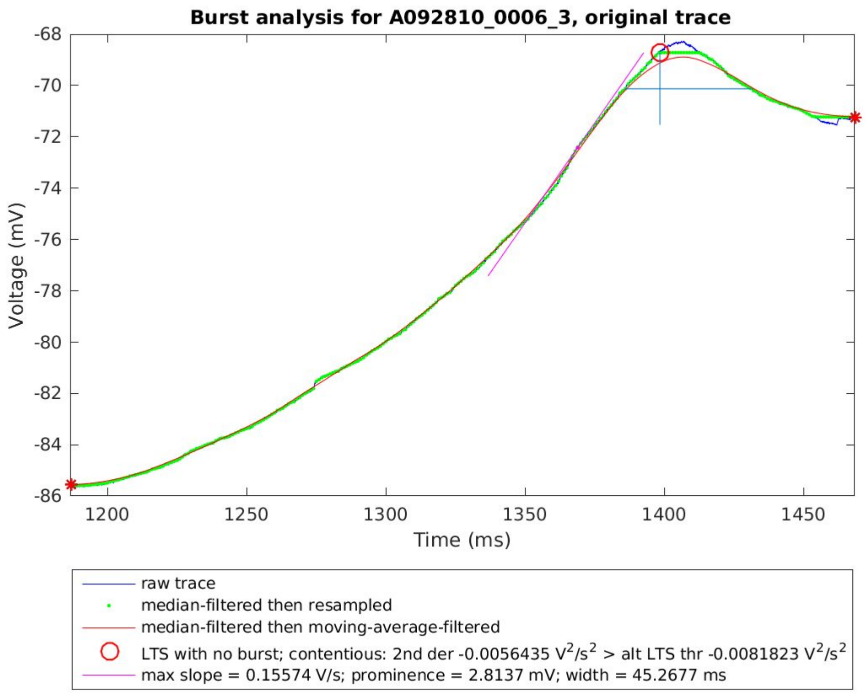

Burst analysis for E091710_0000_9, original trace
Voltage (mV)
Time (ms)
-55
-60
-65
-70
-75
-80
-85
-90
-95
1200
1250
1300
1350
1400
1450
1500
1550
1600
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; definite: 2nd der -0.027684 V²/s²
max slope = 2.4555 V/s; prominence = 15.3229 mV; width = 45.15 ms


Burst analysis for A092810_0006_3, original trace
Voltage (mV)
Time (ms)
-68
-70
-72
-74
-76
-78
-80
-82
-84
-86
1200
1250
1300
1350
1400
1450
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; contentious: 2nd der -0.0056435 V²/s² > alt LTS thr -0.0081823 V²/s²
max slope = 0.16773 V/s; prominence = 2.8137 mV; width = 45.2677 ms

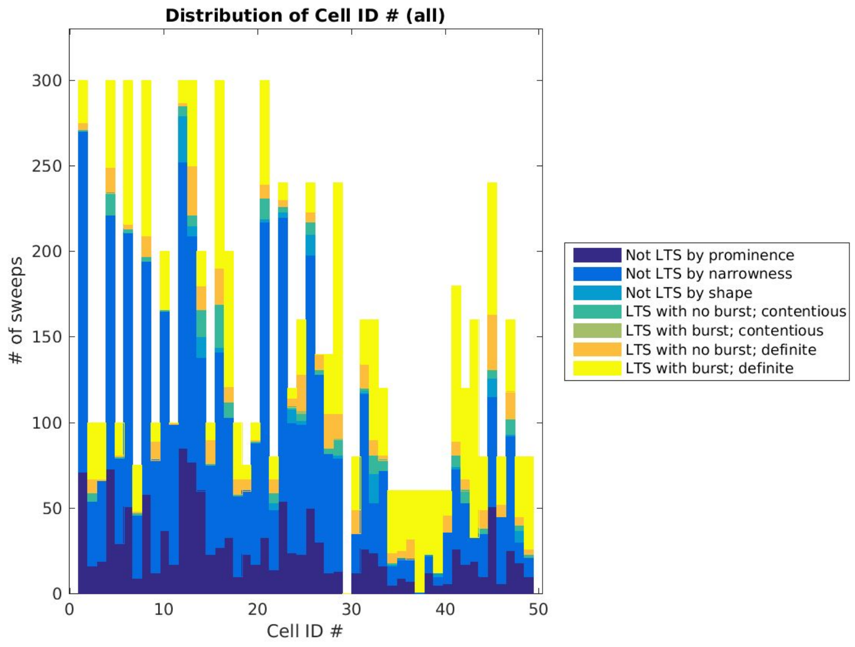
- mafw3_dv == **3 mV**

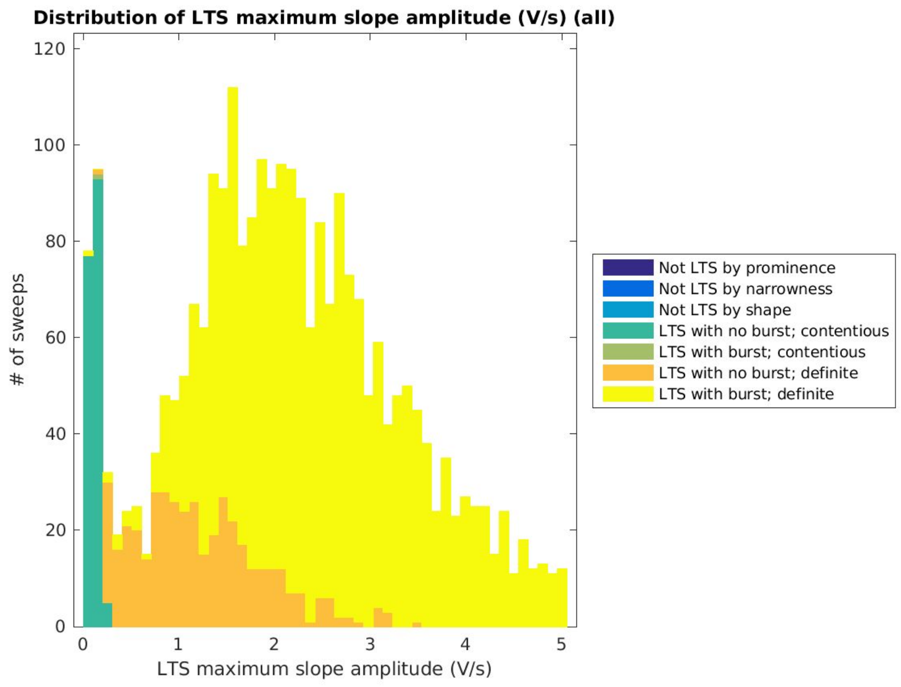

Burst analysis for A092810_0006_20, original trace
Spike Frequency: 367.3095 Hz
Spike Adaptation: 210.5263%
Voltage (mV)
Time (ms)
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
1150
1200
1250
1300
1350
1400
1450
1500
1550
1600
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with burst; definite: 2nd der -0.039315 V²/s²
max slope = 3.6886 V/s; prominence = 20.8801 mV; width = 35.8039 ms
burst onset
spikes


Burst analysis for E091710_0000_9, original trace
Voltage (mV)
Time (ms)
-55
-60
-65
-70
-75
-80
-85
-90
-95
1200
1250
1300
1350
1400
1450
1500
1550
1600
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
LTS with no burst; definite: 2nd der -0.027684 V²/s²
max slope = 2.109 V/s; prominence = 15.3229 mV; width = 45.15 ms

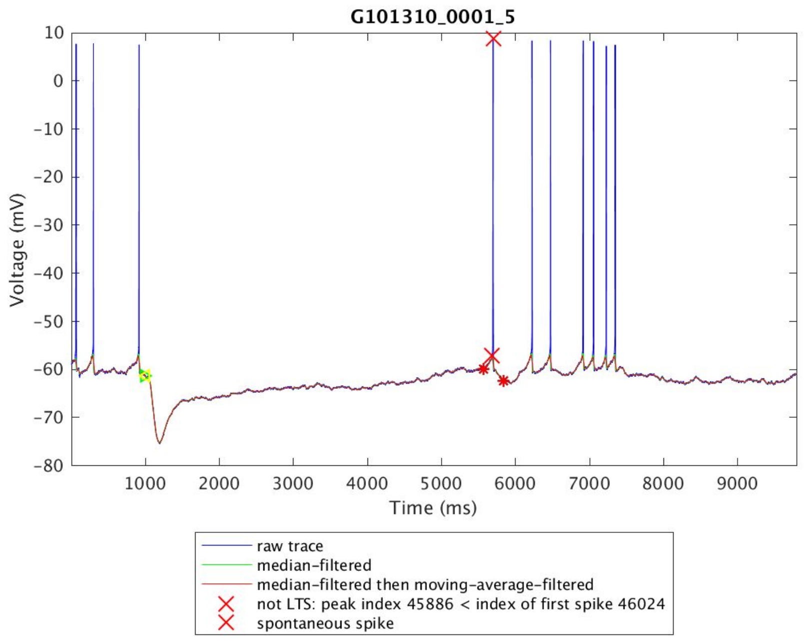


- Reran dclampDataExtractor.m with **E092810_0000** now taken out and with **maxslope** redefined (**dclampDataExtractor12.slurm**, giving the version **old13**)
    - Note, resavedataflag was not on, so **take4/matfiles/** and **dclampdatalog_take4_resave.mat** was modified with **remove_E092810_0000.m** to remove contributions from **E092810_0000** (index)

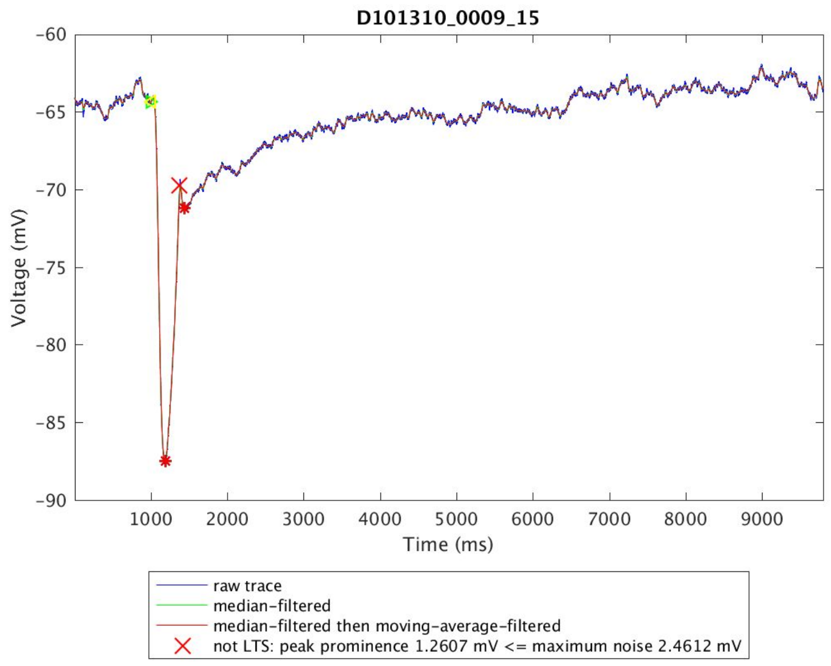




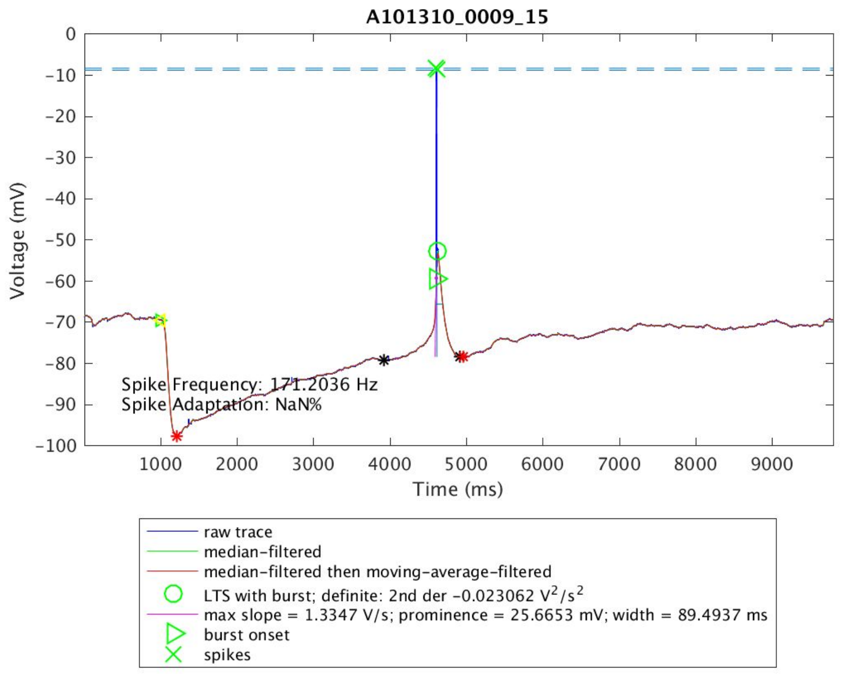
- Modifications to **find_special_cases.m**:
    - 2016-12-05 Moved code to **CreateSubdirAndCopyPNGFiles()**
    - 2016-12-05 Added All_Spontaneous_Spikes, Not_LTS_by_prom && Long_Latency_LTS
- The following sets of special cases are now compiled automatically:
    - **EXAMPLE_All_spontaneous_spikes**: Traces with spikes but not LTSs



- **EXAMPLE_Not_LTS_by_prom**: Traces with peak prominence lower than threshold (peakclass == 1) but with narrowness greater than threshold



- **EXAMPLE_Long-latency_LTS**: Traces with LTS onset time > **3500 ms**



- Modifications to **find_special_cases.m**:
    - 2016-12-06 Moved code to **CreateSubdirAndCopyPNGFiles.m**
- Ran **copy_LTS_figures.sh**, then **backup_figures.sh**, then **update_figures.sh**
- Modifications to **compare_statistics.m**:
    - 2016-12-05 Added peakclass as a comparison stat
    - 2016-12-05 Made the comparison dependent on **filename** instead of index
- Ran **compare_statistics.m**: Compare with **version 12** (old12). There were no traces with altered LTS onset times or altered spikes per peak or altered peak classifications.
- Ran **find_special_cases.m**, reclassified
- Modifications to **find_more_gray_area_traces.sh**:
    - 2016-12-06 Modified to check directories beginning with CONTESTED and OVERRIDE only
    - 2016-12-06 Now checks files in **EXAMPLE_*** as well
    - 2016-12-06 Now checks directories for **noisiness** and **peakclass** separately
- Ran **find_more_gray_area_traces.sh**, reclassified
- Created **check_filecounts.sh** that does the following:
    - This script will check that every sweep is present in **exactly one** directory of the following set of directories (**peakclass**):

      CONTESTED_OVERRIDE_* vs. **OVERRIDE_*** vs. **MAINTAIN_***

      and is present in **exactly one** directory of the following set of directories (**noisiness**):

      CONTESTED_TAKE_OUT_* vs. **TAKE_OUT_*** vs. **KEEP_IN_***
    - If more than one copy exists in a set, the copies will be moved to **UNCLASSIFIED_peakclass** or **UNCLASSIFIED_noisiness**, respectively.
    - If a sweep exists in one set but not the other, it will be copied to the other set's UNCLASSIFIED directory
- Modifications to **update_figures.sh**:
    - 2016-12-06 Force overwrite for _scaled figures under unclassified or CONTESTED* or OVERRIDE*
    - 2016-12-07 Force overwrite for _scaled figures under CONTESTED_* or OVERRIDE_* or MAINTAIN_* or TAKE_OUT_* or KEEP_IN_*
- Modifications to **copy_LTS_figures.sh**:
    - 2016-12-06 Now copies figures only for those in EXAMPLE_*
- Modifications to **backup_figures.sh**:
    - 2016-12-06 Now backs up figures only for those in EXAMPLE_*
- Ran **check_filecounts.sh**, reclassified.
- Created **find_remaining_vtraces_scaled.sh** that places all remaining traces under /**vtraces_scaled_remaining**/. This was not used though since traces were too different are difficult to sort.
- Created **find_remaining_vtraces_scaled.m** that places all remaining traces under /**vtraces_scaled_remaining**/ and then under different subdirectories separated by **peakclass**.

- Went through the entire data using /**vtraces_scaled_remaining**/ and look for possibly **noisy** recordings. Reclassified.
- Ran **find_remaining_vtraces_scaled.m** & **check_filecounts.sh** again to make sure all **7430** voltage traces were classified both in the set **peakclass** and in the set **noisiness**.
- Ran **update_figures.sh**
- Examined each special cases folder and looked for any classification discrepancies
- Brian wrote a Microsoft Visual Basic code (**ScoringWordFileGeneration.docm**) that automatically generates Microsoft Word files from directories with PNG files
    a. For the following folders, each figure is printed on one page, with the following question on the top and **check boxes** for the two answer options:

        Is this an LTS?
        ▢ **LTS**
        ▢ **Not LTS**

Folder name	# of traces
CONTESTED_OVERRIDE_Looks_like_LTS_not_by_narrowness	97
CONTESTED_OVERRIDE_Looks_like_LTS_not_by_prominence	24
CONTESTED_OVERRIDE_Looks_like_missed_LTS	20
CONTESTED_OVERRIDE_Looks_like_Spontaneous_LTSs_or_bursts	42
CONTESTED_OVERRIDE_Wide_LTS_could_be_noise	120

    b. For the following folders, each figure is printed on one page, with the following question on the top and **check boxes** for the two answer options:

        Should we include this trace in the fitting?
        ▢ **Keep in**
        ▢ **Take out**

Folder name	# of traces
CONTESTED_TAKE_OUT_Looks_a_little_noisy	126
CONTESTED_TAKE_OUT_Looks_very_noisy	26
CONTESTED_TAKE_OUT_More_than_one_LTS_no_spont	190
CONTESTED_TAKE_OUT_More_than_one_LTS_with_spontaneous_LTSs	8
CONTESTED_TAKE_OUT_More_than_one_LTS_with_spontaneous_spikes	6
CONTESTED_TAKE_OUT_Spontaneous_bursts	19
CONTESTED_TAKE_OUT_Spontaneous_LTSs	50
CONTESTED_TAKE_OUT_Spontaneous_spikes	168

- Other classified traces are:
  a. In **peakclass**:

Folder name	# of traces
OVERRIDE_Missed_LTS_by_order	1
OVERRIDE_Missed_LTS_by_shape	4
OVERRIDE_Noise_in_trace	44
OVERRIDE_Spikes_per_burst_incorrect	1
OVERRIDE_Spontaneous_LTSs_or_bursts	4
MAINTAIN_True_negative_Not_LTS_by_narrowness	3146
MAINTAIN_True_negative_Not_LTS_by_prominence_in_gray_area	26
MAINTAIN_True_negative_Not_LTS_by_prominence_not_in_gray_area	1232
MAINTAIN_True_negative_Not_LTS_by_shape	119
MAINTAIN_True_positive_LTS_with_burst	1978
MAINTAIN_True_positive_LTS_with_burst_in_gray_area	0
MAINTAIN_True_positive_LTS_with_no_burst	403
MAINTAIN_True_positive_LTS_with_no_burst_in_gray_area	0

  b. In **noisiness**:

Folder name	# of traces
TAKE_OUT_Very_noisy	4
KEEP_IN_Clean_LTS_with_burst	1744
KEEP_IN_Clean_LTS_with_no_burst	386
KEEP_IN_Clean_LTS_with_no_burst_in_gray_area	113
KEEP_IN_Clean_Not_LTS_by_narrowness	3179
KEEP_IN_Clean_Not_LTS_by_overrule	3
KEEP_IN_Clean_Not_LTS_by_prominence	1218
KEEP_IN_Clean_Not_LTS_by_prominence_in_gray_area	21

12/8/2016

<u>Email to John and Mark</u>

Dear John and Mark,

On the voltage trace analysis end, I have finally sorted all 7430 traces and have asked my undergrad to generate a script (which he did in Visual Basic) that compiles the remaining figures I have trouble deciding upon in several Microsoft Word files. There are two things we need to decide for each trace:

(1) **Is this an LTS?**
(2) **Should we include this trace in the fitting?**

Each figure is printed on a page with a question and two options. Each option has an **interactive check box** beside it, so you can score it by hand and we will find a way to calculate the scores later.

Here's a breakdown of the number of traces in each file:
[See above]

The traces in **CONTESTED_OVERRIDE_Looks_like_LTS_not_by_narrowness**, **CONTESTED_OVERRIDE_Looks_like_LTS_not_by_prominence** & **CONTESTED_OVERRIDE_Looks_like_missed_LTS** are possible **false negatives**; whereas the traces in **CONTESTED_OVERRIDE_Looks_like_Spontaneous_LTSs_or_bursts** & **CONTESTED_OVERRIDE_Wide_LTS_could_be_noise** are possible **false positives**.

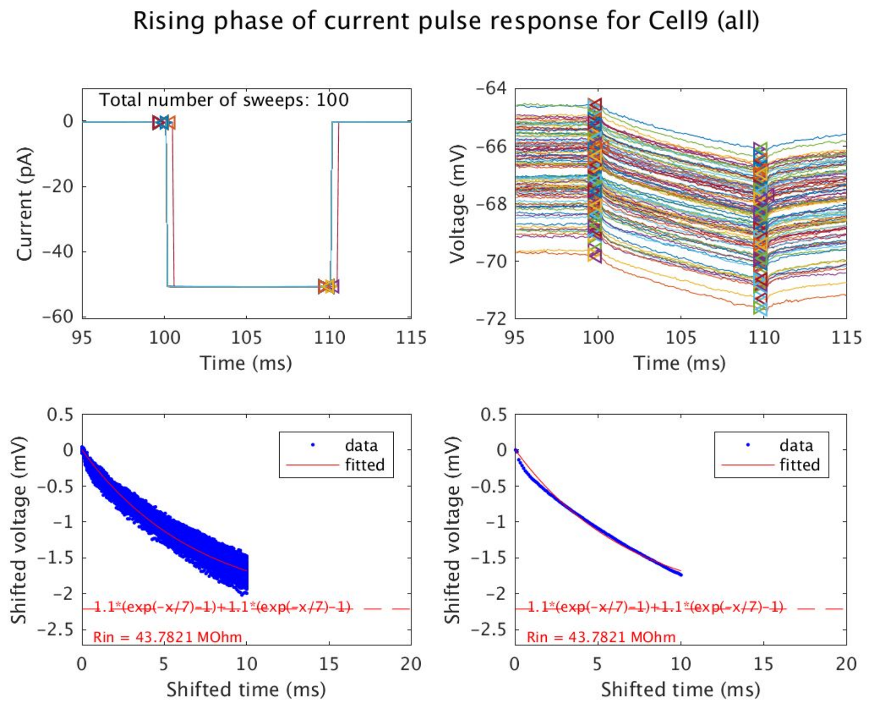
The traces in **CONTESTED_TAKE_OUT_Looks_a_little_noisy**, **CONTESTED_TAKE_OUT_Looks_very_noisy**, **CONTESTED_TAKE_OUT_Spontaneous_bursts**, **CONTESTED_TAKE_OUT_Spontaneous_LTSs**, **CONTESTED_TAKE_OUT_Spontaneous_spikes**, **CONTESTED_TAKE_OUT_More_than_one_LTS_with_spontaneous_LTSs** & **CONTESTED_TAKE_OUT_More_than_one_LTS_with_spontaneous_spikes** all are noisier than usual and some of them might disrupt the fitting process. However, if we take out traces (if one trace is taken out, I will take out all 5 repetitions so that the statistics would make more sense), we are also potentially losing good LTS data that could help with the fitting.

The traces in **CONTESTED_TAKE_OUT_More_than_one_LTS_no_spont** are actually not noisy but just has more than one LTSs. I was wondering whether the simulations can actually produce such LTS series with accuracy. On the other hand, maybe we could use this LTS series behavior to gain insight into certain parameters. So it's probably not necessary to score this one as I can attempt to fit these LTS series with the simulations.

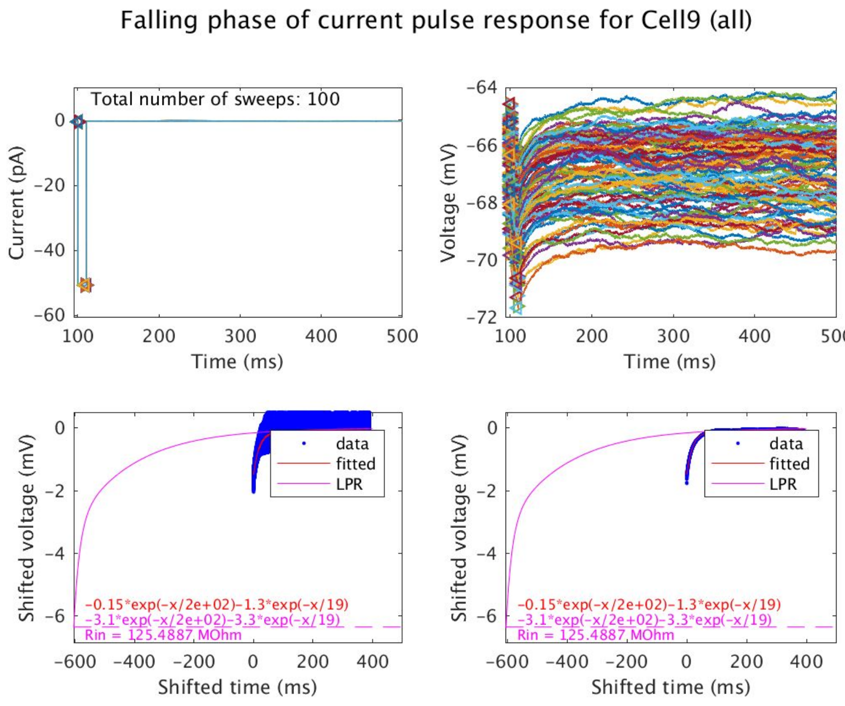
12/1/2016~

<u>Passive fitting (cont'd)</u>

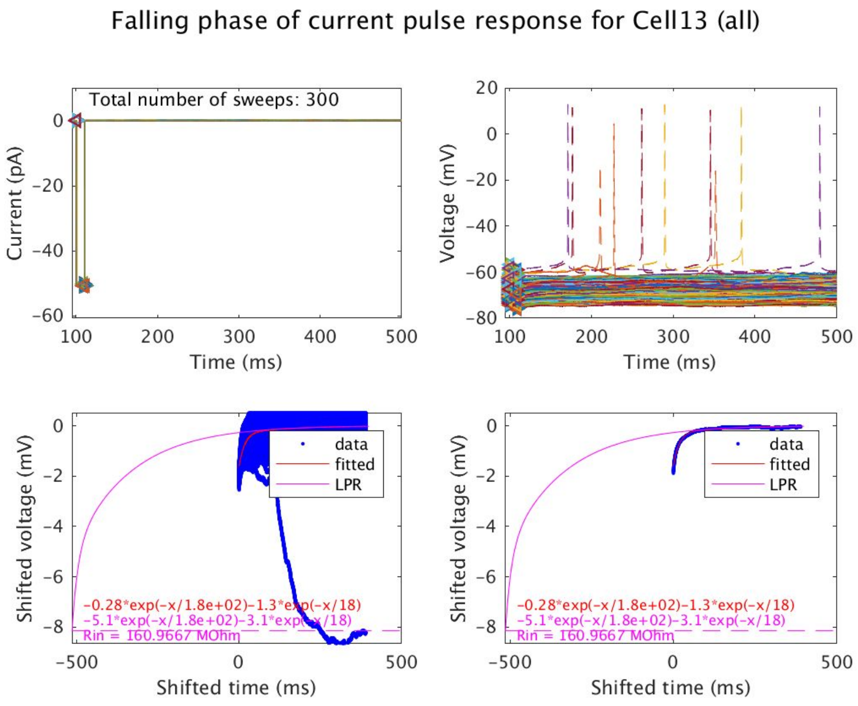
- Modified **dclampPassiveFitter.m**:
    - 2016-12-01 - Added groupmode, made default to be grouping by cell only
- Modified **PlotHistogramsRefineThreshold.m**:
    - 2016-12-01 - Added groupmode, made default to be grouping by cell only
    - 2016-12-01 - Changed number of bins of passive parameter histograms to **10**
- Modified **plot_and_save_histogram.m**:
    - 2016-12-01 If classes doesn't exist, plot **regular histogram** instead of stacked histogram
    - 2016-12-01 Added **countlabel**
- In version **old13** of dclampDataExtractor.m, all traces from the same **cell** were pooled together to give one set of parameters for each cell.
    - Remaining problem #1: need to account for **series resistance** in the parameter estimation
    - Remaining problem #2: some curves are fitted with a **single exponential** only



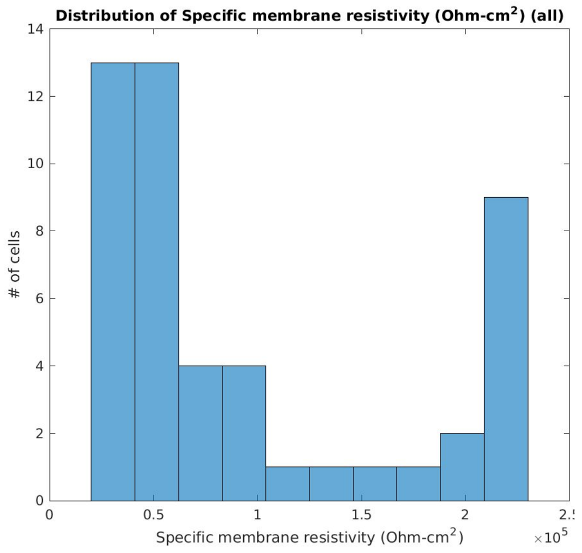
- Remaining problem #3: sometimes τ_0 **saturates** to maximum allowed value (**200 ms**)

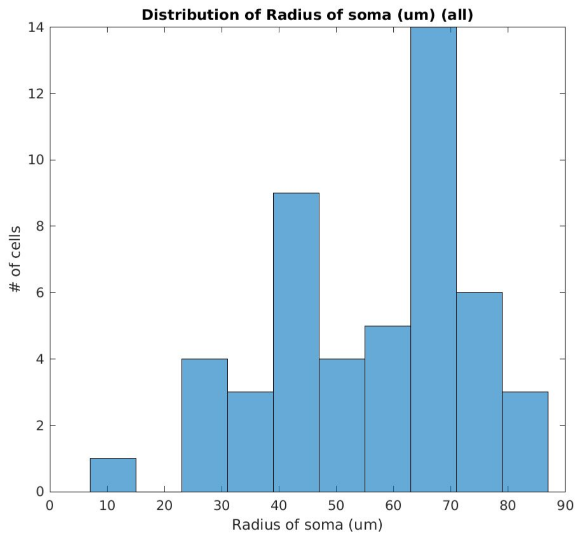


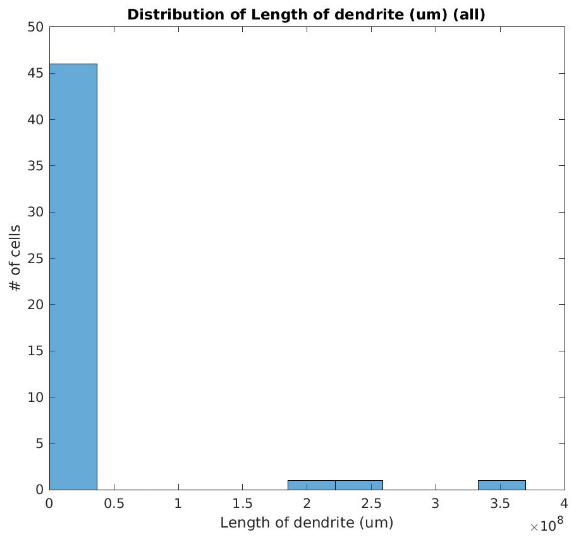
- Remaining problem #4: some traces are very **noisy**

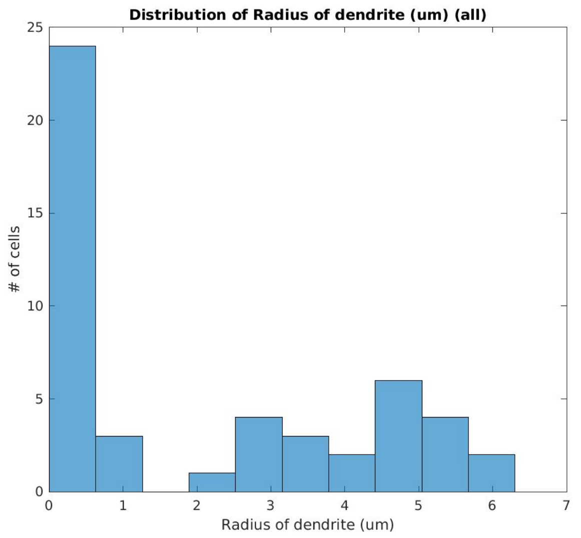


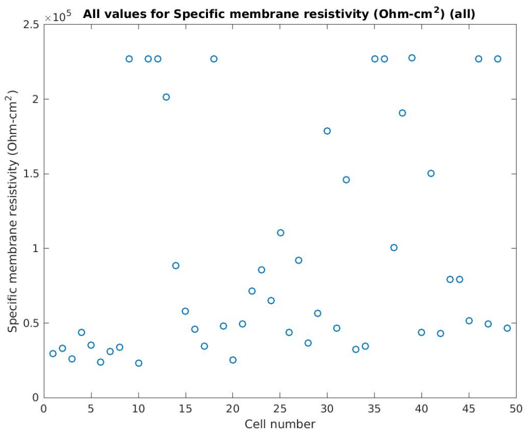
- Distribution of parameters

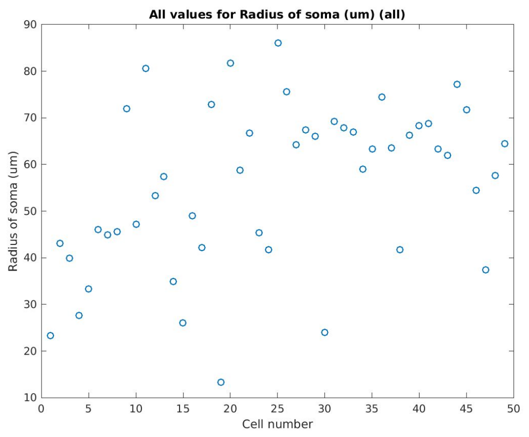


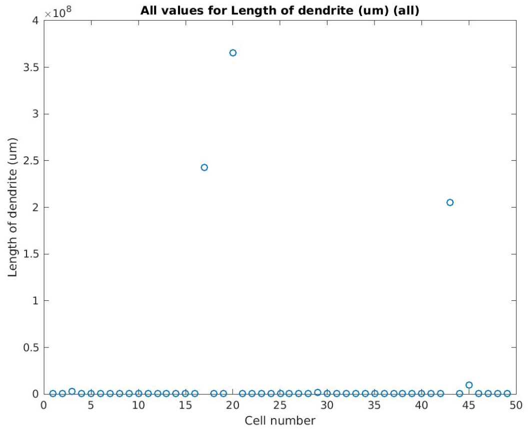


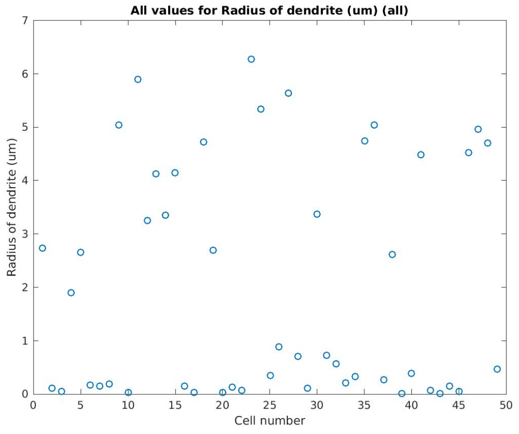












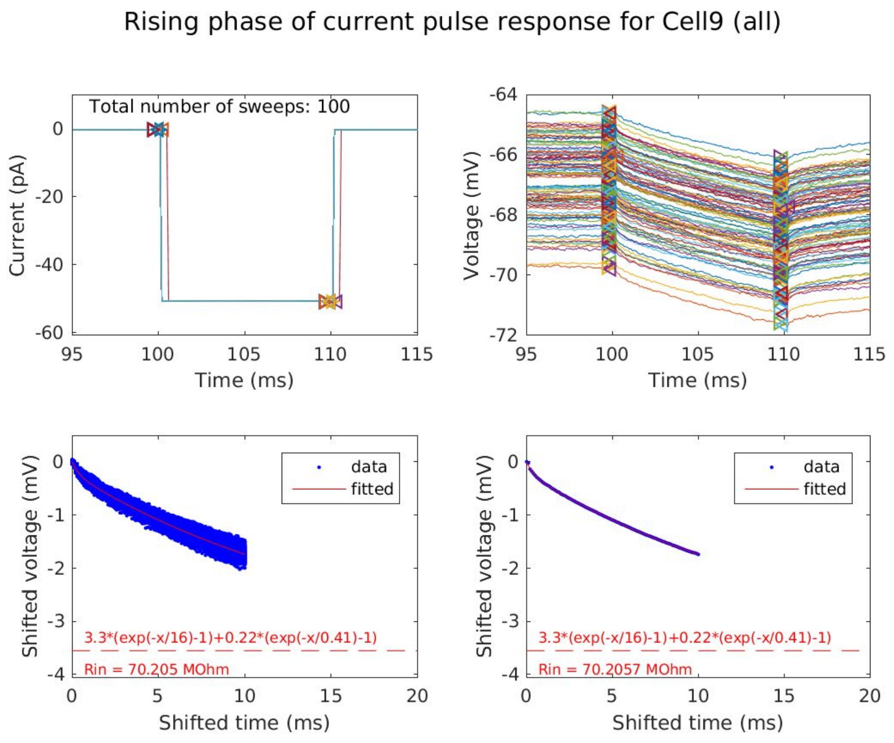
- Account for series resistance: Given **total input resistance of the setup** R_{in} and **dendritic-to-somatic conductance ratio** ρ, we compute the somatic and dendritic resistances as follows:
  - First, assume the **series resistance** R_s to be fixed at **10 MΩ**, then the **input resistance of the cell** (total membrane resistance)
  $$R_{in} = R_s + R_N$$
  $$\Rightarrow R_N = R_{in} - R_s$$
  - The **input conductance (**G_N**, [μS])** is then found by:
  $$G_N = \frac{1}{R_N}$$
  - By definition (Johnston & Wu, p. 90), we have
  $$\rho = \frac{G_D}{G_S}$$
  So the **somatic** and **dendritic conductances (**G_S, G_D**, [μS])** are then found by:
  $$G_N = G_S + G_D = G_S(1+\rho)$$
  $$\Rightarrow G_S = \frac{G_N}{1+\rho}, G_D = \rho G_S$$
  - The **somatic** and **dendritic resistances (**R_S, R_D**, [MΩ])** are then found by:
  $$R_S = \frac{1}{G_S}, R_D = \frac{1}{G_D}$$

- Modified **find_passive_params.m**:
  - 2016-12-04 Added series resistance **Rs** and changed the way somatic and dendritic resistances are computed
  - 2016-12-04 Increase the upper bound of tau to **500 ms**
  - Added **rmse_R** && **rmse_F**, the **root-mean-squared errors** of the rising and falling phase, respectively, for each sweep
  - 2016-12-04 Added the functions **fit_setup** && **measure_error**

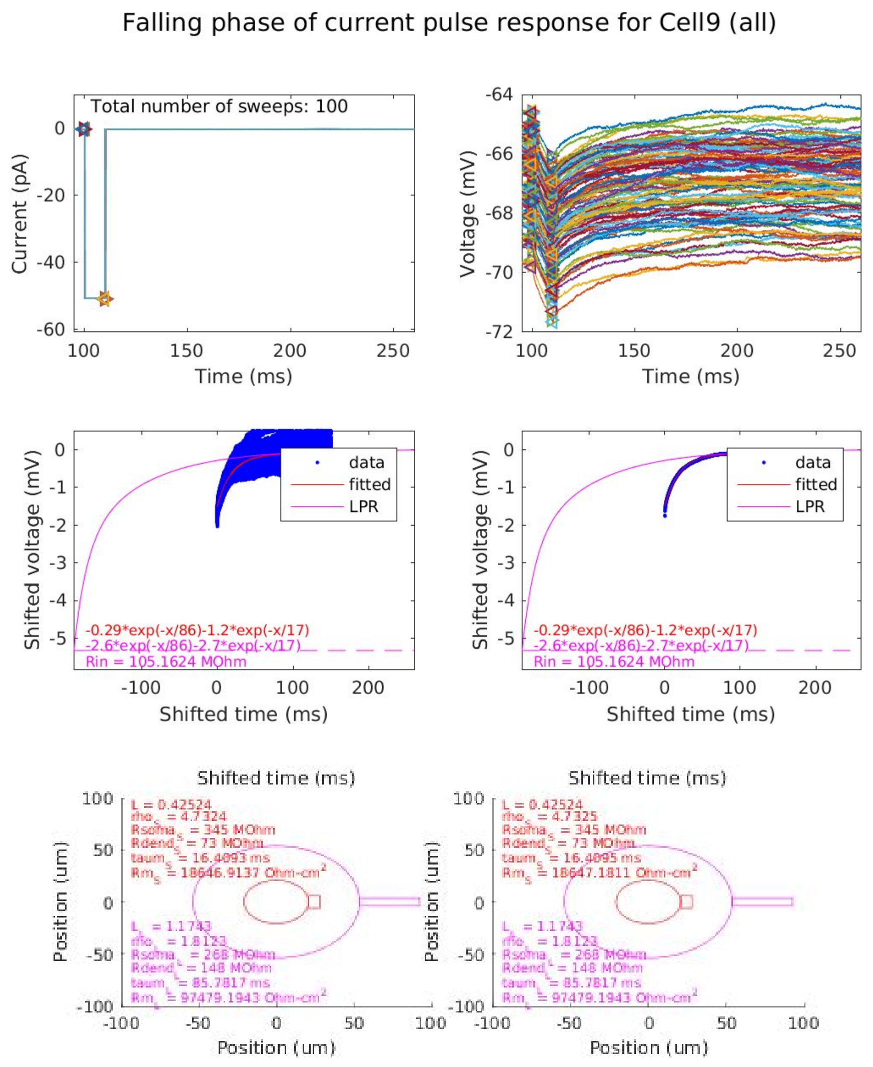
- Modified **dclampPassiveFitter.m**:
    - 2016-12-04 Changed current pulse response to last just **150 ms** (cprwin is changed from [95, 500] to **[95, 260]**)
    - 2016-12-04 Added **logheader_swpinfo** && **logvariables_swpinfo**
    - 2016-12-04 Added **Rmemb**, **Gmemb**, **Gsoma**, **Gdend** to be saved in params
    - 2016-12-04 Added **rmse_R** && **rmse_F**, the root-mean-squared errors of the rising and falling phase, respectively, for each sweep
- Modified **PlotHistogramsRefineThreshold.m**:
    - 2016-12-04 - Changed logvariables to **logvariables_params** for mpassive
    - 2016-12-04 - Added **dclampPassiveLog_byswps** but plot only logvariables_swpinfo
- Reran dclampPassiveFitter(0)
    - **dclampPassiveFitter2.slurm** ran into error and terminated**:**

        Error using dclampPassiveFitter (line 406)

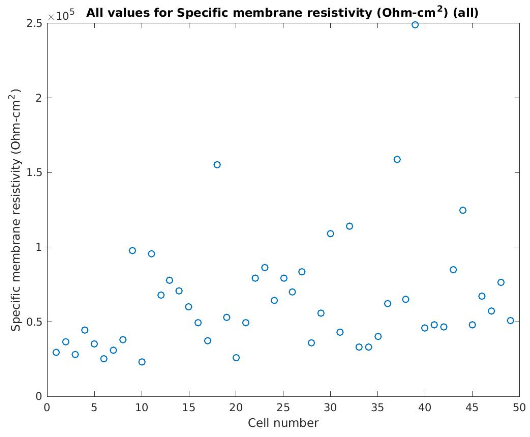
        “X must be a matrix with one or two columns”
- Modified **find_passive_params.m**:
    - 2016-12-04 Fixed measure_error so that sweeps yielding nonsensical responses have rmse = **Inf**
- Reran dclampPassiveFitter(0) on **fishfish** (version **old13-1**) instead with no error.
    - Somehow the fitting for the rising phase for **Cell #9** was altered to give two exponentials this time. This might be an effect of the **increase in tau_max**. It could also be an effect of **fishfish vs. Rivanna**.

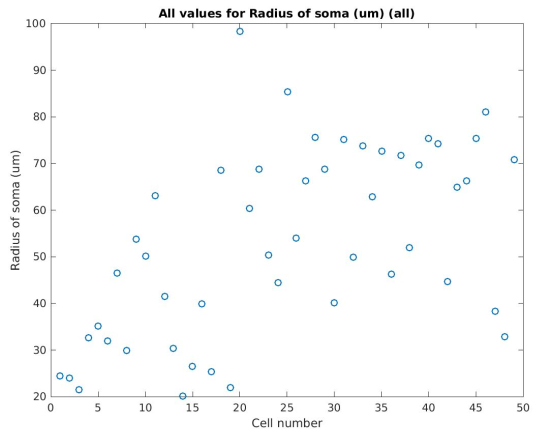


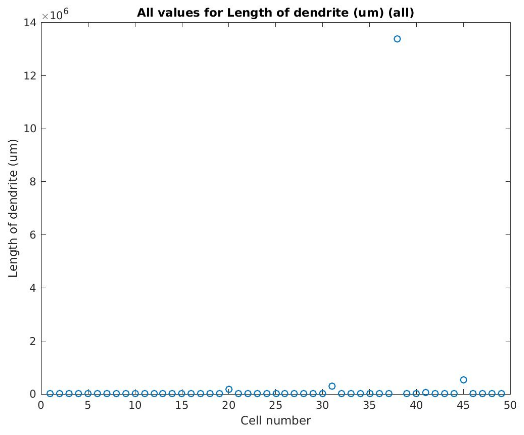
- The falling phase for **Cell #9** didn't saturate like before, presumably because the range of fitting was decreased to **150 ms**.

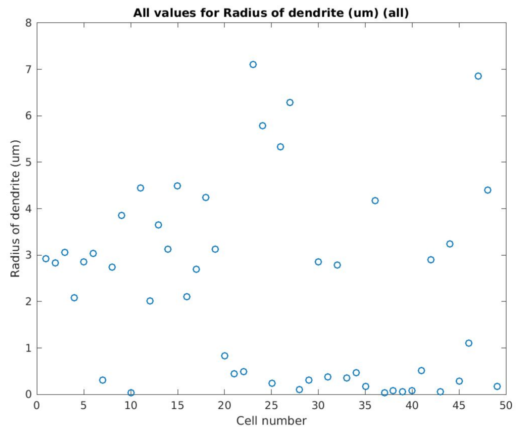


.

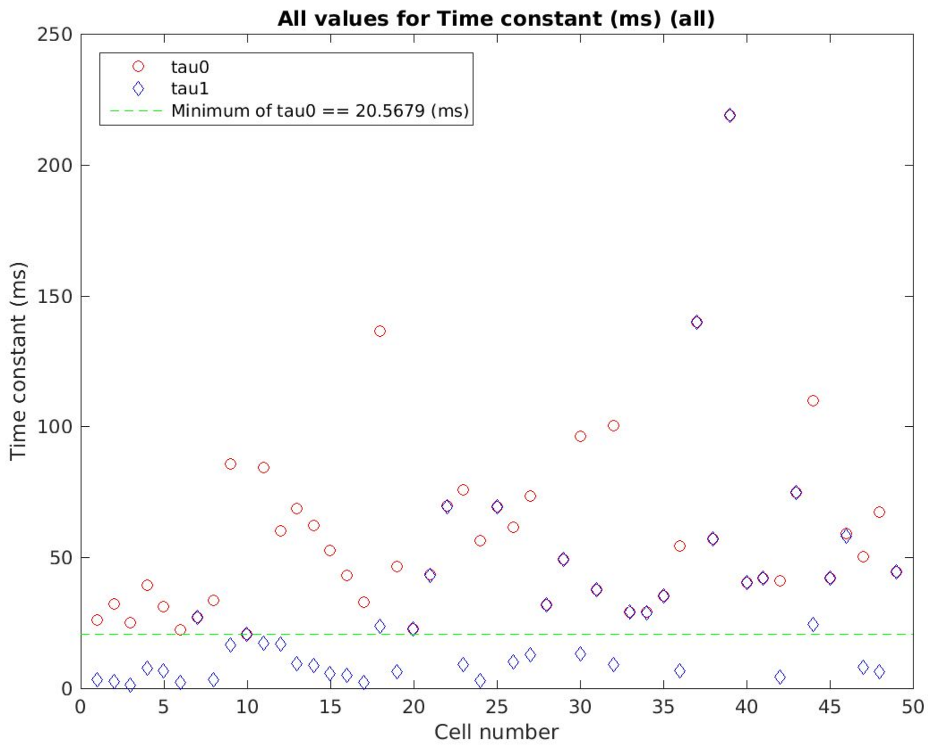




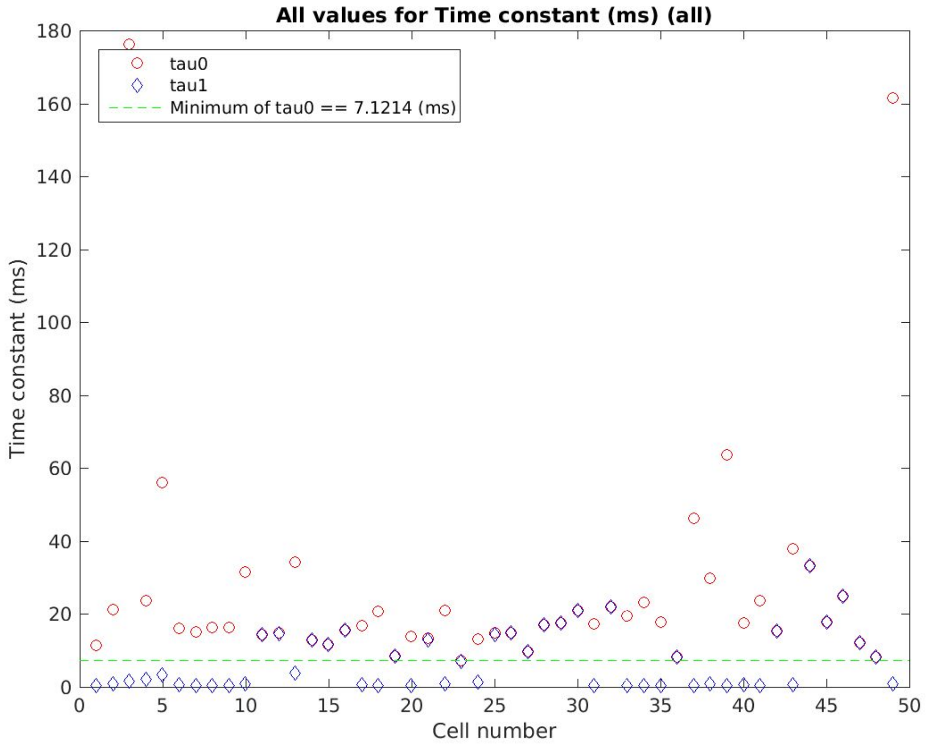




- Modified **PlotHistogramsRefineThreshold.m**:
    - 2016-12-05 - Added the bar graphs **tau0_tau1_c** and **tau0_tau1_rising_c**
- Plotted current double exponential fits to determine the **ranges** needed for constraining τ_0 and τ_1
    - From the falling phase:

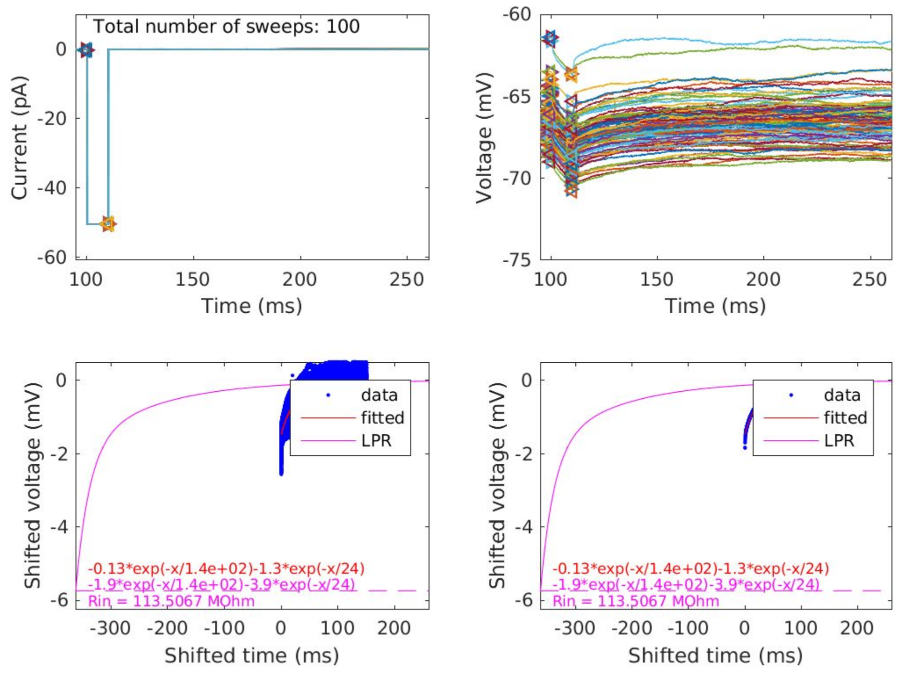


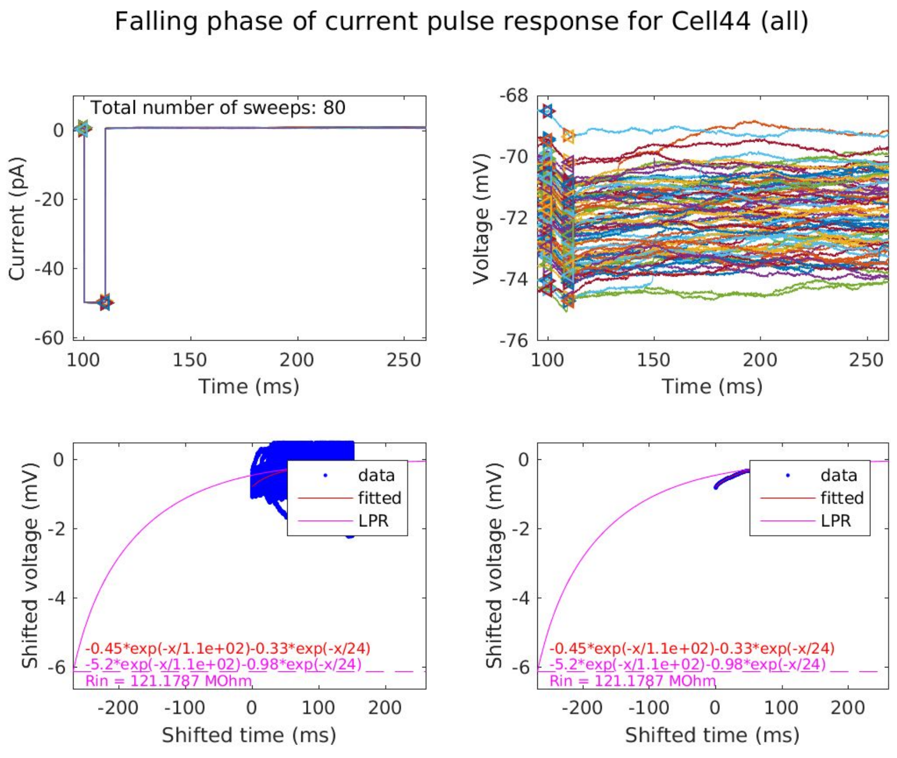
- From the rising phase:



- From the falling phase, the two cells with τ_1 greater than the minimum of τ_0:

Falling phase of current pulse response for Cell18 (all)

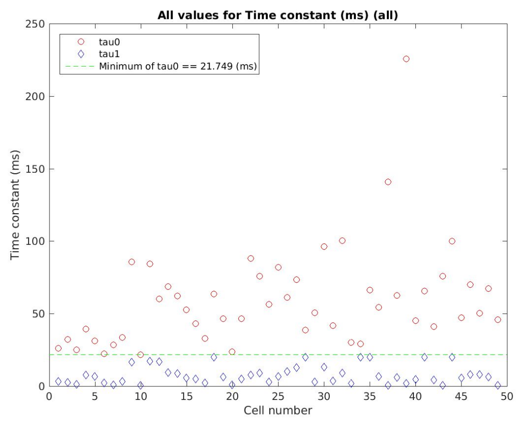




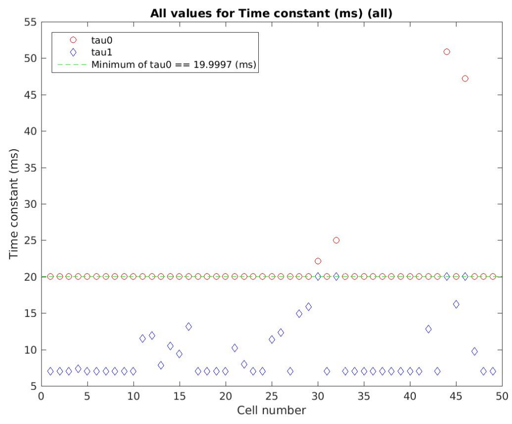
- Modified **find_passive_params.m**:
    - 2016-12-05 Added **tau0_range** = **[20, 500]** and **tau1_range** = **[0, 20]**
    - 2016-12-05 Removed typtau and tau_max, changed initial conditions to **tau0_range(1)** and **tau1_range(2)**

- Fixed range of τ_0 to **20~500 ms** and range of τ_1 to **0~20 ms**. Initial conditions both at **20 ms**. Reran dclampPassiveFitter(0) on fishfish (version **old13-2**).
    - Problem: For many traces, the rising phase can't fit well with such boundary conditions:

Falling phase

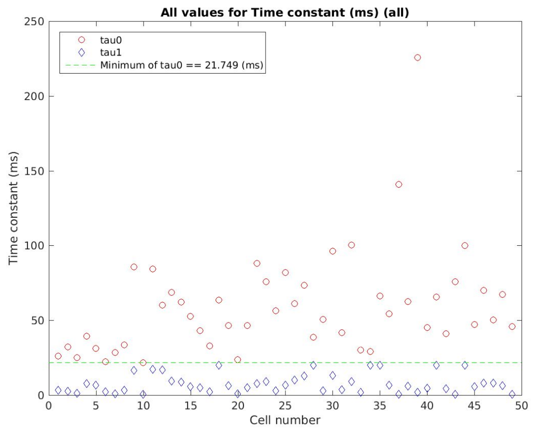


Rising phase

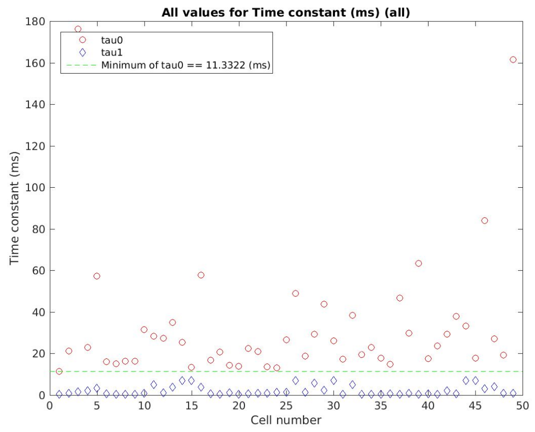


- Modified **find_passive_params.m**:
  - 2016-12-05 Added **tau0_range_R** and **tau1_range_R**
- For the **rising phase only**, changed fixed range of τ_0 to **7~200 ms** and range of τ_1 to **0~7 ms**. Initial conditions both at **7 ms**. Reran dclampPassiveFitter(0) on fishfish (version **old13-3**)
  - The rising phase fitted much better:

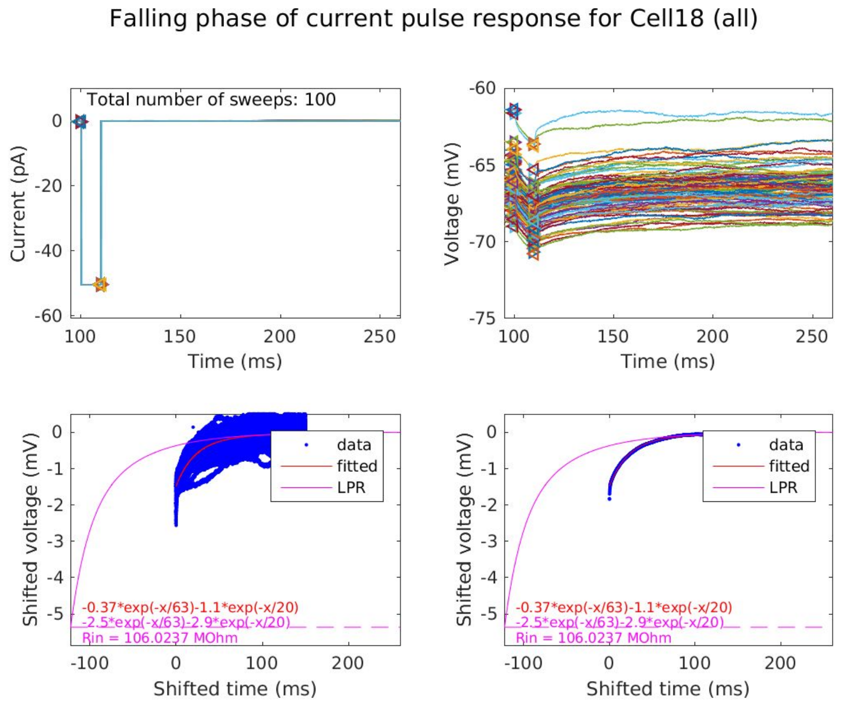
Falling phase

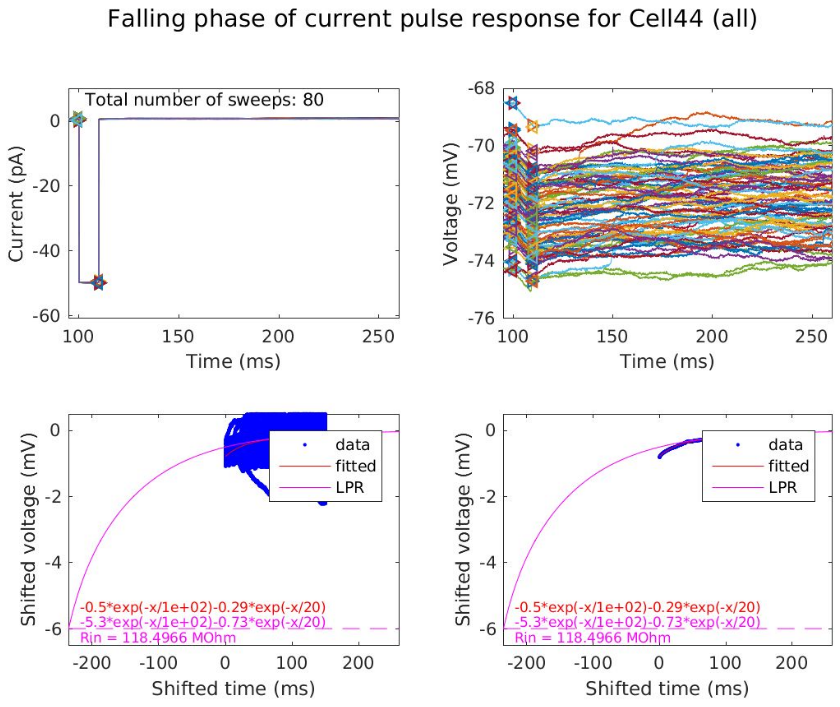


Rising phase

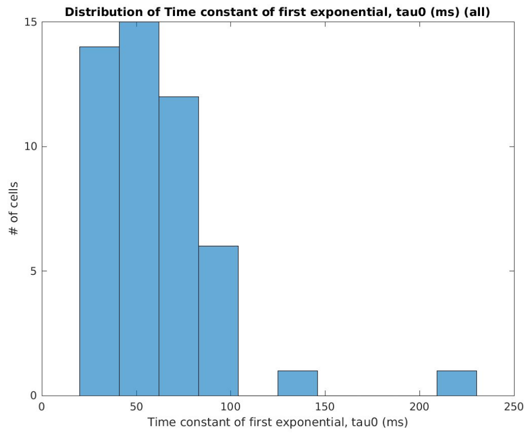


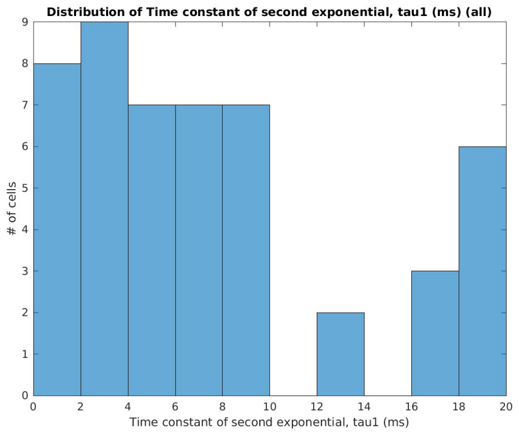
  - However, in many case τ_1 did not move much from the initial condition 20 ms, which is at an upper boundary. For instance, see the following cells:





- The distribution of τ_0 and τ_1 in the falling phase: (The median of τ_0 is **52.5527 ms** and the median of τ_1 is **6.1804 ms**. For the rising phase it is 23.0042 ms and 0.8481 ms, respectively.)



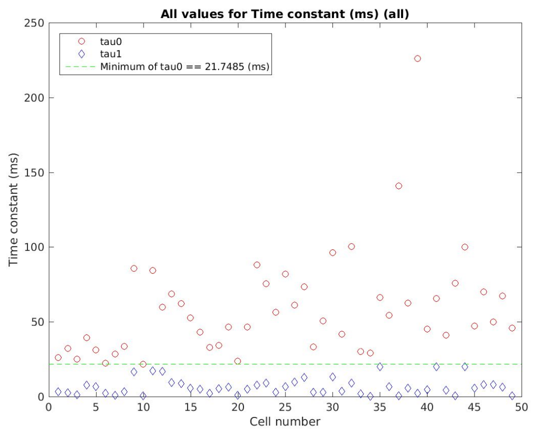


- Modified **find_passive_params.m**:
  - 2016-12-05 Added **typtau0**, **typtau1**, **typtau0_R**, **typtau1_R**
- Modified **PlotHistogramsRefineThreshold.m**:
  - 2016-12-05 - Moved part of the code to **structs2vecs.m**

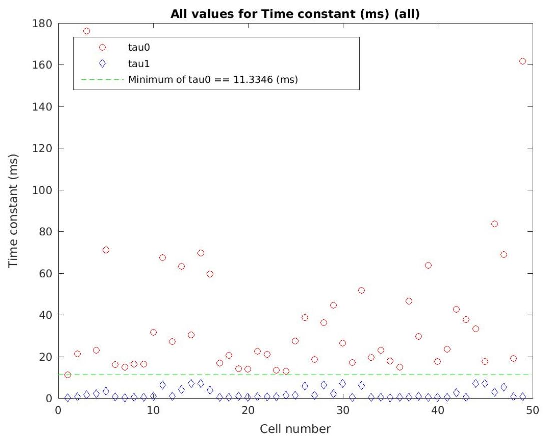
- For the **falling phase**, changed initial condition of τ_0 to **50 ms** and initial condition of τ_1 to **6 ms**. For the **rising phase**, changed initial condition of τ_0 to **23 ms** and initial condition of τ_1 to **0.8 ms**. Reran dclampPassiveFitter(0) on fishfish (version **old13-4**)

- The result was mostly the same as before, showing that the **initial conditions** are not as important as the **boundary conditions**:

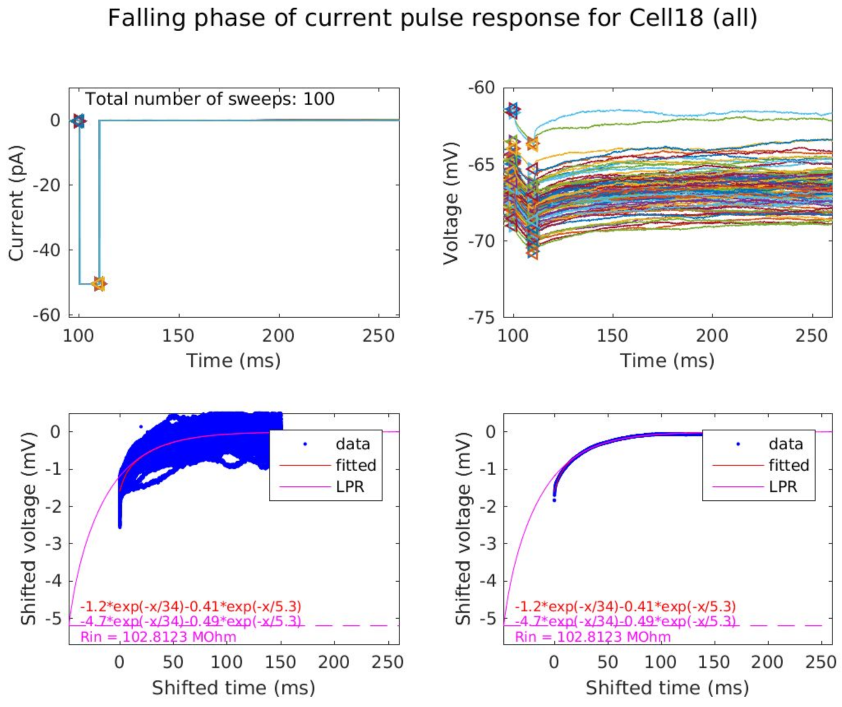
Falling phase



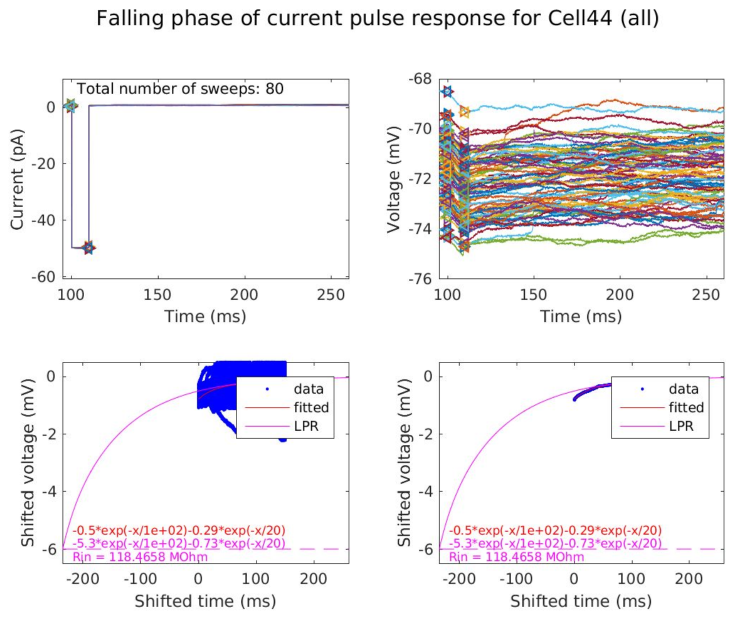
Rising phase



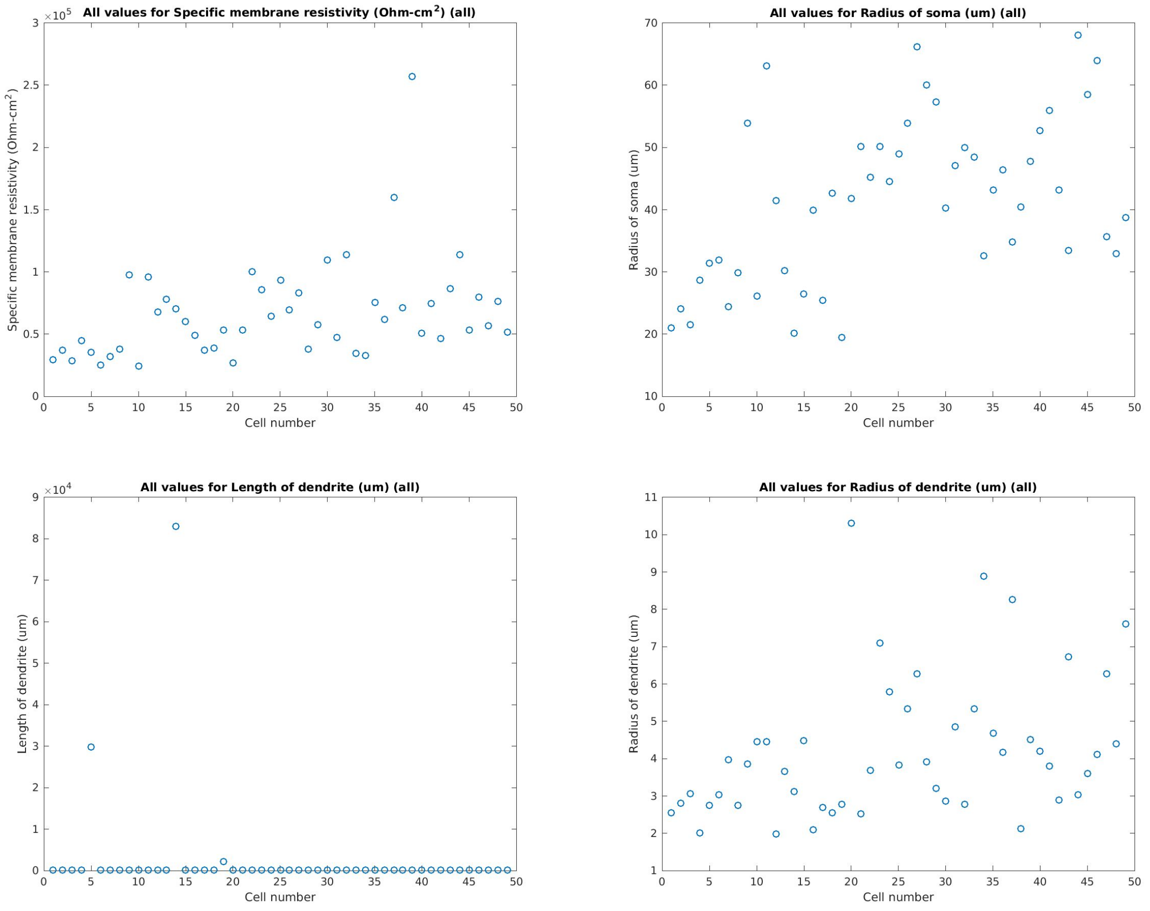
- Changing the initial conditions moved the estimates away from the boundaries in some cases:



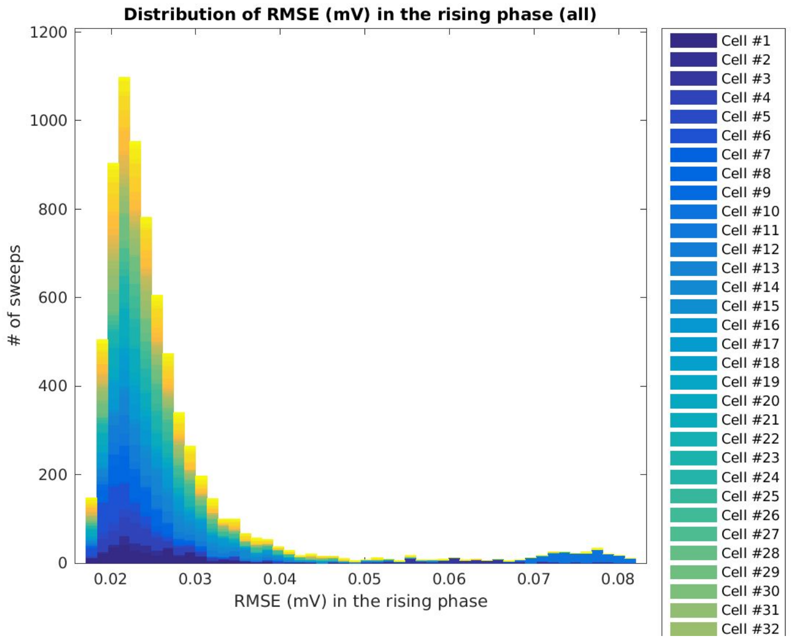
- But not others

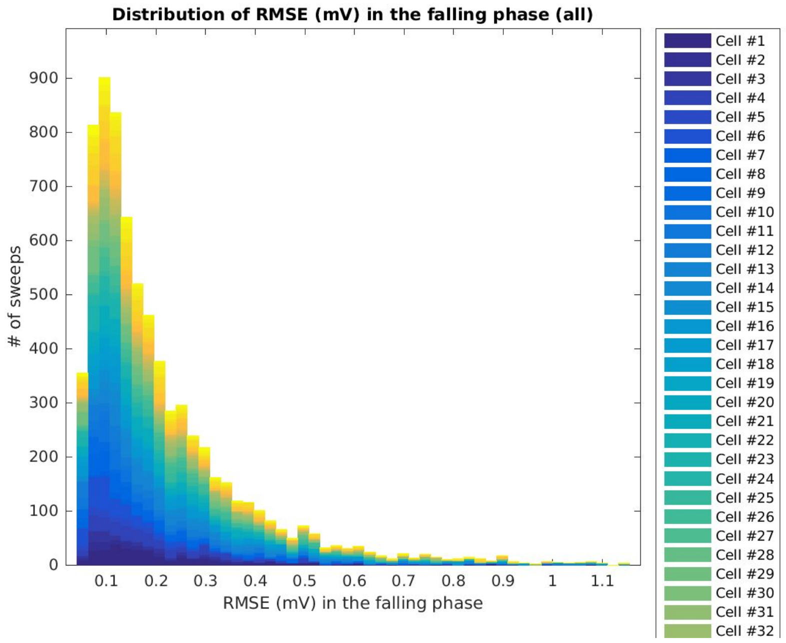


- The dendritic radii makes more sense than before; the outliers of the dendritic length are two magnitudes smaller too



- A **root-mean-squared error** was computed relative to the fit for each individual trace. A histogram of errors was plotted.





  - How can a **threshold** be determined from the **falling-phase histogram**?
  - Are the traces with high RMSE in the rising phase the same traces with high RMSE in the falling phase? Can we use a threshold from the **rising-phase** histogram to infer a threshold in the **falling-phase** histogram?

12/4/2016

Email to John

Dear John,

I realized that I had some outstanding questions about the abf files and the passive fitting process.

- Abf files:
  - How does PClamp record **conductance** values?
  - Are the recorded **current** values recorded by PClamp or set by dynamic clamp?
  - Are the recorded **voltage** values exactly the same as those read by dynamic clamp?
- Passive fitting
  - Does **Ra** affect input resistance? If you look at Figure 4 of Mainen et al 1996, input resistance increases as **Ra** increases, but why? I thought **R_input = Rm + Rs**, so where is Ra in the equation?
  - Where in the files might we be able to find the values of the **resting membrane potential**?
  - How do we deduce initial values for the following parameters from the estimated parameters we got from double exponential fits (the membrane resistivity **Rm [Ohm-cm^2]**, the somatic radius **rad_soma**, the dendritic length **length_dend**, the dendritic radius **rad_dend**)? Note that we are no longer varying the axial resistivity **Ra** and the specific membrane capacitance **cm**.
    - The leak conductance **g_pas [S/cm^2]**: Is this just the inverse of **Rm**?
    - The leak reversal potential **e_pas [mV]**: I don't know how to get this without information about the resting membrane potential.
    - The dendritic surface area correction factor **corrD**: I'm assuming this is a correction factor that's needed to account for the change in surface area versus volume ratio when you do a compartment reduction. Maybe we should make the dendritic length or radius constant and deduce such a correction factor? Or maybe we should just estimate the dendritic length and radius and get rid of this parameter since we are not performing any reduction from a detailed neuron in the first place.
  - Which channels to include when fitting the current pulse response? Figure 4 of Amarillo et al 2014 shows that the resting membrane potential is in fact a contribution from 7 different channels (persistent sodium current **INaP**, the hyperpolarization-activated cationic current **Ih**, the low-threshold activated calcium current **IT**, the low-threshold transient potassium current **IA**, the potassium leak current **IKleak**, and the sodium leak current **INaleak** and the inwardly rectifying potassium current **IKir**), so it might not make sense to leave them out of the fitting. Of course the leak conductances still play the most

significant role. Therefore, maybe an iterative approach would be ideal. I know you said that for simplicity, we can just assume that all contributions are lumped into a single leak channel. However, Destexhe et al 1996a says that they only fitted to voltage-clamp recordings and not **current-clamp** recordings "because the model included only a subset of currents present." Why is this? Shouldn't we include all currents (all types of channels) in our current pulse response fits then?

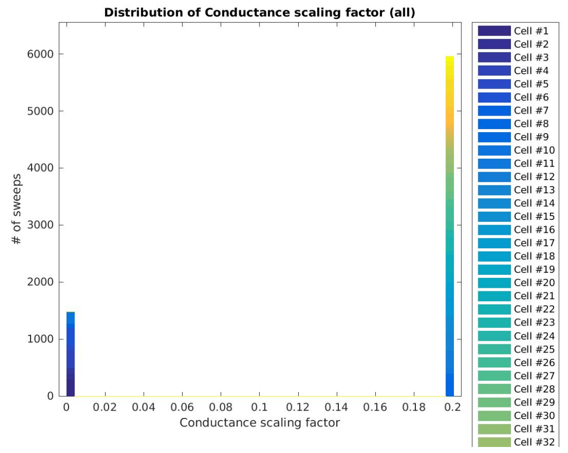
I'm still working on improving the double exponential fit, but Mark thought you could discuss while at AES so I've listed some of the questions I might encounter soon.

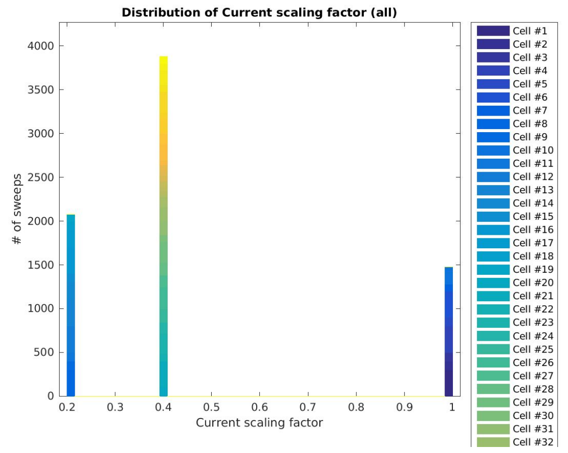
Thanks,
Adam

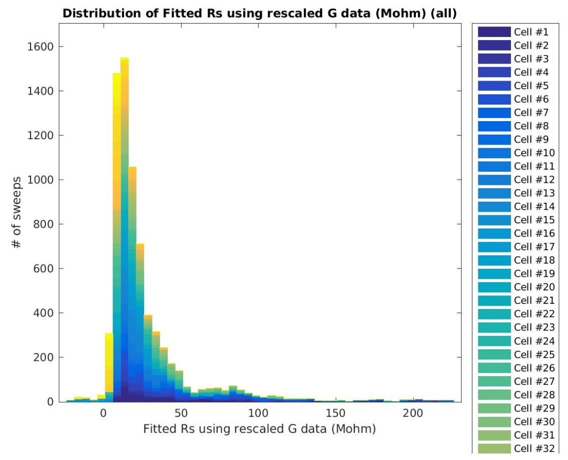
12/8/2016~12/9/2016

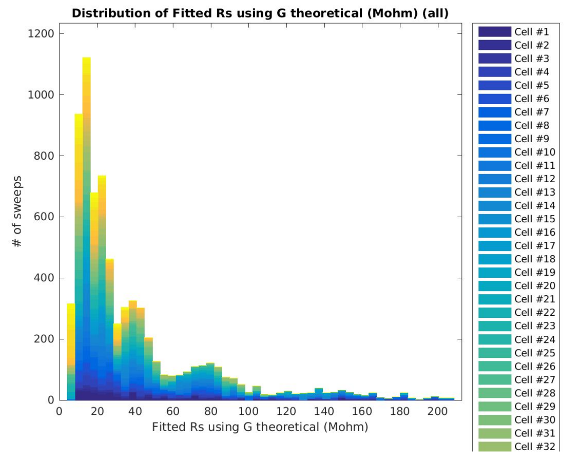
Trace comparison (cont'd)

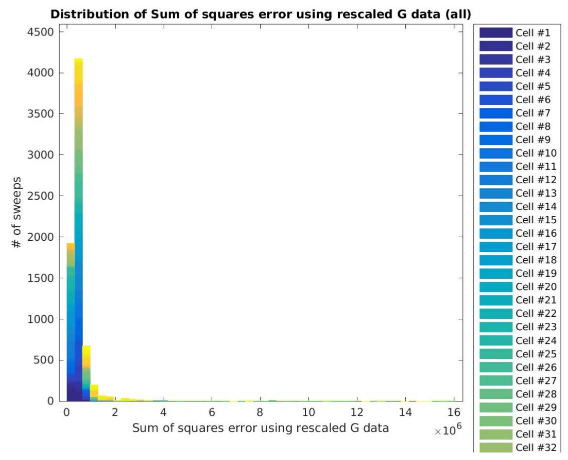
- Modifications to **PlotHistogramsRefineThreshold.m**:
  - 2016-12-08 - Changed the grouping for variables in **trace_comparison.mat** & **dclampPassiveLog_byswps_all.mat** to be by **cell ID#**
- These were the trace correction results from before, replotted:

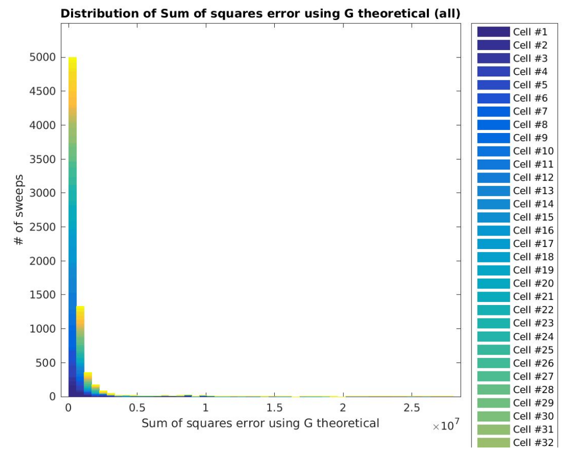




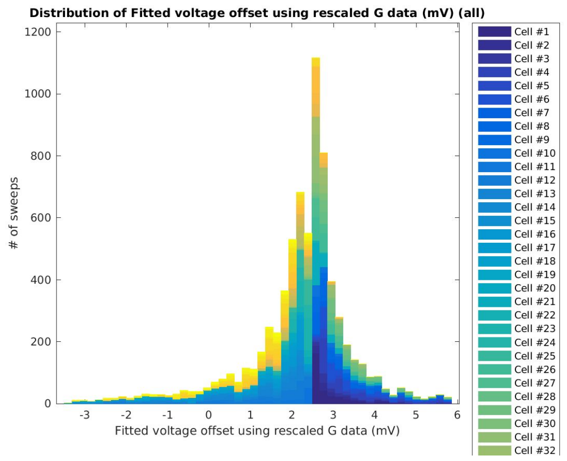


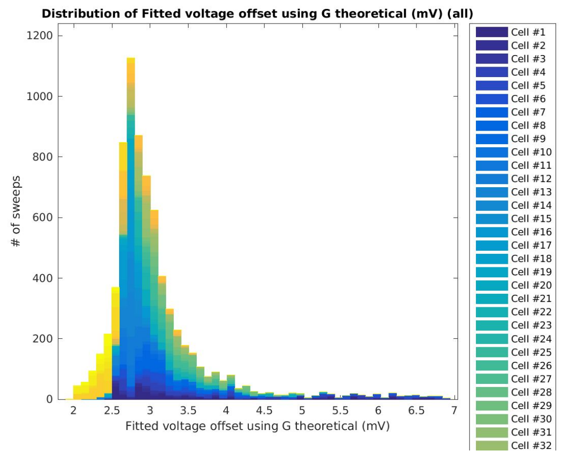


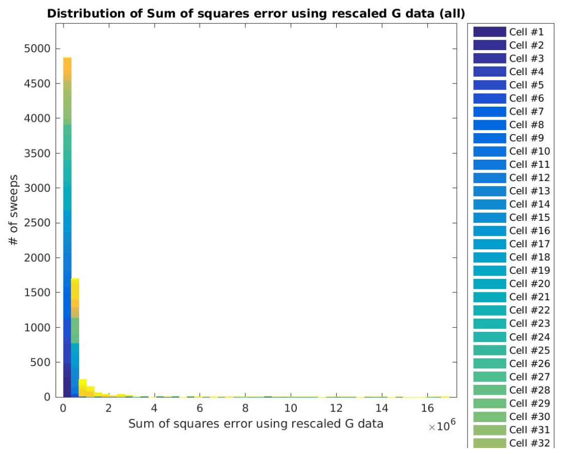


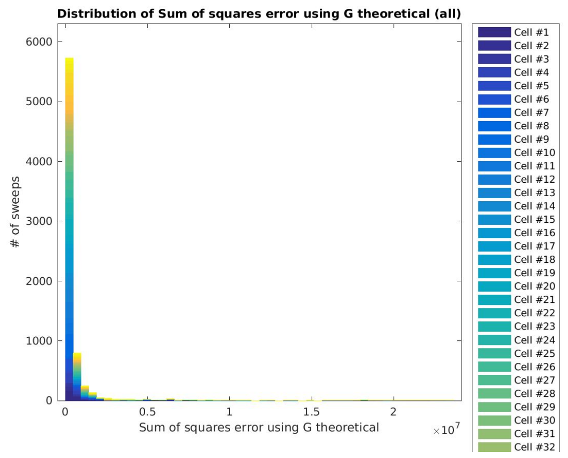


- Modifications to **trace_comparison.m**:
    - 2016-12-08 - Changed the series resistance compensation to a **fixed voltage offset**
- Ran **trace_comparison.m** (**trace_comparison4.slurm**) with the series resistance correction changed to a **fixed voltage offset per trace**



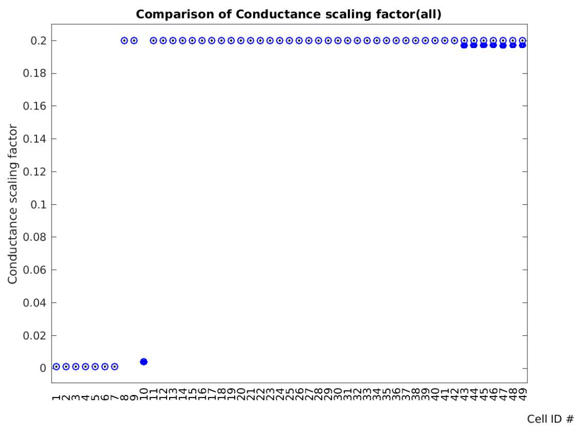


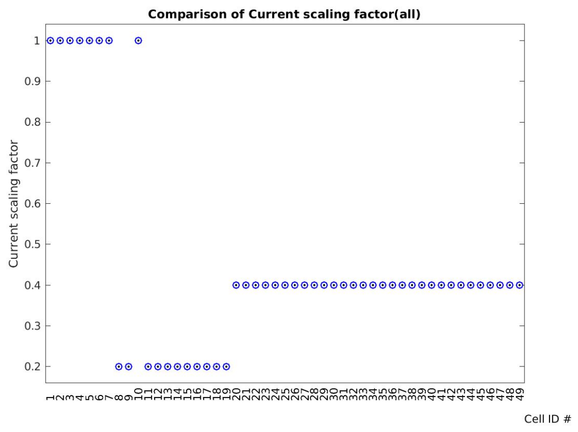




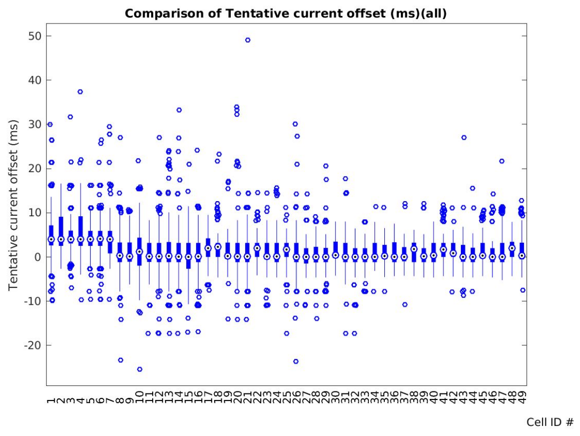
- Created **plot_and_save_boxplot.m** to plot box plots
- Modifications to **PlotHistogramsRefineThreshold.m**:
    - % 2016-12-08 - Now plots **boxplots** as well for variables in trace_comparison.mat & dclampPassiveLog_byswps_all.mat

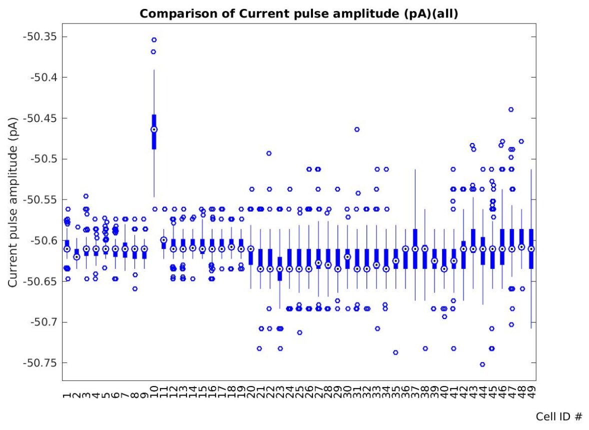
- Better visualization of the **scaling factors** used to correct **conductance** and **current** traces:



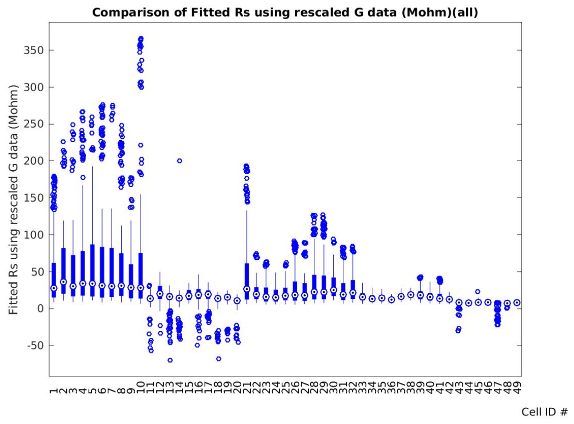


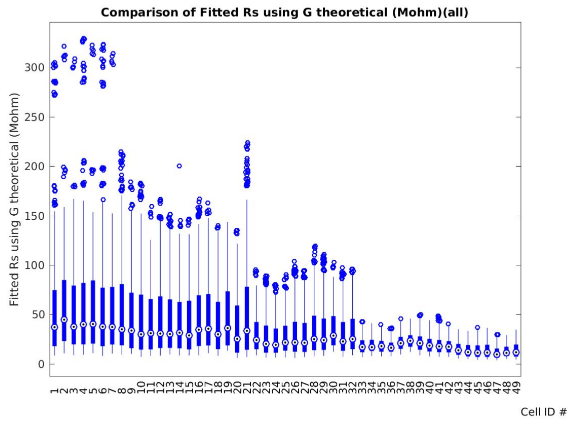
- Better visualization of the **tentative IPSC offsets** and the **current pulse amplitude**:

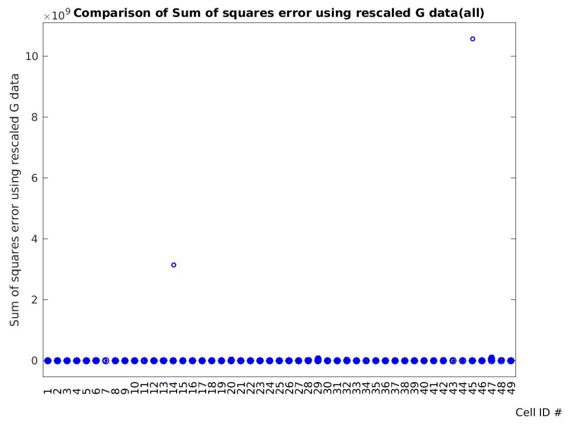


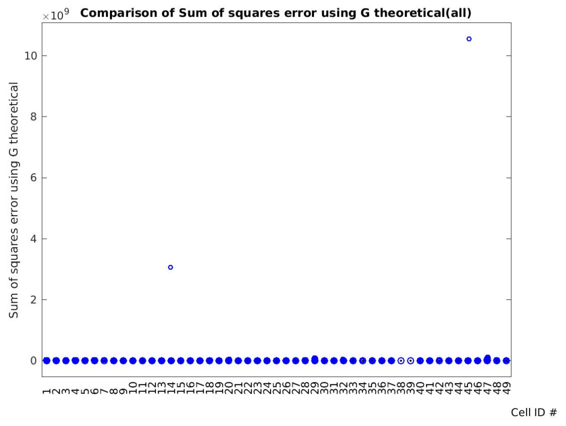


- Better visualization of the **series resistance** fits **across cells** and their respective **sum-of-squares errors**:

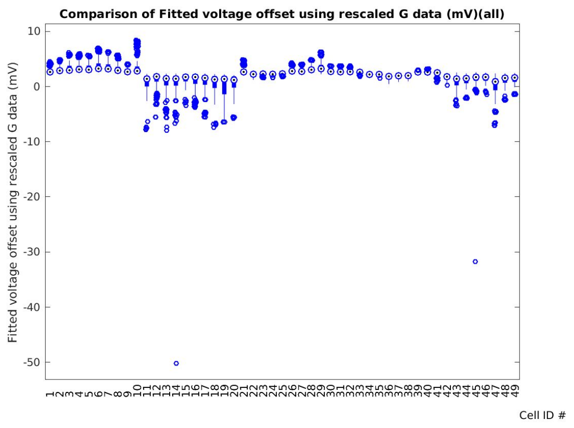


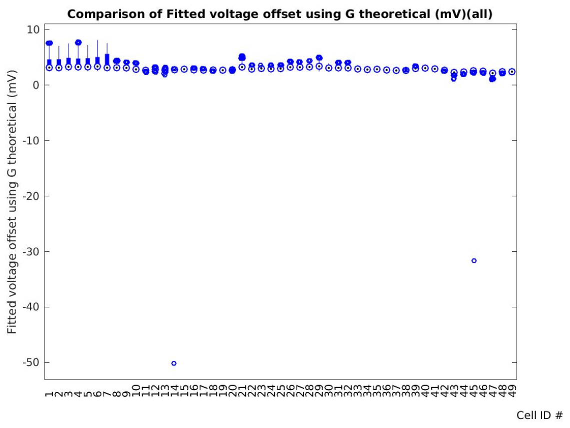


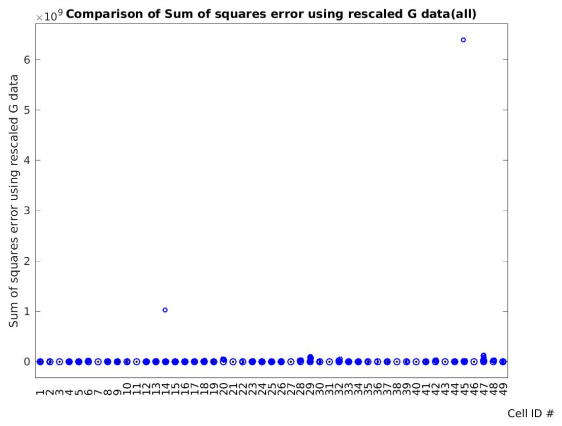


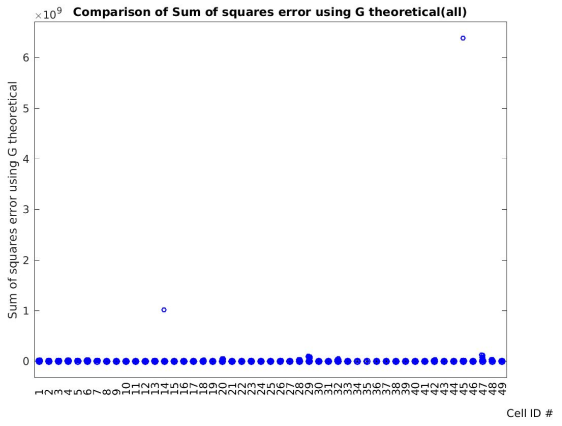


- Better visualization of the **voltage offset** fits **across cells** and their respective **sum-of-squares errors**:



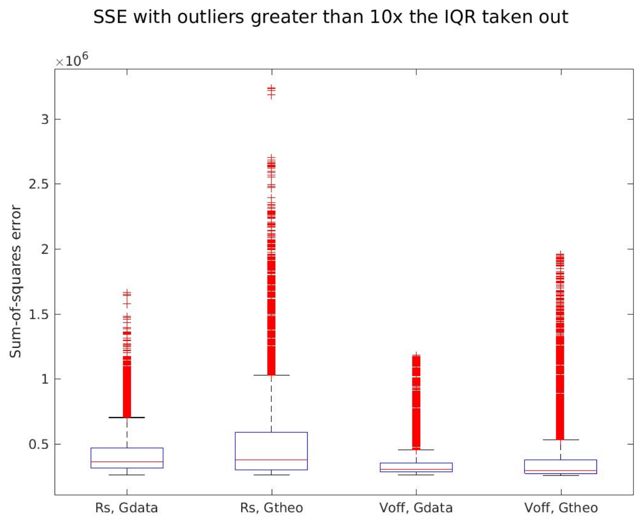




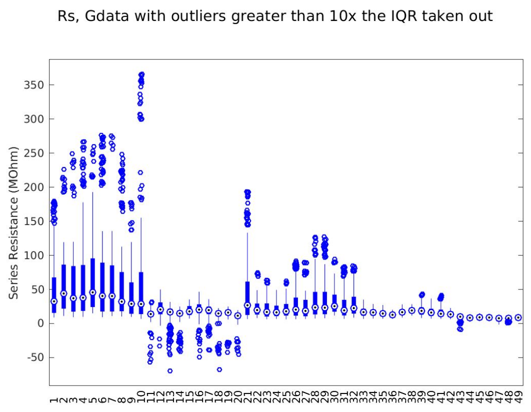


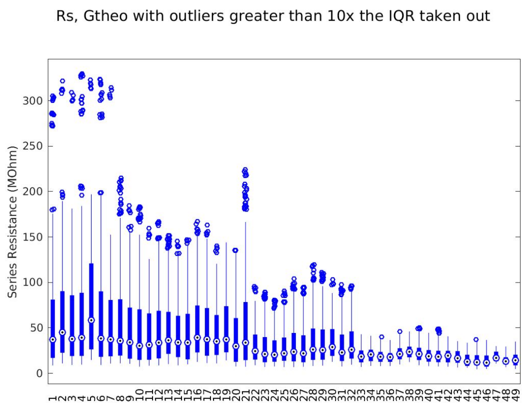
- Created **compare_sse.m** to compare the sum-of-squares error with boxplots after the outliers are taken out (with **remove_outliers.m**)
    - between the different trace correction strategies
    - across cells

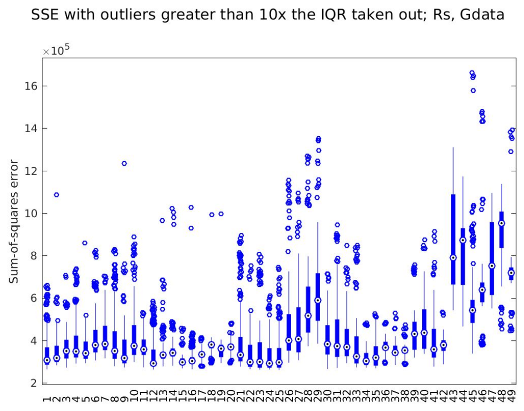
- After taking out outliers of **SSE** with **ratio of whisker length to interquartile range** (**wl2IQR**) set to 10, boxplots were ran again. The **sum-of-square errors** over all traces were on average smaller for the **voltage offset** fits.

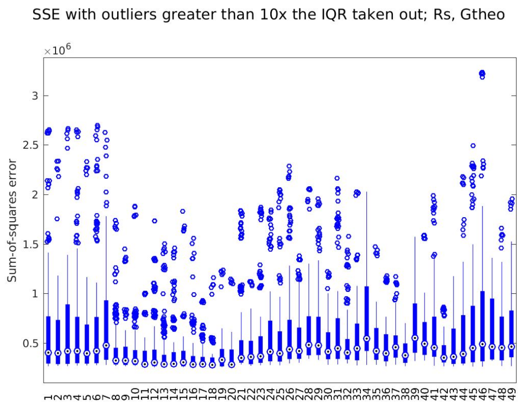


  - The distribution of the fitted series resistances definitely seems to **vary more within cells** than across cells (better for the case using **G data**)

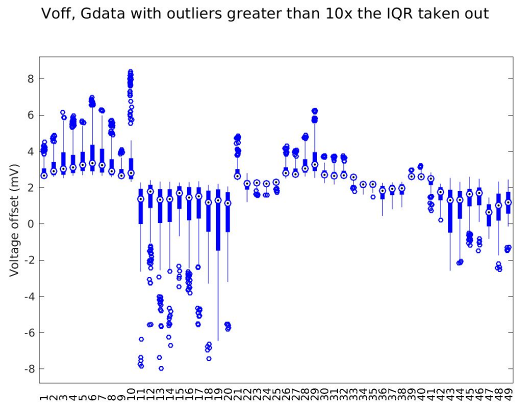


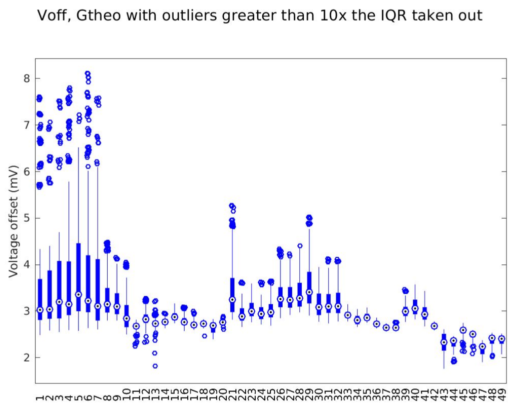


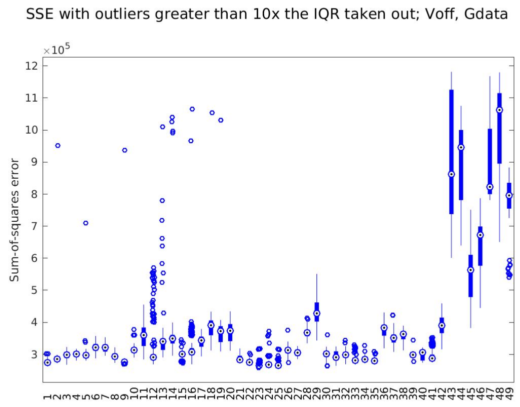


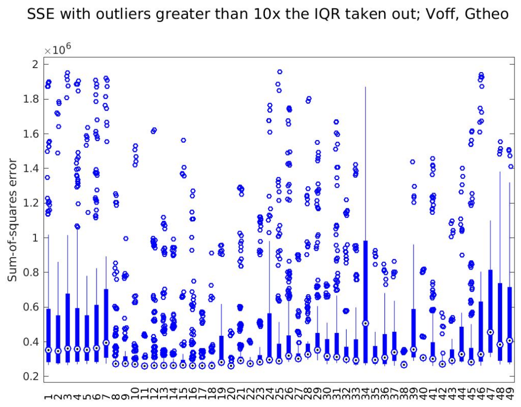


○ However, the distribution of the fitted **voltage offsets** also seems to **vary more within cells** than across cells





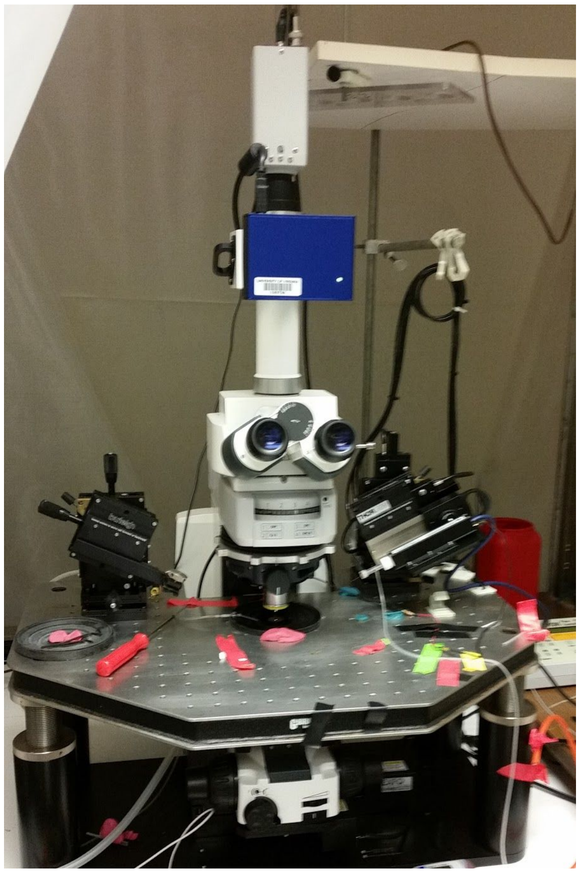




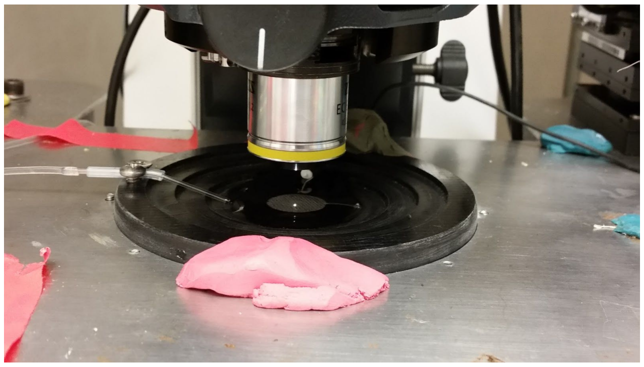
12/2/2016 & 12/9/2016

Set up of Dr. Paula Barrett's patch clamp rig

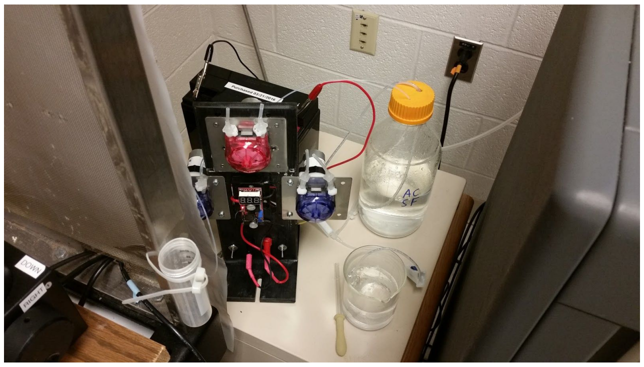
- Microscope:
    - A1 Examiner
    - Has two cameras:



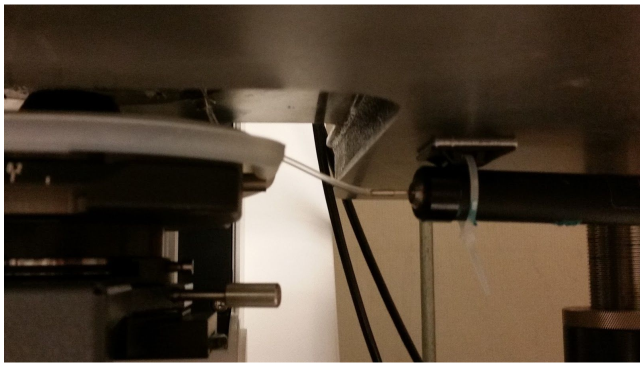
- Chamber size:
    - Must be **110 mm** in diameter; 106 mm is too small
- A **vacuum pipette** was made by Mark (angled and tip-narrowed by heating with a burner) and is secured to the rim of the chamber by **screws**. A ground wire was attached from behind.



- Solution pump was built by Peter & Mark (I added a **switch** to turn on or off **outflow**). **Teflon** tubing was used but was connected with **Tygon R3603** tubing.



- A **solution heater** was secured to the bottom of the microscope by sticky squares in conjunction with **Zip ties**. The solution flows through the Teflon tubing into the heater, than into the chamber from the bottom.



- Flow rate:
  - Initial:
    - 10.0 - 3.2 = 6.8 mL per 2 min
      => 3.4 mL/min
  - After tweaking the knob:
    - 10.0 - 4.1 = 5.9 mL per 2 min
      => **2.95 mL/min**
- Temperature:
  - Initial:
    - **21.8 ℃**, 21.9 ℃, 21.7 ℃
  - After heating to “45.6 ℃”
    - Stabilizes (for at least 7 min) at **31.5~34.5 ℃** around the chamber
    - Center of chamber, around **33 ℃**
  - After 30 min at “45.6 ℃”
    - Center of chamber: 33.1 ℃, **33.3 ℃**
  - After > 2 hours at “45.6 ℃”
    - Center of chamber: **33.3 ℃**
- Bath level was stable, so vacuum strength was already optimal
- Air table uses **N_2** and was working.

- **Carbogen** (**95% O_2**, **5% CO_2**) was connected with **silicone** tubing.
- The **40x** objective originally on the microscope had a narrower thread width and was attached to an adaptor and an adaptive lens, but the attempt to take off the adaptor was unsuccessful. Therefore, the **63x** objective couldn't be placed in

12/9/2016

<u>**Test Dr. Paula Barrett's patch clamp rig**</u>

- ACSF: **296**, **293**, **291**, **297** mmol/kg
  Cutting Solution (NMDG): **305**, **310**, **308** mmol/kg
  Age of mouse: **P23**
- The manipulators for the patch electrode can only move in the **diagonal** (combination of x and z), y or z directions. Initially the coarse manipulator didn't move in the z direction, but it was fixed after manually moving the patch electrode up and down to get rid of some friction
- The **membrane test** wasn't working. New pipettes were tried (with a different ramp temperature), but the recording was basically flat even with the **model cell**.
- A **large air bubble** went under the chamber when the ACSF level was running low. This should be prevented in the future by monitoring ACSF levels.

12/12/2016

<u>Email to Dr. Ferdinando Nicoletti</u>

Dear Dr. Nicoletti,

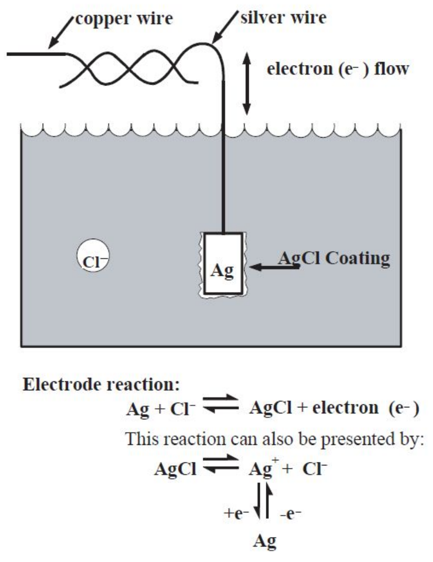
When I saw your poster titled “Modulation of thalamic and cortical GABA transporter: the potential mechanism for the anti-absence activity of mGlu5 receptors” at SFN, you mentioned that it remains unknown whether metabotropic glutamate receptors directly modulate T-type calcium channels in the dendrites of thalamocortical neurons, and that you would consider a possible collaboration to investigate this. I was wondering if you are still interested, as our lab has the appropriate equipment for performing the appropriate electrophysiological recordings. If so, how would you like to proceed?

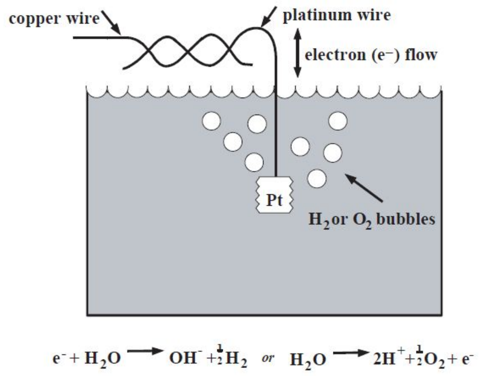
Thanks,
Adam Lu
Beenhakker Lab
Department of Pharmacology
University of Virginia

12/13/2016

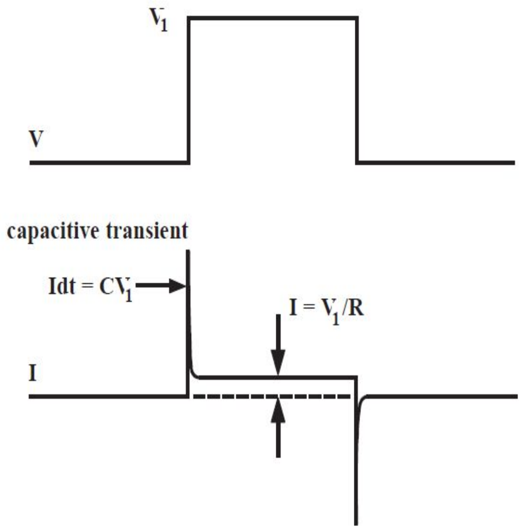
<u>**Notes on The Axon Guide**</u>

- Ch 1: **Bioelectricity**
    - Typically, a single sodium channel passes **10^4 ions/ms** or **1.6 pA** of current
    - At **20 ℃**, a 10-fold change in concentration corresponds to a **58 mV** difference:
      $$E_{rev} = 58 \text{ mV } \log_{10} \frac{[A]_o}{[A]_i}$$
    - **Liquid junction potential**:
        - Since the internal solution and the bath solution often have different ion (e.g., Cl^-) concentrations, there is a difference in **half cell potentials**, creating a liquid junction potential
    - The most common electrode interface is **Ag/AgCl**
        - Internal solution must contain **Cl^-** ions.
        - **Reversible**: $Ag^+ + Cl^- \rightleftarrows AgCl$
        - **Exhaustable**: if **bare Ag** come in contact with the solution, **Ag^+** leaking from the wire can poison many proteins
        - Little polarization, used in **recording**
        - Predictable liquid junction potential
    - The another common electrode interface is **Pt**
        - Internal solution must contain **Cl^-** ions.
        - **Irreversible**: $2\ H_2O \rightarrow 2\ H_2 + O_2$
        - **Inexhaustable**: platinum is inert
        - Local pH changes
        - Pt-hydrogen has little polarization
        - Unpredictable liquid junction potential

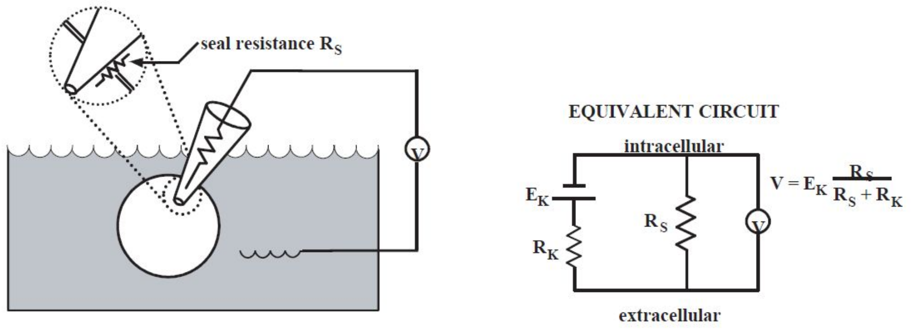




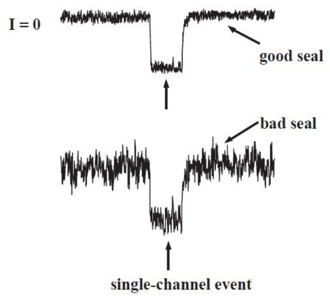
- Cell membranes are typically **< 10 nm** thick, causing a small voltage difference to produce a large electric field. The typical capacitance is **1 μF/cm^2** or **0.01 pF/μm^2**.
- Benefits of a **voltage clamp**:
  - Eliminates the **capacitive current** (other than the brief transients) so that current is proportional to membrane conductance
  - **Channel gating** is often a function of membrane voltage, so can be held constant



- Patch clamp:
  - For single-channel recordings, currents are on the order of **pA**
  - For whole-cell recordings, currents are on the order of several **nA**
  - The **electronic ammeter** must be carefully designed to avoid noise.
- Pipettes:
  - **Glass** is usually used; **quartz** is used for ultra-low-noise single-channel recordings.
- Seal:
  - The **seal resistance** must be very high for the membrane voltage to be accurate:



        - If a current of **N ions/ms** passes through an open channel, then the current will fluctuate from one millisecond to the next with a standard deviation of √N. Same with noise through seal



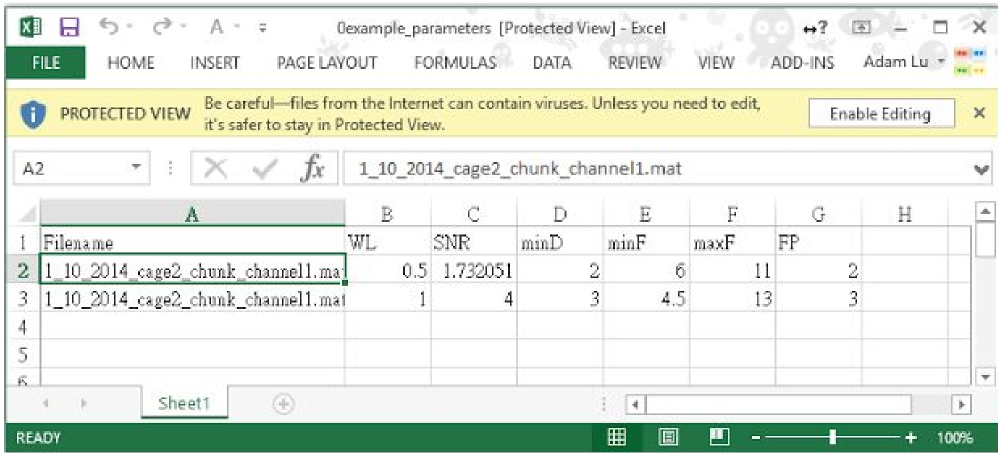
- Ch 2: **The Laboratory Setup**
    - *In vitro* extracellular recordings:
        - For **field potentials** in slices
        - Requires **complex chamber** that warms, oxygenates and perfuses tissues
        - A low-power dissecting microscope with at least **15 cm** working distance
        - Micromanipulator could be **coarse**
        - Voltage range: **10 μV ~ 10 mV**
        - Voltage amplifier: Gain of at least **1,000**
    - Single-channel patch clamp:
        - For patching a **10~20 μm** cell
        - Unperfused culture dish at room temperature
        - Microscope should magnify up to **300~400 fold** with **contrast enhancement**; focus mechanism should move the **objective** not the stage; **inverted** is preferred over dissecting:
            - Easier top access
            - Larger, more solid platform to bolt micromanipulator
        - Micromanipulator should permit fine, smooth movement down to **2 μm/s**
    - Patched slice:
        - Combination of complex chamber and high magnification
    - **Vibration isolation**:
        - Stable, well-designed micromanipulator:
            - The moment arm from the tip of the electrode, through the body of the manipulator, to the cell in the chamber, is as short as possible.
        - Micromanipulator bolted directly to the **microscope stage**
        - Remote-controlled manipulators:
            - **Motorized**: solid & compact; slow & clumsy with **backlash**
            - **Hydraulic**: fast, no backlash; **slow drift**
            - **Piezoelectric**: like motorized but **stepwise**
        - Anti-vibration tables: heavy slab on pneumatic support (partially inflated inner tubes or nitrogen)

- **Electrical isolation**:
    - **Radiative** electrical pickup:
        - Use a **Faraday cage**
        - Shield sources of noise (detect with oscilloscope probe)
        - Use local shielding around the electrode, parts of the microscope and **perfusion tubing**
        - Move away or replace the offending sources
        - Never directly ground the solution other than at the **ground wire**
    - **Magnetically**-induced pickup:
        - **Non-sinusoidal shape** with a frequency that is a higher harmonic of the line frequency
        - Often from electromagnets in **power supplies**
        - Move power supplies away from circuitry
        - **Twist signal wires together** to decrease area of loop cut by magnetic flux
        - Shield the magnetic source with **mu-metal**
    - **Ground-loop** noise:
        - Arises when different grounds of the shielding are at slightly different potentials
        - Connect all the shields and **ground at one place only**
        - Try lifting the BNC shielding or disconnecting some power grounds
        - Try different power sockets -- some may have lower resistance to the mains earth line than others
        - Provide **computers** with a special power line with its own ground or use **optical isolation**
    - **Microelectrode headstages** should be grounded through a low resistance (for instance, **1 MΩ**), whereas **patch-clamp headstages** should be left **open circuit**
- Equipment placement:
    - Small rooms
    - Keep **perfusion stopcocks** and **micromanipulator controllers** off the vibration-isolation table
    - **Oscilloscope** should be at eye level
    - **Computers** should be as far as possible

12/13/2016

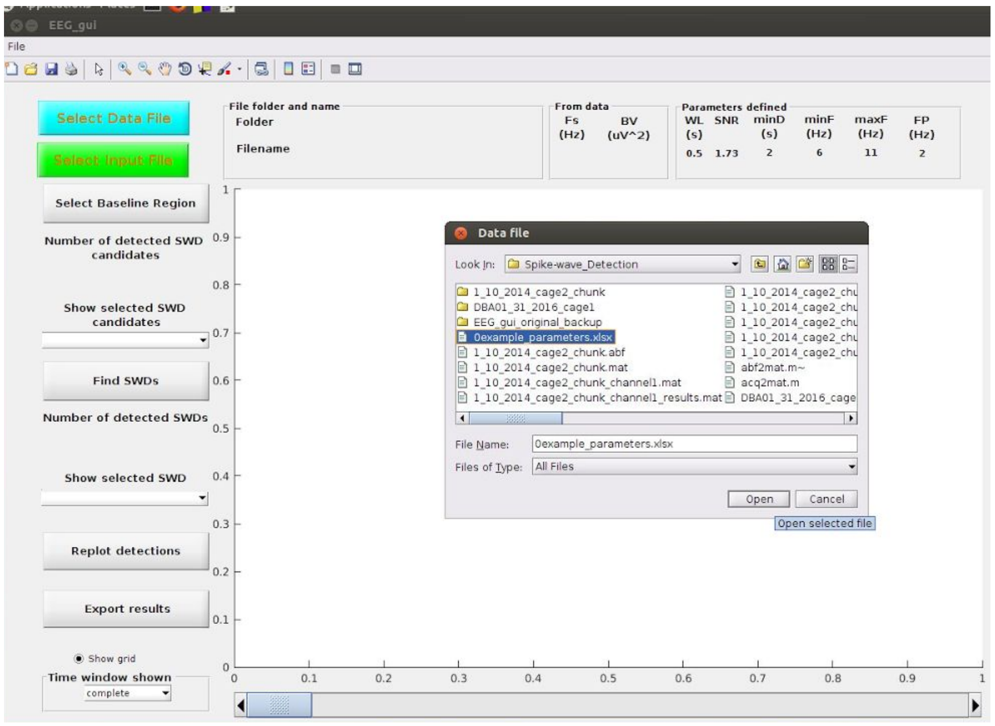
<u>Spike-Wave Detection (cont'd)</u>

- Modifications to **EEG_gui.m**:
    - 2016-12-13 Now saves **parameters** and **detection results** in a matfile for each data file
- Output variables:
    - **datafilename**: full path to data file
    - **params**: parameters used in the detection
    - **params_log**: Contains the name and units of parameters according to **params**
    - **SWD_cands**: contains 3 column vectors: start_times, finish_times, durations
    - **SWDs**: contains 3 column vectors: start_times, finish_times, durations


0example_parameters [Protected View] - Excel
FILE
HOME
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
ADD-INS
Adam Lu
PROTECTED VIEW
Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
A2
Sheet1
READY
100%

1_10_2014_cage2_chunk_channel1.mat

	A	B	C	D	E	F	G	H
1	Filename	WL	SNR	minD	minF	maxF	FP	
2	1_10_2014_cage2_chunk_channel1.mat	0.5	1.732051	2	6	11	2	
3	1_10_2014_cage2_chunk_channel1.mat	1	4	3	4.5	13	3	
4								
5								



Parameters defined

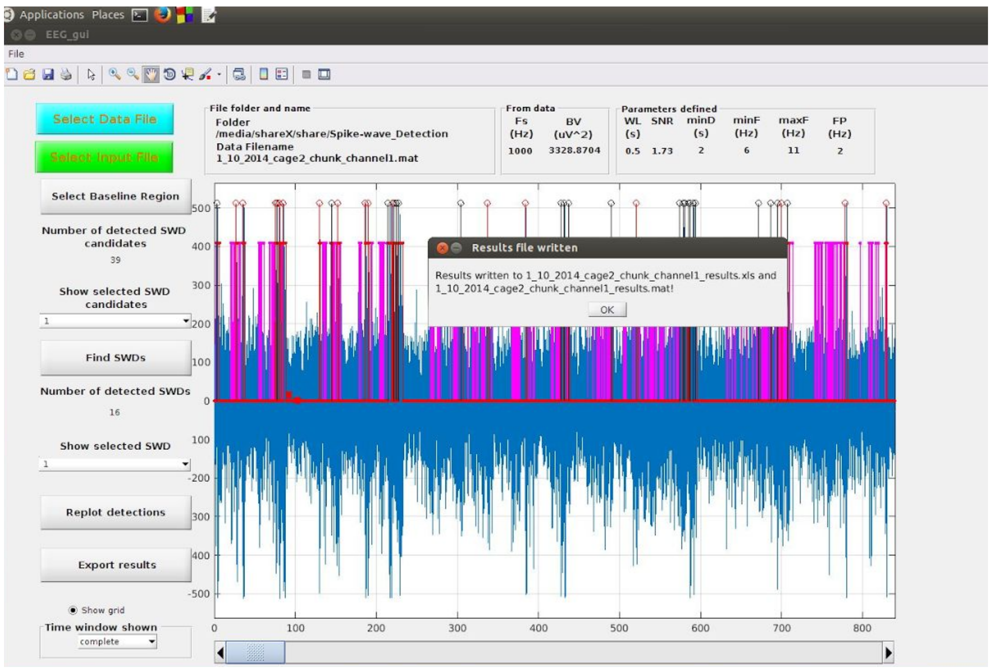
WL (s)	SNR	minD (s)	minF (Hz)	maxF (Hz)	FP (Hz)
0.5	1.73	2	6	11	2

Look In: Spike-wave_Detection

```
1_10_2014_cage2_chunk                         1_10_2014_cage2_chu
DBA01_31_2016_cage1                           1_10_2014_cage2_chu
EEG_gui_original_backup                       1_10_2014_cage2_chu
0example_parameters.xlsx                      1_10_2014_cage2_chu
1_10_2014_cage2_chunk.abf                     1_10_2014_cage2_chu
1_10_2014_cage2_chunk.mat                     abf2mat.m~
1_10_2014_cage2_chunk_channel1.mat            acq2mat.m
1_10_2014_cage2_chunk_channel1_results.mat    DBA01_31_2016_cage
```

File Name: 0example_parameters.xlsx

Files of Type: All Files



File folder and name
Folder
/media/shareX/share/Spike-wave_Detection
Data Filename
1_10_2014_cage2_chunk_channel1.mat

From data

Fs (Hz)	BV (uV^2)
1000	3328.8704

Parameters defined

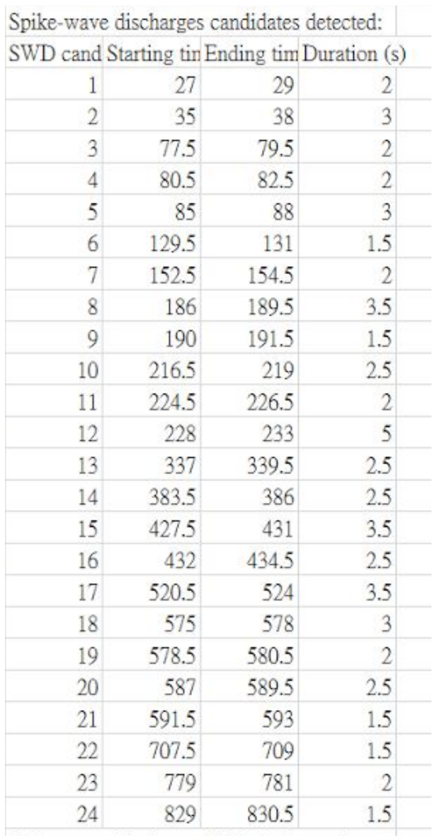
WL (s)	SNR	minD (s)	minF (Hz)	maxF (Hz)	FP (Hz)
0.5	1.73	2	6	11	2

Number of detected SWD candidates
39

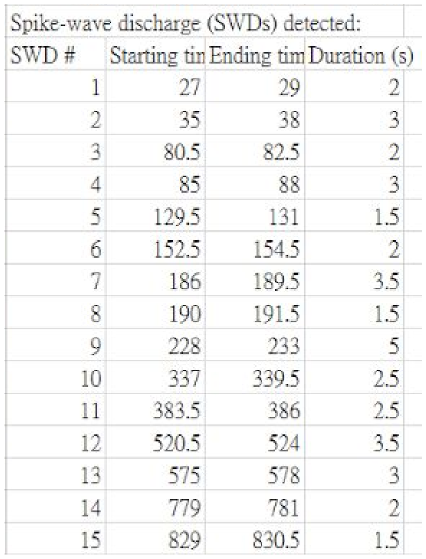
Number of detected SWDs
16

Results file written

Results written to 1_10_2014_cage2_chunk_channel1_results.xls and 1_10_2014_cage2_chunk_channel1_results.mat!


Spike-wave discharges candidates detected:

SWD cand	Starting tin	Ending tim	Duration (s)
1	27	29	2
2	35	38	3
3	77.5	79.5	2
4	80.5	82.5	2
5	85	88	3
6	129.5	131	1.5
7	152.5	154.5	2
8	186	189.5	3.5
9	190	191.5	1.5
10	216.5	219	2.5
11	224.5	226.5	2
12	228	233	5
13	337	339.5	2.5
14	383.5	386	2.5
15	427.5	431	3.5
16	432	434.5	2.5
17	520.5	524	3.5
18	575	578	3
19	578.5	580.5	2
20	587	589.5	2.5
21	591.5	593	1.5
22	707.5	709	1.5
23	779	781	2
24	829	830.5	1.5


Spike-wave discharge (SWDs) detected:

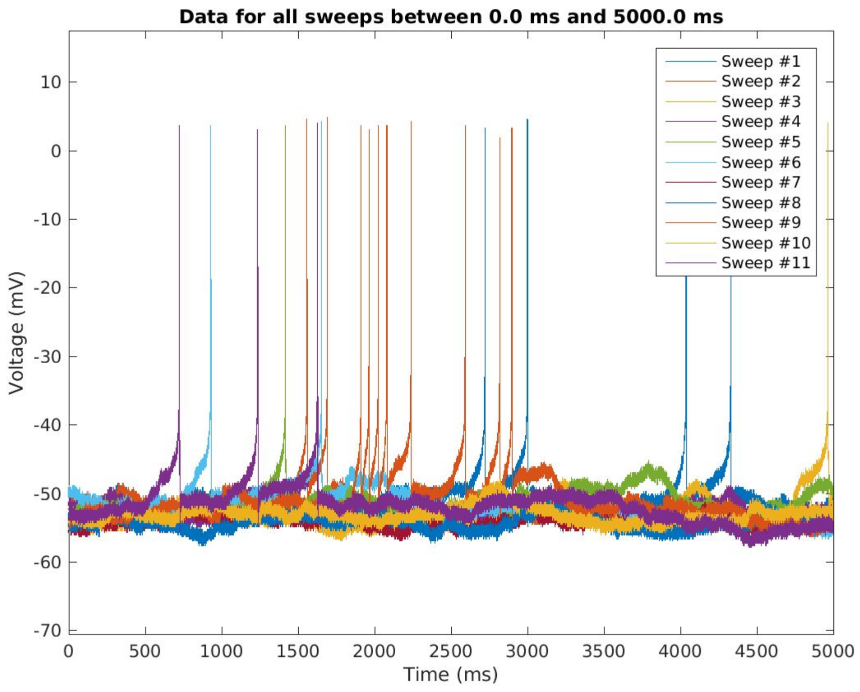
SWD #	Starting tin	Ending tim	Duration (s)
1	27	29	2
2	35	38	3
3	80.5	82.5	2
4	85	88	3
5	129.5	131	1.5
6	152.5	154.5	2
7	186	189.5	3.5
8	190	191.5	1.5
9	228	233	5
10	337	339.5	2.5
11	383.5	386	2.5
12	520.5	524	3.5
13	575	578	3
14	779	781	2
15	829	830.5	1.5

12/14/2016

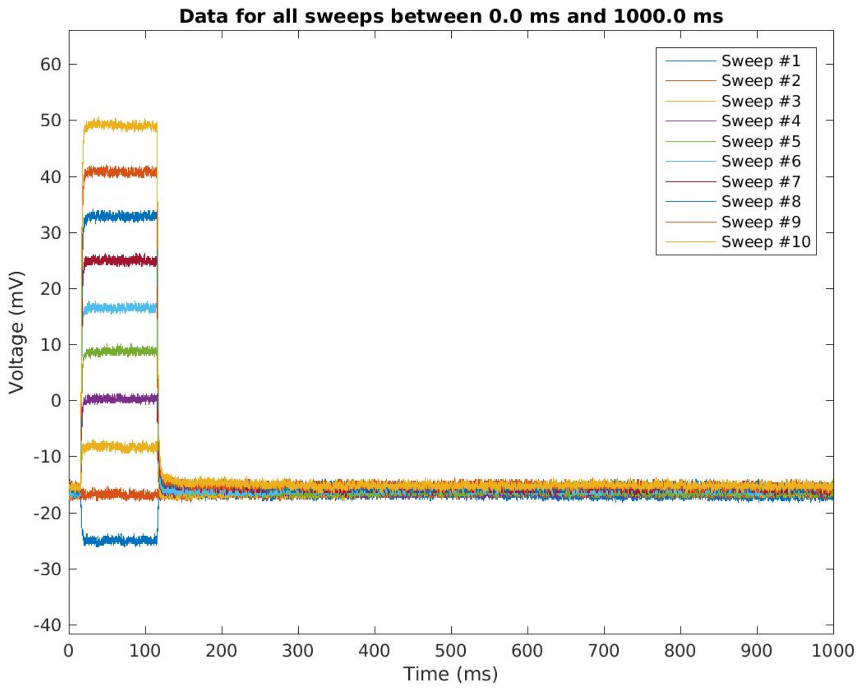
Test Dr. Paula Barrett's patch clamp rig

- ACSF: **296**, 297, 300, 296 mmol/kg
- Cutting Solution (NMDG): **290, 295** mmol/kg
- Recalibrated: Standard was 286, 288, 293 → 290 mmol/kg
- Cutting Solution (NMDG) after calibration: **289** mmol/kg
- ACSF after calibration: **297** mmol/kg
- Age of **rat**: **P13**
- Switched objective to **63x**, but the camera had an extra magnification that limited the field of view, so switched back to **40x**
- Made **flat gold bars** from gold wire and hammer. This was essential for keeping the slices flat
- There was a discrepancy between the **center of image** under the **eyepiece** and the center of image under the **camera** -- the latter was to the **left and above** of the former. The center of image under the **camera** was more accurate, because after switching the objective to 40x, pipette tips were easier to find if it was originally centered under the camera.
- There was a leak in the **pipette holder**, which prevented positive and negative pressures from working. The leak was repaired after the **orange gasket** was replaced. The pressures were checked with a **manometer**.
- On the Axopatch 200B, the **knob** for setting the holding VClamp and holding IClamp was the same. The knobs are now set to ∓60 mV by default when in VClamp mode. If this knob was active, any value in the protocol of PClamp will be added on. Instead of using this knob on the amplifier, a constant holding potential/holding current could also be set in PClamp on the upper left hand side.
- The ACSF ran out at the end. **2 liters** instead of 1 liter should be made in the future.
- **Oxygen tubing** fell out for an undetermined amount of time after the change of ACSF bottle, which might have compromised the health of the slices. A **clip** on the tubing should be used in the future.
- Pipettes:
    - Pulled from **borosilicate glass** of **outer diameter 1.5 mm** and **inner diameter 0.86 mm**
    - Program **55**:
        - Ramp value: **655**
        - x2: Heat: **622**, Vel: **39**, Time: **250**
        - x1: Heat: **623**, Vel: **39**, Time: **250**
        - Pressure: **500**; Air flush: **10** seconds
        - Jaws @ **70 ℃**
    - The **pipette resistances in bath** noted in this experiment was:
        - 8.1 MΩ, 9.4 MΩ, 12.4 MΩ, 10.1 MΩ, 10.2 MΩ, 13.2 MΩ, 12.5 MΩ
        - Average: **10.8 MΩ**

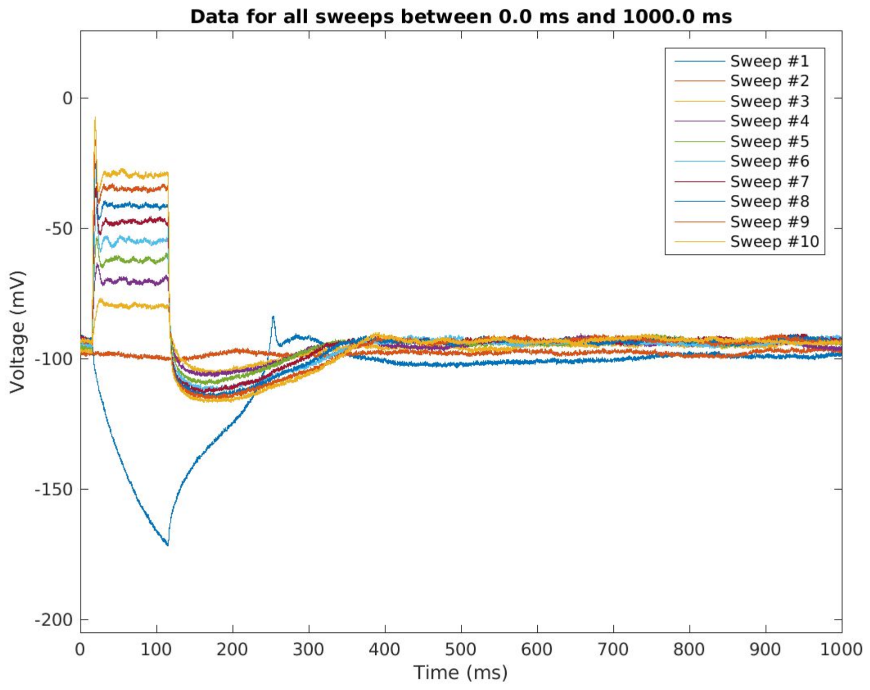
- Results from **A20161214** (a killed RT neuron):
    - Initial condition:



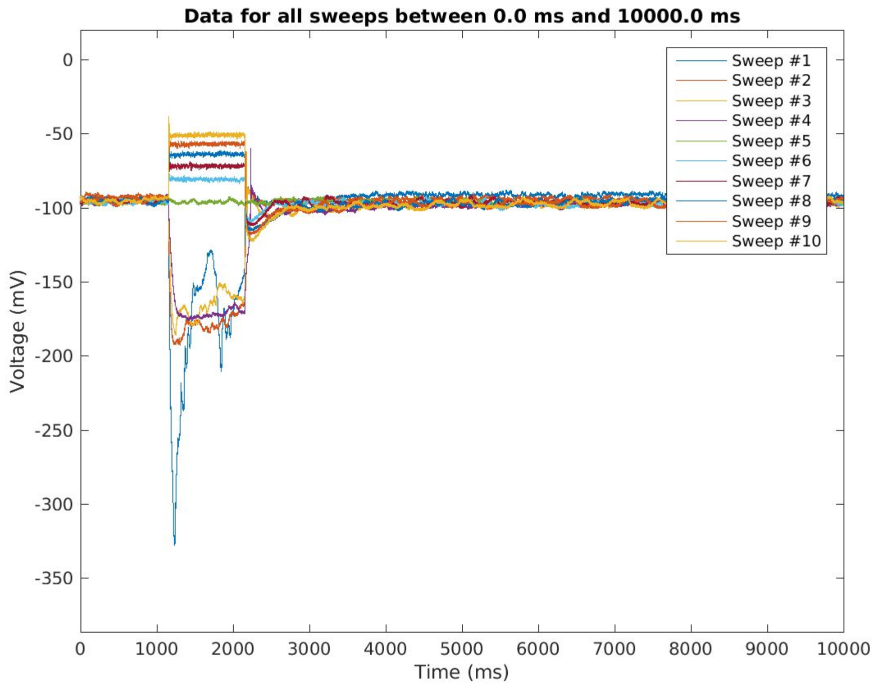
    - Current Injections after an inappropriate current injection:



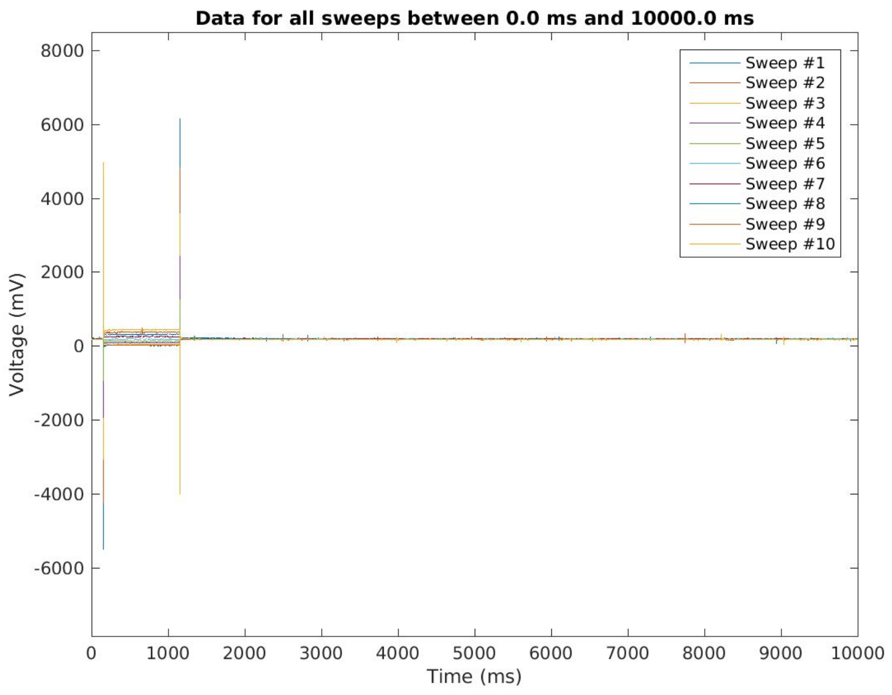
- Results from **B20161214** (a silent RT neuron):
  - Current injections



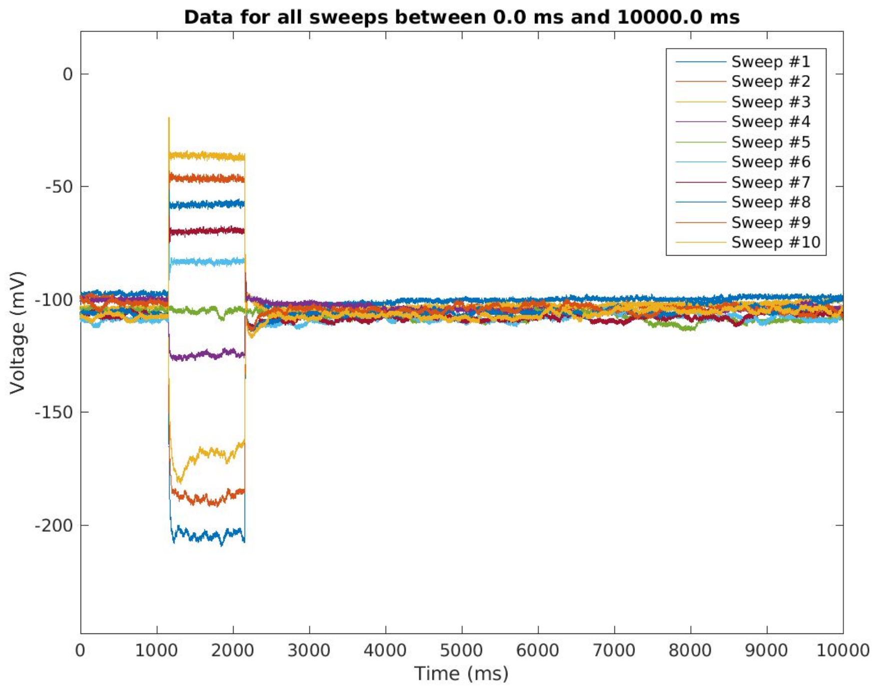
  - Less happy:



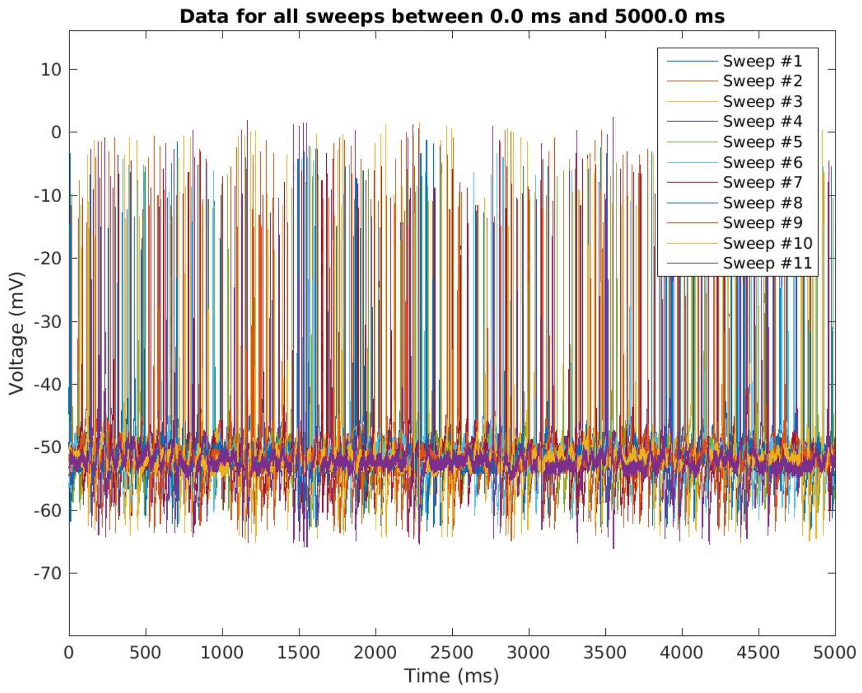
- Voltage clamp (Note: y-axis should be **Current (pA)**):



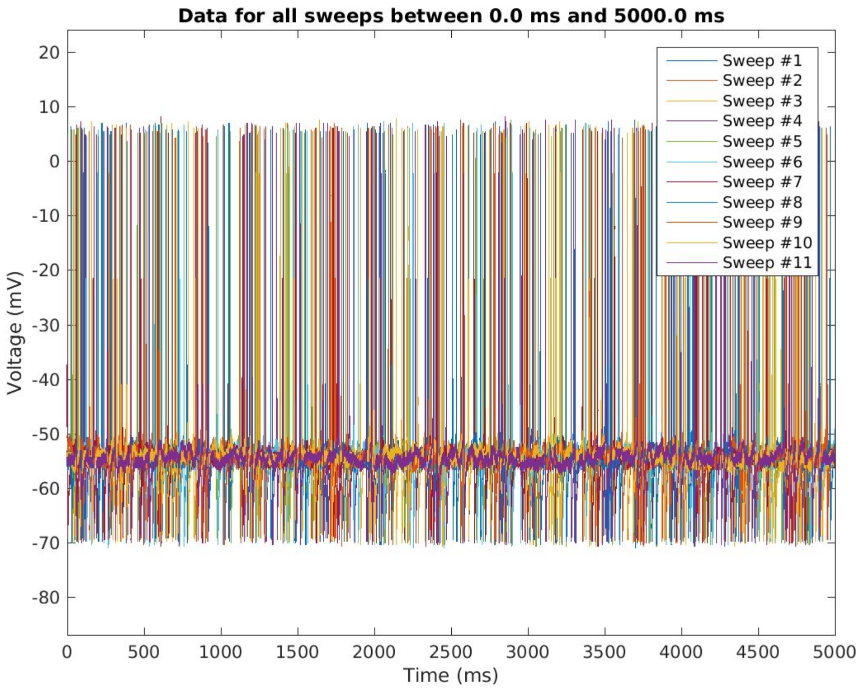
- Cell died?



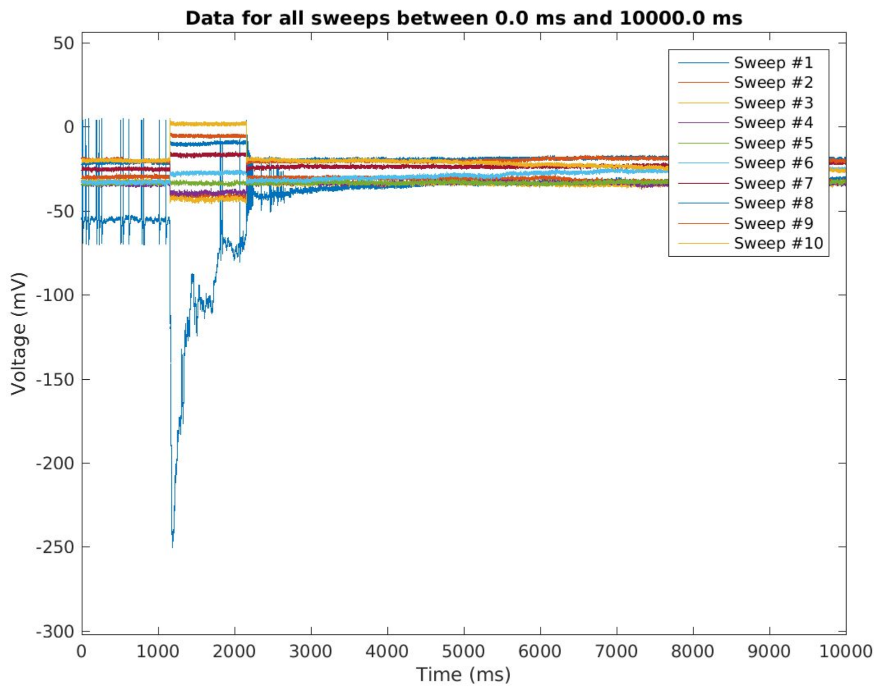
- Results from **C20161214** (an overactive RT neuron):
    - Initial condition:



    - As resting membrane potential becomes more negative, **spike amplitudes** increase:



- ○ Killed with current injection, why?



12/15/2016

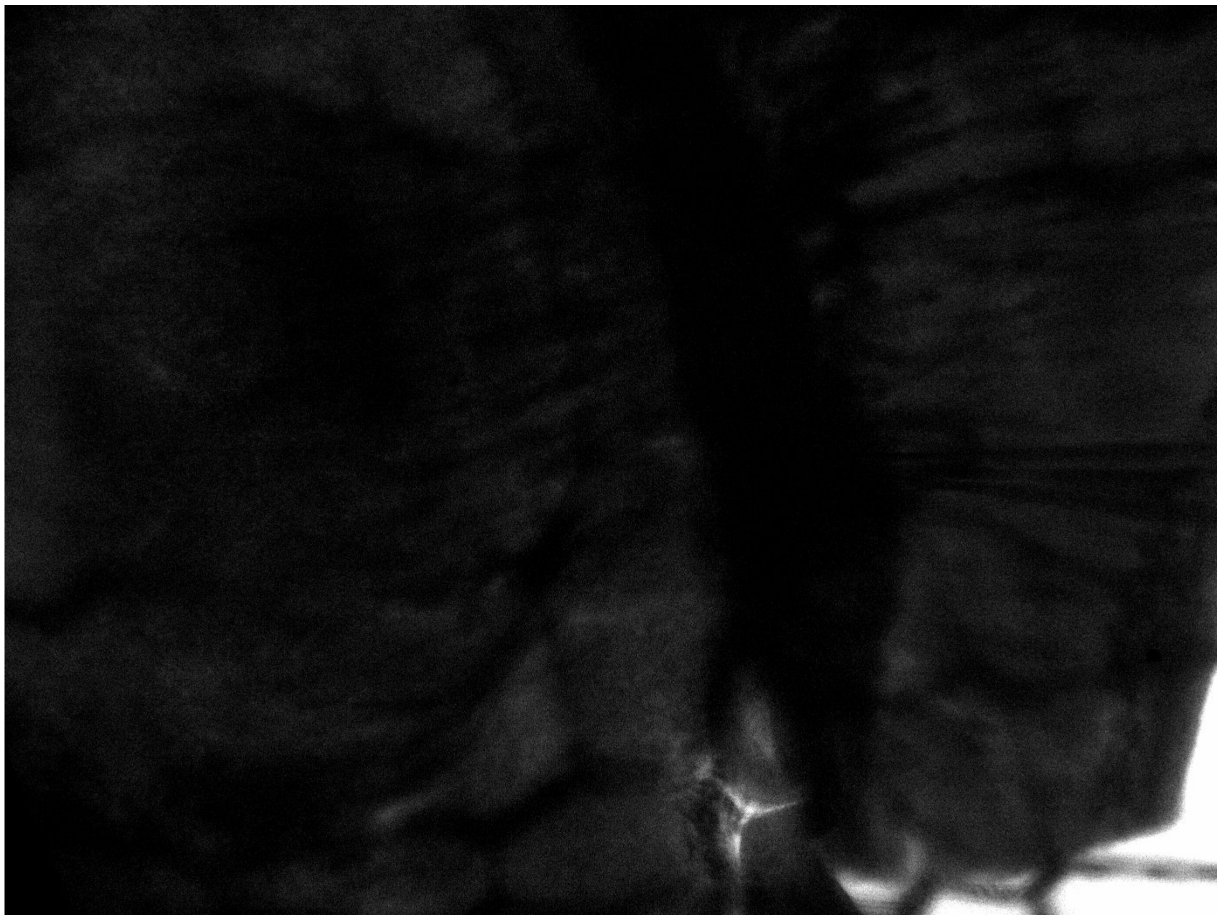
Troubleshoot Dr. Paula Barrett's patch clamp rig

- AxioCam was not connected to the computer initially. Turns out that the **PCI interface board** was not inserted correctly in the computer and was thus not connected.
- **Pclamp protocols** were aberrant because the Input/Output channels were incorrect. This was fixed by adding new channels to **Lab Bench** and using the scaling factors written on the back of the amplifier, as well as Mark's **Model Cell Tester**, to find the correct **scaling factors**.
- An extra cable was added so that the **current commands** can be recorded in PClamp as well.

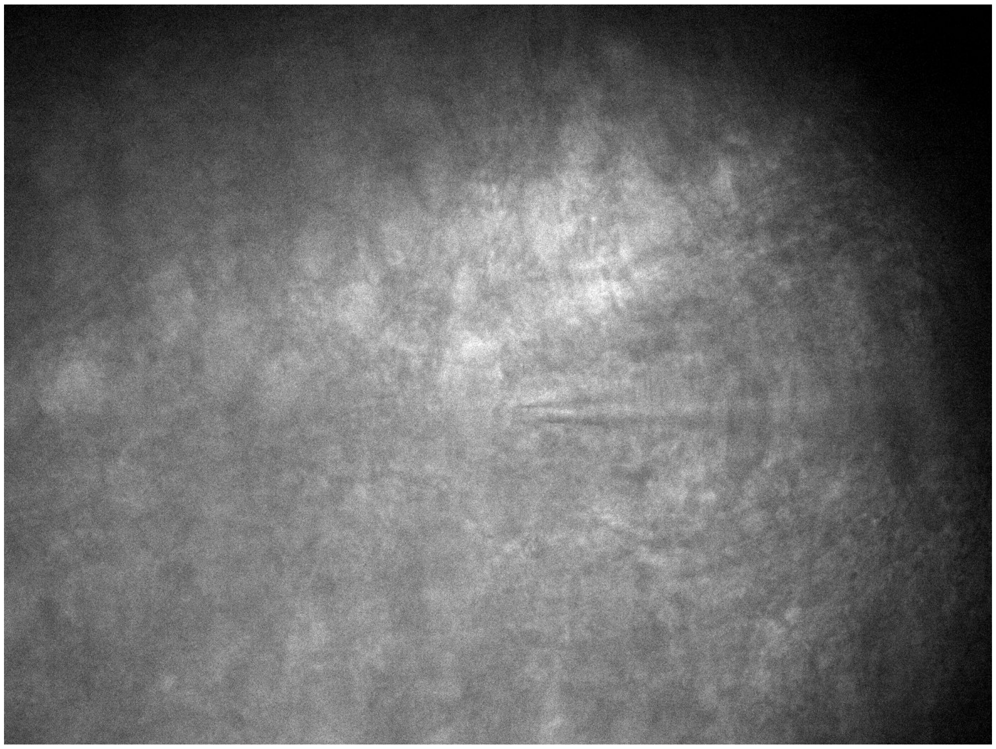
12/16/2016

<u>Test Dr. Paula Barrett's patch clamp rig</u>

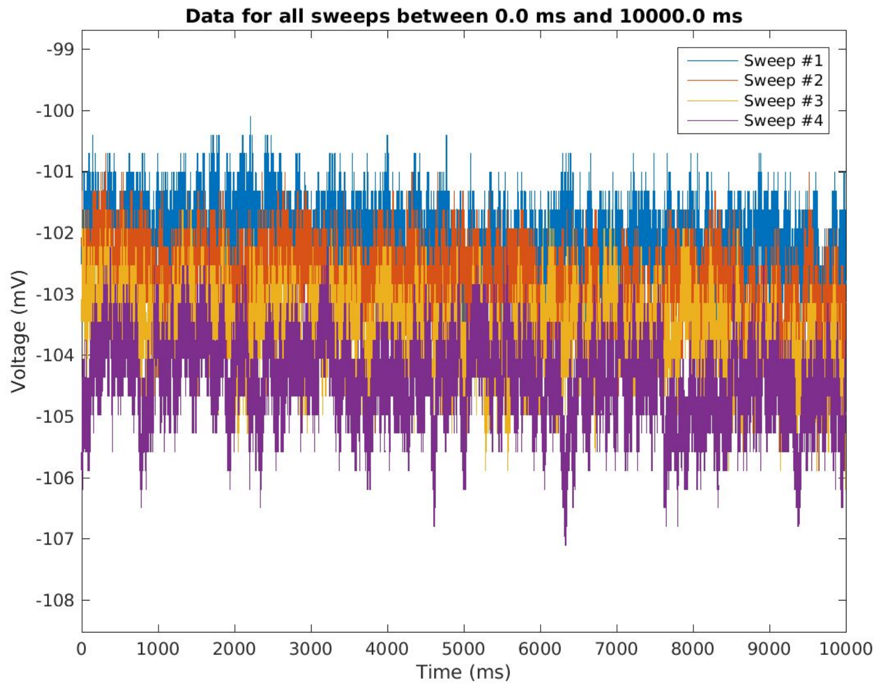
- ACSF: **291** mmol/kg
- Age of **rat**: **P15**
- The image under **AxioCam** is most likely the same magnification as the image under the eyepiece, but is slightly **cropped**.
- Pipette settings were as before. The **pipette resistances in bath** noted in this experiment was:
    - 8.8 MΩ, 6.3 MΩ, **11.0 MΩ** (**A20161216**), 9.1 MΩ, 11.6 MΩ
    - Average: **9.4 MΩ**
- The **pipette offset** was initially not set correctly for **A20161216**, which caused the voltage recordings to be more negative by about 60 mV. This was subsequently corrected halfway through the recording.
- The **voltage trace** during **episodic stimulation mode** (but not gap-free mode) was initially at **0** in the ClampEx interface. However, the recorded abf files show the correct voltage levels (i.e., the **telegraphed baseline reading** from the amplifier was apparently added only after the recording is saved).
- Results from **A20161216** (a normal RT neuron):
    - Image 2.5x of **RT**:



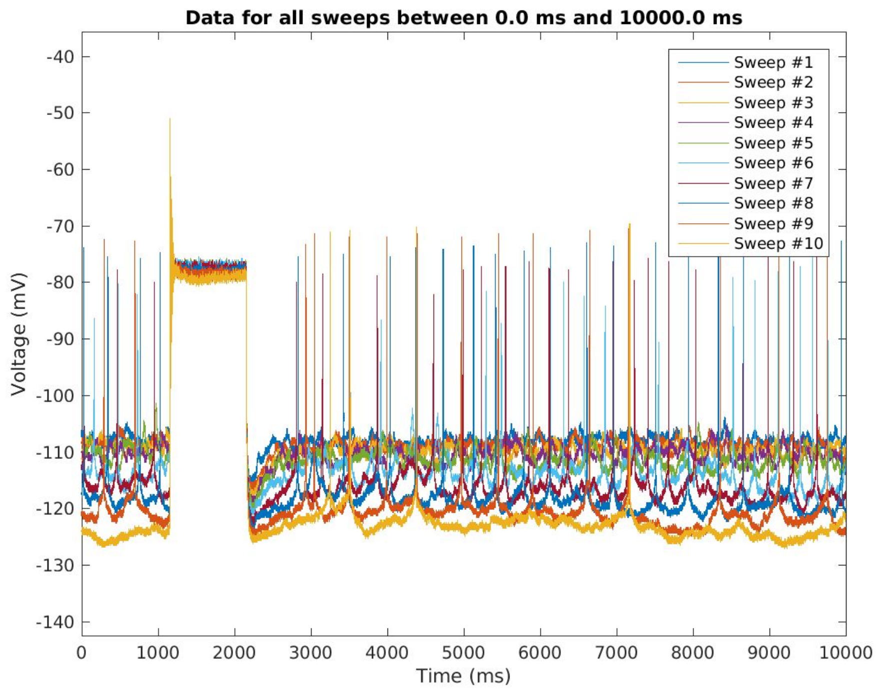
- Image 40x:



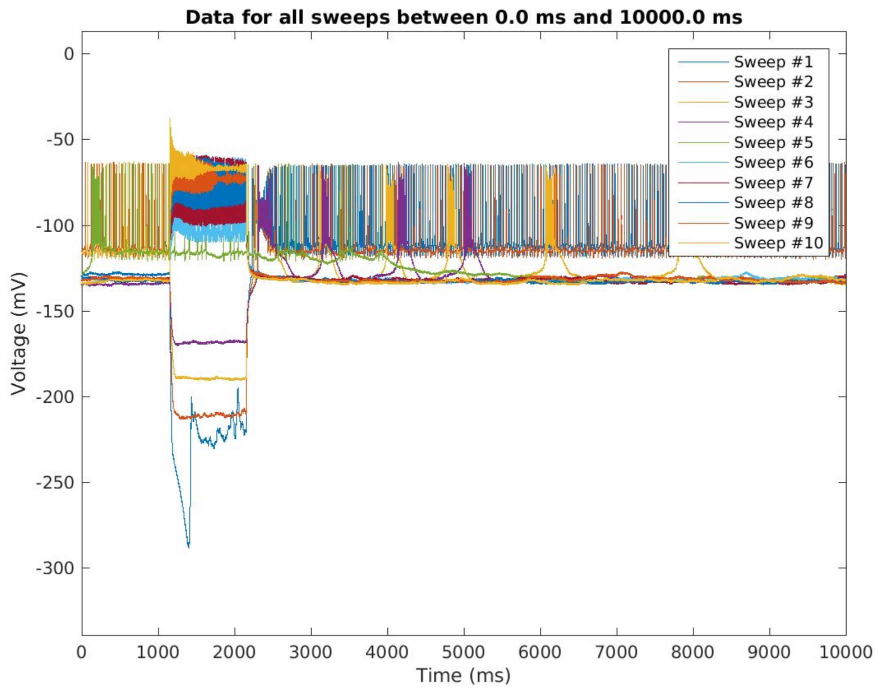
- Initial condition (Note: all voltage values are **shifted by -60 mV**):



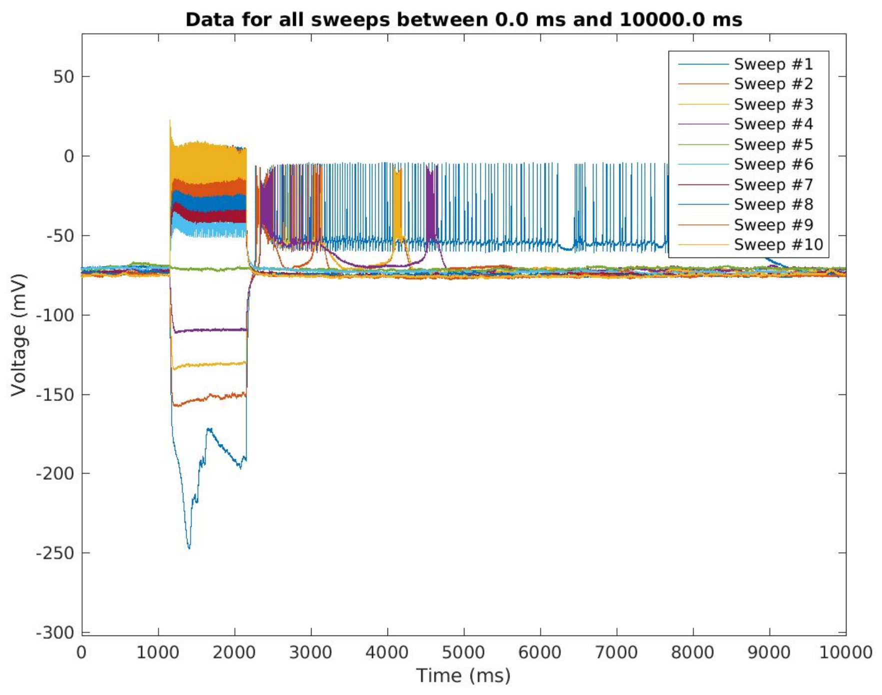
- Current injections **+300 pA** repeated 10 times (Note: all voltage values are **shifted by -60 mV**), under more depolarized resting membrane potentials (tonic mode):



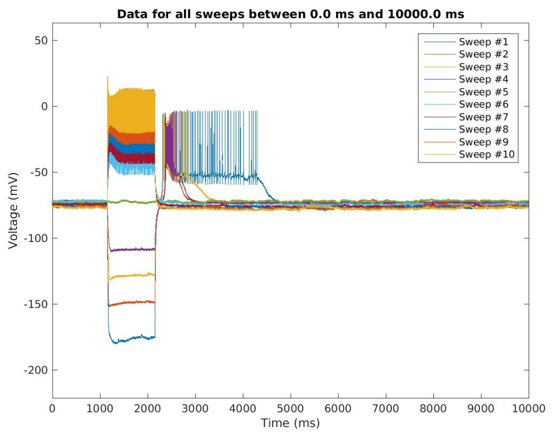
- Current injections (**-400 pA** to **+500 pA** with 100 pA step) under hyperpolarized resting membrane potentials (**bursting mode**)

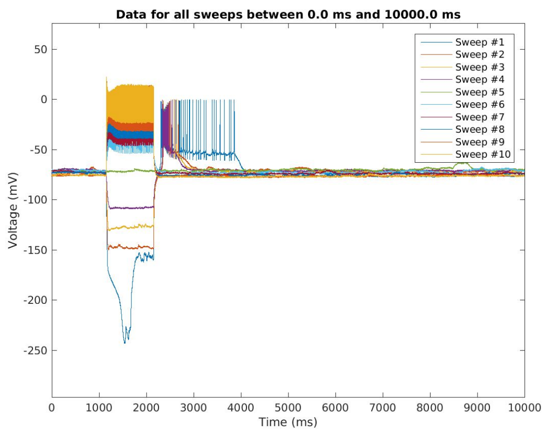


- Current injections (**-400 pA** to **+500 pA** with 100 pA step) after **pipette offset correction**

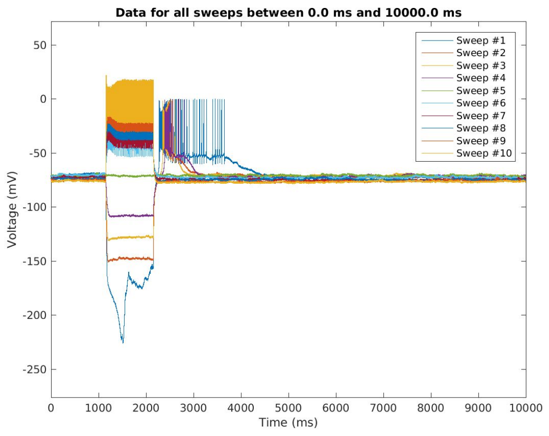


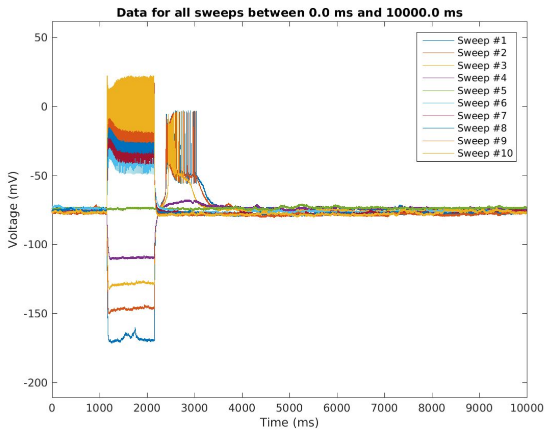
- Bursting not as pronounced as time progresses. Is this a change in channel gating or a change in network activity?

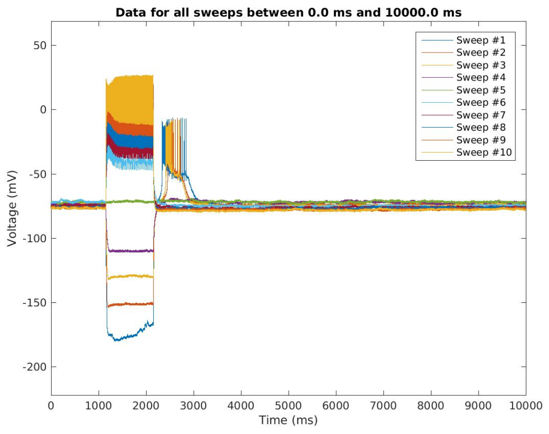


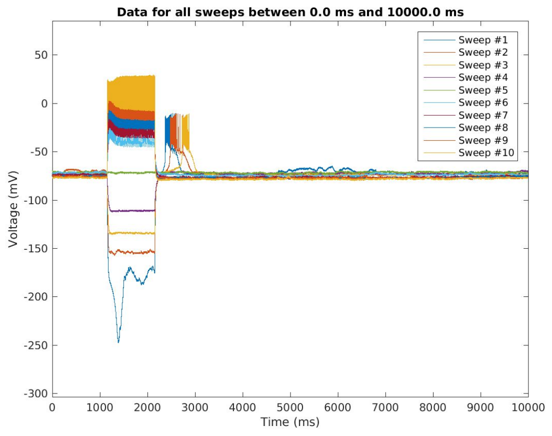


20161218

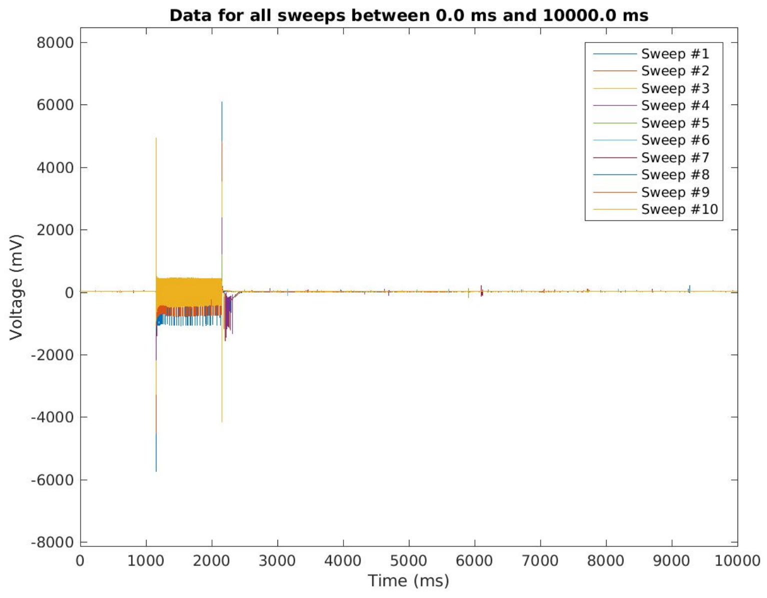




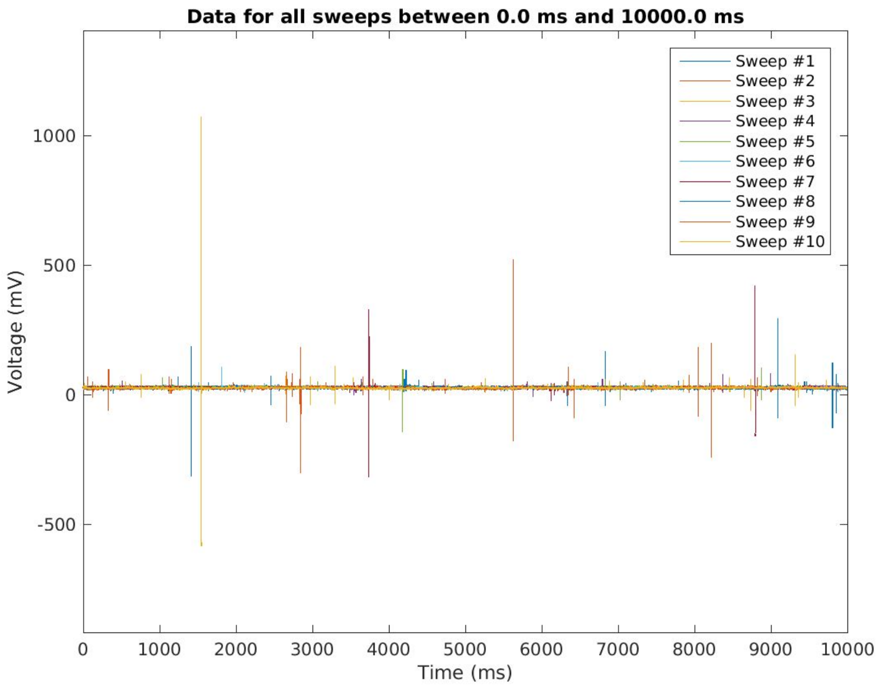




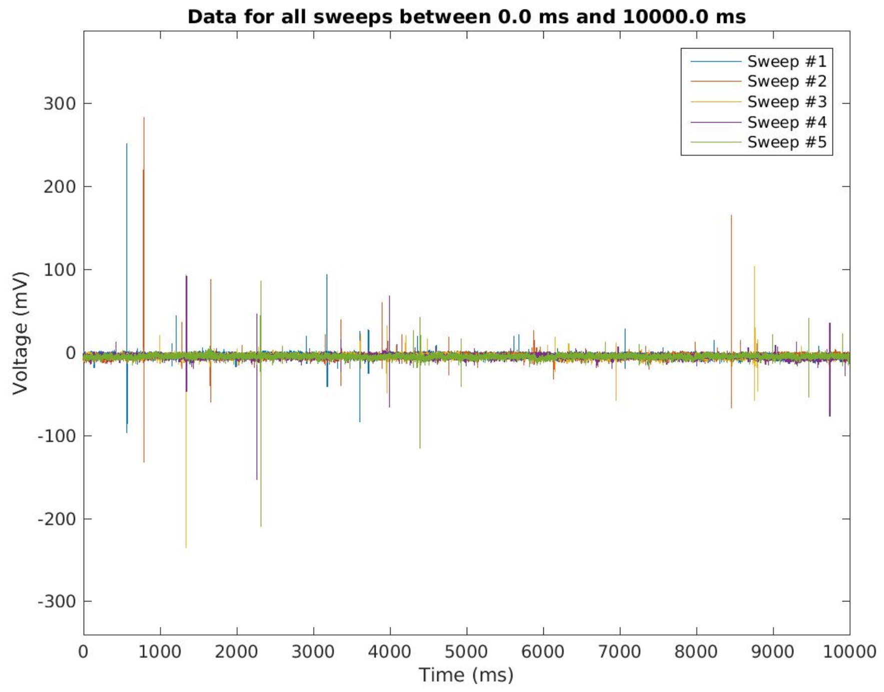
- Voltage step (@ **-60 mV**, then step at **-110 mV** to **-20 mV** with step **10 mV**):
  Note: y-axis should be **Current (pA)**



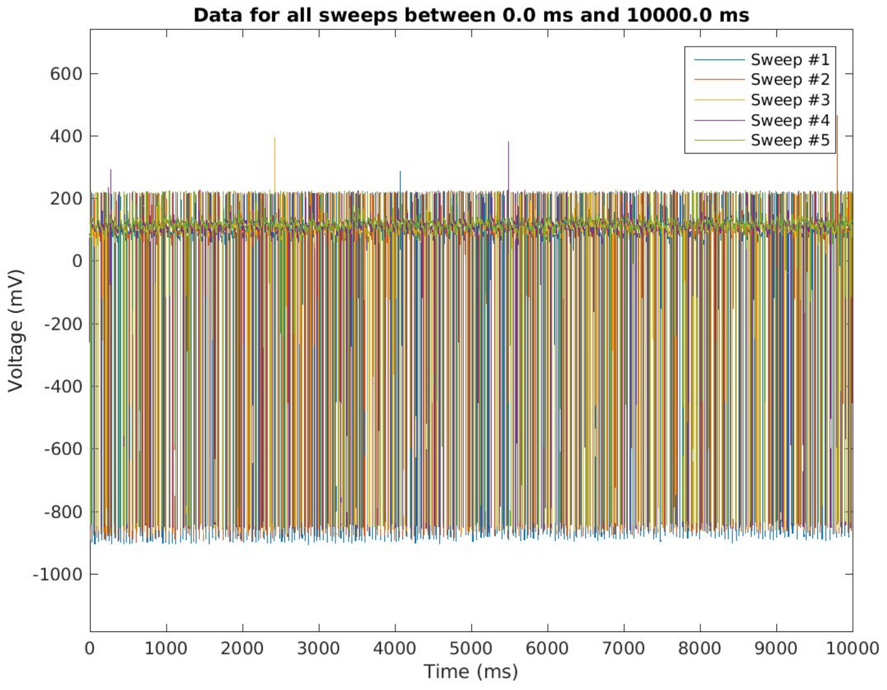
- Voltage clamp @ **-60 mV** (Note: y-axis should be **Current (pA)**)



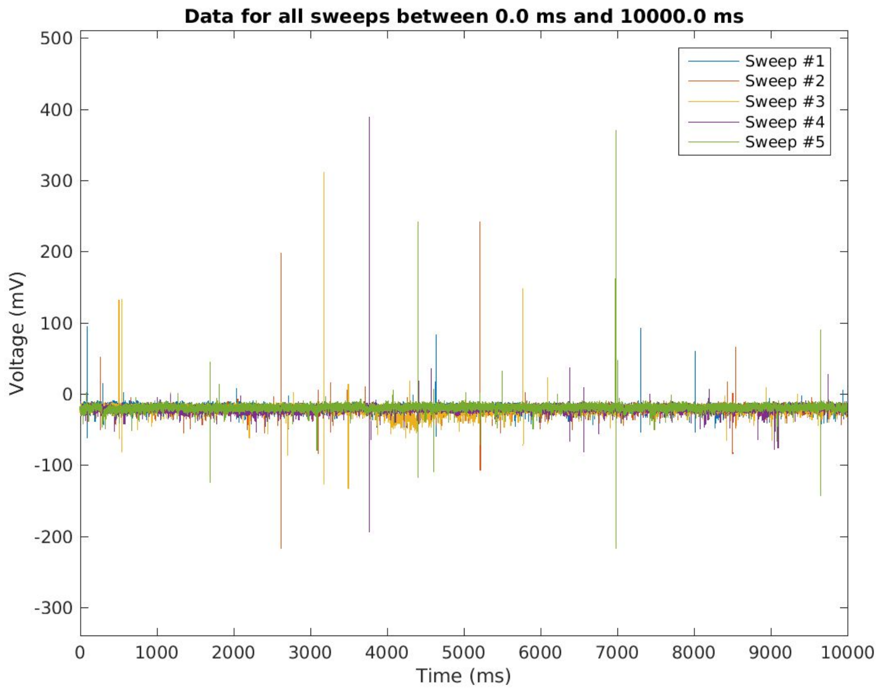
- Voltage clamp @ **-75 mV** (Note: y-axis should be **Current (pA)**)



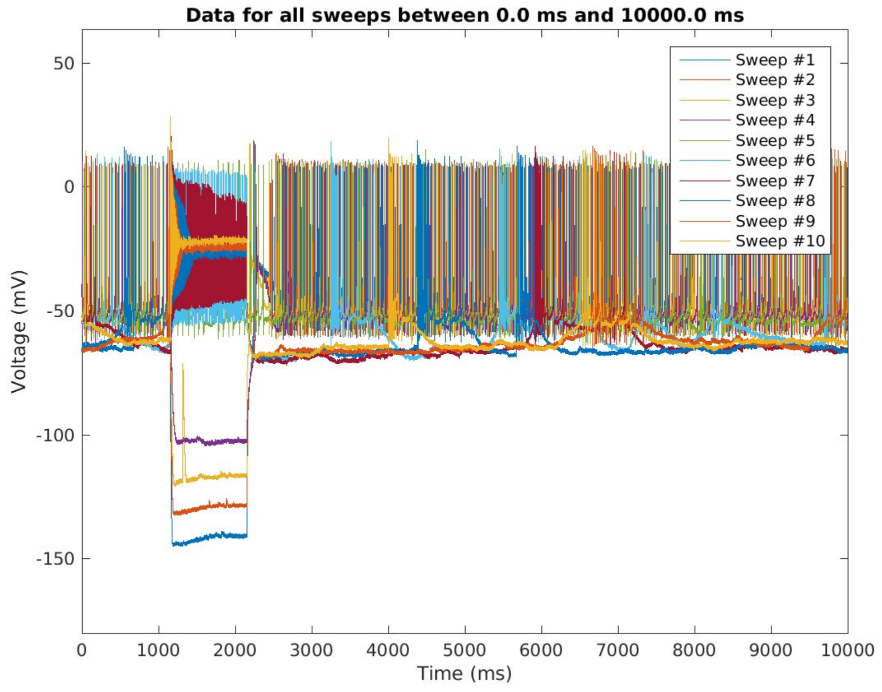
- Voltage clamp @ **-40 mV** (Note: y-axis should be **Current (pA)**)



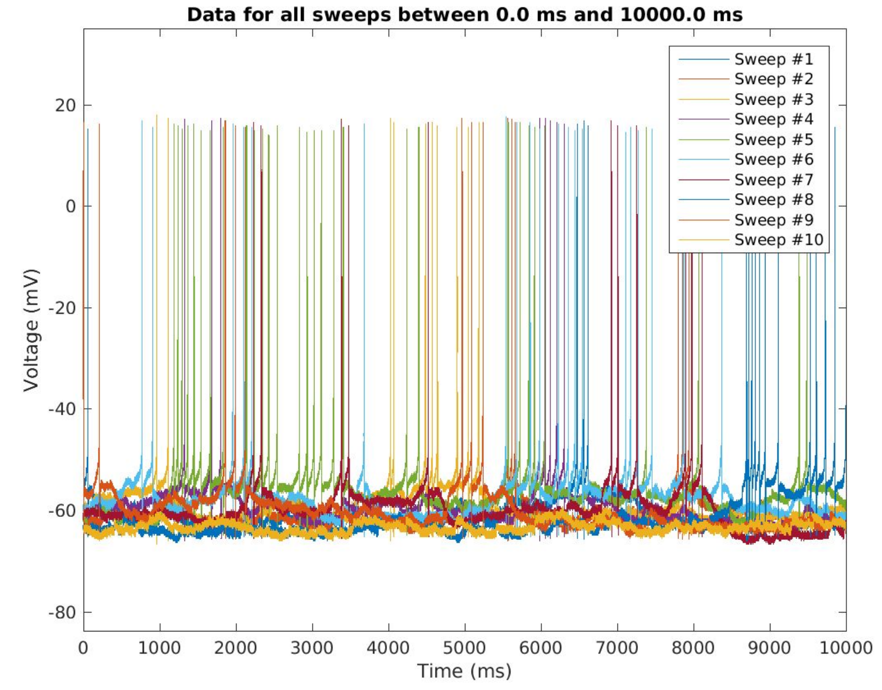
- Results from **B20161216** (an overactive RT neuron):
  - Voltage clamp @ **-70 mV** (Note: y-axis should be **Current (pA)**)



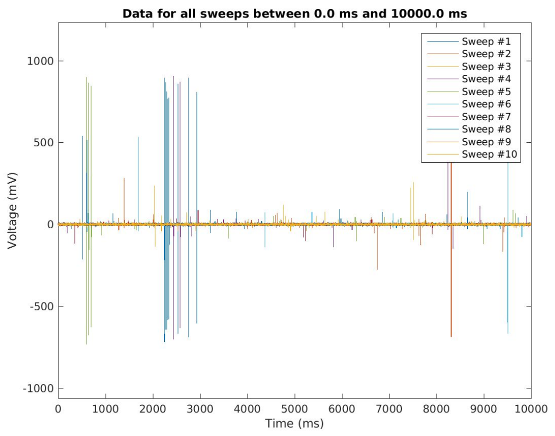
- Current injections (**-400 pA** to **+500 pA** with 100 pA step)

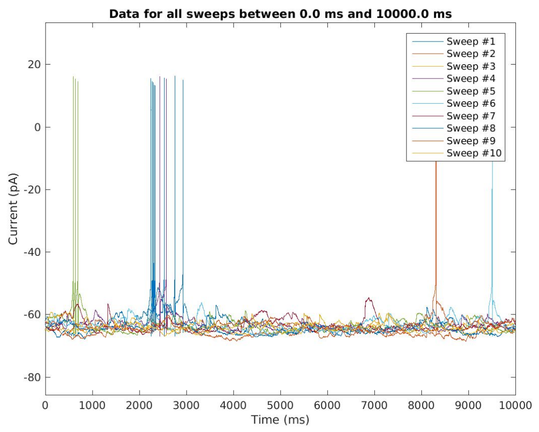


- I = 0

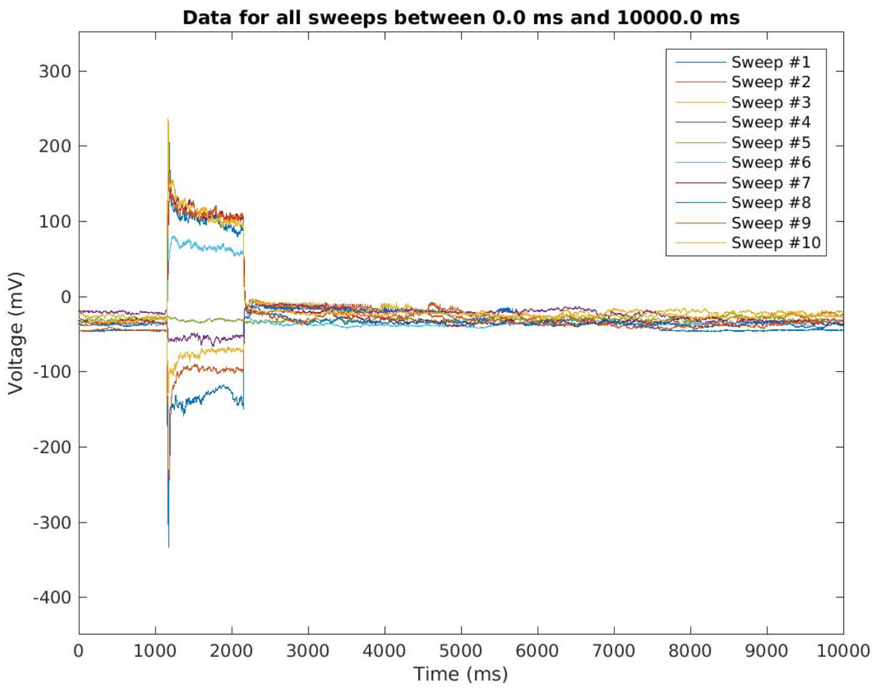


- Voltage clamp @ **-60 mV** (Note: y-axis should be **Current (pA)** on the left and **Voltage (mV)** on the right). Why is there **poor clamping**?





- Cell died



<u>Plan for this week and week after break</u>

- Patching:
    - Read the AXON Guide, Ch 3
    - Fix **plot_traces_abf.m** so that voltage clamp recordings are graphed appropriately
    - Apply drugs such as **K_{ATP} blockers**

- EEG:
    - Practice **implants**
    - Read about **microdialysis**

- Data analysis:
    - Write **code** for **analyzing scoring results**
    - Await response from John about the **noisy recordings**
    - Score & analyze scoring results

- Passive fitting:
    - Figure out a threshold in the histograms and **take out noisy traces** in the curve fitting method
    - Await response from John on questions about the **passive fitting**
    - Try the fitting of current pulse response with **simulations**

- SWD detection:
    - Figure out how to **screen** through detection results

10/20/2016~10/21/2016

Notes from Destexhe et al 1996a

- Background Facts:
    1. Anatomy:
        a. Reticular thalamic (**RE**) neurons receive collaterals from most thalamocortical and corticothalamic fibers
    2. Burst features compared to TC neurons:
        a. **Broader** and develop **more slowly** (from slower activation and inactivation kinetics)
        b. **Higher burst threshold**: stronger current pulses are needed to evoke bursts
        c. **Accelerando-decelerando pattern**: Spikes within a burst typically increases then decreases in frequency
        d. Bursts **develop gradually** rather than all-or-none
    3. RE neurons have been classified in **different morphological classes**. However, no clear differences in electrophysiological properties have been correlated with this morphological diversity.
- Hypothesis: **Higher distal densities** of **T-type calcium channels** are necessary to reproduce burst features mentioned above
- Experimental Methods:
    1. Slices:
        a. Young rats (P8-P15)
        b. **Somatosensory sector** of RE nucleus
    2. *In vivo* recordings:
        a. Adult cats; unanesthetized, chronically implanted for extracellular recordings and **urethane**-anesthetized for intracellular recordings
        b. **Somatosensory** and more rostral sectors of RE nucleus
- Modeling procedures:
    1. Morphology of intact cell:
        a. Stained with **biocytin** under **100X** objective
        b. Serial sections of **80 um** with a computer tracing system
    2. Passive fitting:
        a. Fitted response to **short voltage pulse** with **double exponential**
        b. Also did a **simplex algorithm** minimizing **squared error**; repeated from different initial values of parameters
    3. Model reduction:
        a. Collapse dendritic compartments into equivalent cylinders based on the conservation of **axial resistance** (produces correct electrotonic attenuation)
        b. Will need a dendritic correction factor (**corrD**) to rescale conductance and capacitance values
- Current Model:

1. Intact cell model:
    a. Geometry:
        - 4 primary dendrites; total length = **3785 µm**
        - Total membrane area = **15,115.5 µm^2**
        - Soma membrane area = **1760 µm^2**
        - Soma diameter **~ 20-25 µm**
        - **80** or **230** compartments (give identical results)
    b. Passive fitting:
        - ~**150-450** iterations were required to converge to a minimum error
        - Values from different initial conditions were considered uniform
        - Only fitted to voltage-clamp recordings and not current-clamp recordings because the model included only a subset of currents present.
    c. Mechanisms included:
        - T-type calcium currents
        - Calcium diffusion/extrusion
        - HH-type currents
2. Dissociated cell model:
    a. Truncate all but most proximal dendrites, leaving **8** compartments
    b. Total membrane area = **3639 µm^2**
3. **3**-compartment model:
    a. Geometry:

	L (µm)	Diam (µm)
soma	34.546	14.075
proximal dendrite	103.24	5.56
distal dendrite	190.69	3.06

4. Single-compartment model:
    a. Truncate all but most proximal dendrites, leaving **8** compartments

- Results:
    1. *In vivo* **extracellular** recordings:
        a. In **awake** animals, RE neurons exhibit tonic firing (**20-60 Hz**)
        b. During **slow wave sleep**, RE neurons exhibit burst firing (up to **400 Hz**)
    2. *In vivo* **intracellular** recordings:
        a. At depolarized membrane potentials (e.g., **-68 mV**), a current pulse evokes tonic discharges
        b. At hyperpolarized membrane potentials (e.g., **-95 mV**), a current pulse evokes bursts with characteristic **accelerando-decelerando pattern** with a longer first interval

    c. *In vivo*, bursts can be activated gradually in a **graded** fashion; in contrast, *in vitro* recordings exhibit all-or-none behavior
3. Presumed *in vivo* **dendritic recording** of an RE cell:
    a. Depolarizing current pulses evoke a **broad spike** (presumable a calcium spike) with **smaller-amplitude spikes** (presumably electrotonically attenuated sodium spikes from the soma)
    b. Sodium spikes **increase in frequency** during the **rising phase** of the calcium spike and **decrease in frequency** during the **decaying phase** of the calcium spike
4. Morphology
    a. The dendritic arborizations tended to spread in planes parallel to the long axis of the **nucleus**
5. *In vitro* voltage clamp in **dissociated** vs. **intact** cells:
    a. Steady-state inactivation protocol in dissociated cells showed a low peak T-current amplitude of **~130 pA**,
    b. Voltage step protocols in dissociated cells showed **much lower amplitude** (**~150 pA**) than in intact cells (**~2 nA**). In comparison, dissociated TC cells show a higher amplitude.
    c. Voltage step protocols in dissociated cells showed **faster kinetics** than in intact cells
6. Passive fitting:
    a. Input resistance (**Rin**) = 141-146 MΩ
    b. Axial resistivity (**Ra**) = 200~300 Ω-cm (higher than most cells)
    c. Membrane time constant (**taum**) = 20 msec
    d. Total capacitance (**Cm**) = **151 pF**
    e. A high series resistance (**20-50 MΩ**) was needed for the best fit
7. Dissociated cell model
    a. Had an **input capacitance** slightly larger than that measured in dissociated RE cells
    b. Compared with TC neurons, T currents in this model has relatively **slow inactivation**, a nearly **voltage-independent** rate of inactivation and a **more depolarized** active voltage range.
    c. Using Result **#5a**, assuming a uniform density, the estimated T-current density was **0.045 mS/cm^2**
    d. Using Result **#5a**, assuming that T-currents were only located in the soma, the estimated T-current density was **0.1 mS/cm^2**
    e. **Space clamp** was almost perfect in the dissociated cell model, showing that the T current kinetic parameters estimated were reliable
8. Intact cell model:
    a. Both a uniform T-current density of **0.045 mS/cm^2** in the dendrites and a somatic density of **0.1 mS/cm^2** gave rise to a total current of 500 pA, but **failed to generate bursts** under current clamp

   b. The threshold for producing bursts was ~**0.3 mS/cm^2** for uniform density and ~**3 mS/cm^2** for somatic density
   c. With a higher density of ~**0.5 mS/cm^2** in the distal dendrites (and **0.045 mS/cm^2** elsewhere), bursts could be generated under current clamp. A broad calcium spike appeared in the distal
   d. A higher distal T-current density also reproduced the higher amplitude of the current seen in the soma (Result **#5b**)

9. Dendritically generated bursts:
   a. After a hyperpolarizing impulse, a **broad calcium spike** appeared in distal dendrites, which elicits **sodium spikes** in the soma.
   b. In some dendrites, a broad spike with riding small-amplitude spikes could be observed with the accelerando-decelerando pattern, all consistent with Result **#3a** & **#3b**.
   c. Throughout burst, membrane potential remain high in dendrite, feeding soma with current. Note: dendritic sodium currents are *not* included.
   d. Experimentally, it was extremely difficult to obtain **voltage-clamp control** in intact RE cells. This was reproduced in the model by showing a voltage difference between soma and distal dendrites as high as **~80 mV** during transients (cf. **60 mV** using constant field equations), more evidence of higher T-current density in distal dendrites
   e. As a consequence of poor voltage clamp, dendritic currents occurred and added a **slower component** to the current decay seen in the soma, causing the slower kinetics as observed in Result **#5c**
10. Creating the graded burst response
    a. In current-clamp mode, the intact cell model normally generated all-or-none burst responses
    b. By adding **sustained depolarizing currents**, a graded burst response was observed. Therefore, a possible explanation of Result **#2c** is that RE cells are continuously bombarded by **excitatory synaptic inputs** *in vivo* but not *in vitro*.
    c. For even higher intensities of sustained depolarizing currents, it became increasingly difficult to generate a burst
11. **3-compartment** model:
    a. Passive fits were just as good as intact cell model
    b. Voltage-clamp traces were similar to those in intact cell model with the same T-current density
    c. As in intact cell model, current-clamp behavior did not show bursting activity unless T-current density was increased by an order of magnitude
    d. As in intact cell model, high distal density showed similar bursting behavior, though the **burst threshold** was slightly different.
    e. The burst profile is similar to the intact cell model, with the accelerando-decelerando pattern

f. **Graded burst responses** could be simulated using similar current densities to the intact cell model

12. **Single-compartment** model:
    a. Capacitive transients were fitted well but not as good as other models
    b. Bursts are relatively broad but *do not have* the typical accelerando-decelerando pattern.
    c. Graded burst responses could *not* be simulated at all

- Discussions:
    1. The hypothesis is true. Calcium currents in dendrites are essential for generating the bursting responses of RE cells.
    2. Result **#5b** implies that there is less T-current in the dendrites of TC cells compared with RE cells.
    3. Positive shift of T-current activation curve, distality of T-currents and higher axial resistance all contribute to a **higher burst threshold** in RE cells relative to TC cells.
    4. The fact that all the burst properties could be reproduced in a **3**-compartment model suggests that morphology has little influence on electrophysiological properties recorded in the soma, explaining Background Fact **#3**. Instead, the morphological diversity might be related to the **organization of synaptic inputs**.
    5. The presence of additional depolarizing currents abolishes bursts
       => Wake -> Tonic input -> no bursts -> no spindles
       Sleep -> Phasic input -> bursts -> spindles
    6. Future directions:
        a. A **calcium-dependent potassium current**, **I_{KCa}**, was shown to underlie **repetitive bursting** in RE cells. This current was not included and more data are needed to investigate a somatic versus dendritic localization for ***$I_{K[Ca]}$***.
        b. **GABAergic collaterals** between RE neurons have been identified. The possibility of rebound bursts in GABAergically-connected RE cells could be tested by stimulating RE cells after **blockade of excitatory synaptic transmission**
        c. In cats, the RE nucleus is characterized by the presence of **dendro-dendritic** GABAergic synapses. This could generate synchronized oscillations.

10/17/2016~10/21/2016

<u>**Rivanna/Parallelization (cont'd)**</u>

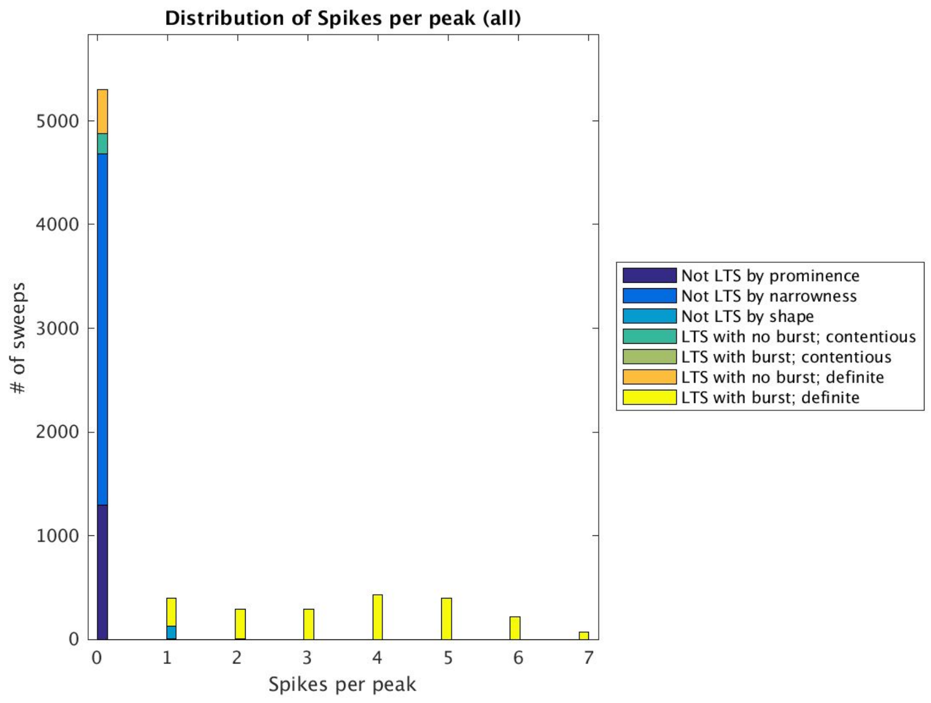
- Updates to **FindIndtoFit.m**:
  - Fixed FindIndToFit.m so that the special cases folder do not have to exist if fitmode == 1
- Updates to **dclampDataExtractor.m**:
  - Removed close(h) again from **find_LTS.m** and included **close all** inside the parfor loops
  - Made functionsdirectory, homedirectory **dependent on existence** (so that you don't need to change the code manually when uploading onto Rivanna)
- Attempt #5 (**dclampDataExtractor5~8.slurm**):
  - Modified **dclampDataExtractor.m** to include **close all** inside the parfor loop
  - Result:
    1. For **dclampDataExtractor5.slurm**, still got this error:

       ```
       slurmstepd: Exceeded job memory limit at some point.
       ```
    2. For **dclampDataExtractor6~8.slurm**, Successful completion of sweep analyses with no error!
    3. However, there remains a warning that "**ProbDistUnivParam** will be removed in a future release. Use the pdf method of an object returned by **fitdist** or **makedist** instead."
  - Conclusion:
    1. Memory issue might depend on the node. Will still be helpful to improve memory usage (maybe use **clear all** inside parfor loops too?)
    2. Need to fix **ProbDistUnivParam.m**

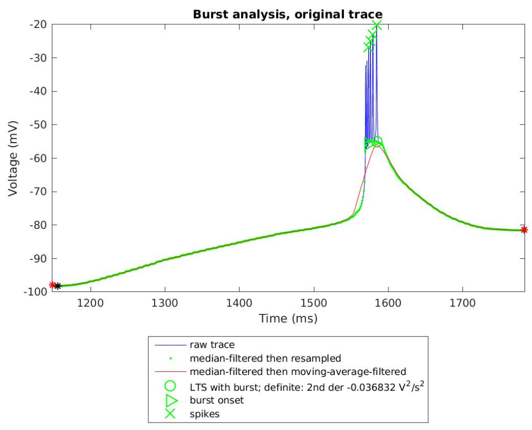
10/17/2016~10/21/2016

<u>Dynamic clamp data analysis (cont'd)</u>

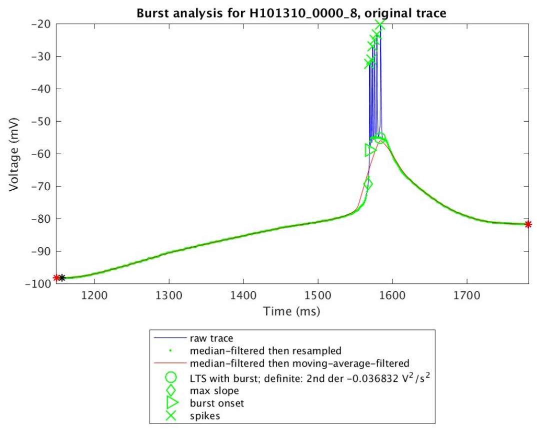
- Updates to **test_sweep.m**:
    - Added iterations to test parfor
    - Added infolder, sweeps
- Updates to **find_LTS.m**:
    - Changed **actual spike threshold** to **5 mV above LTS peak**

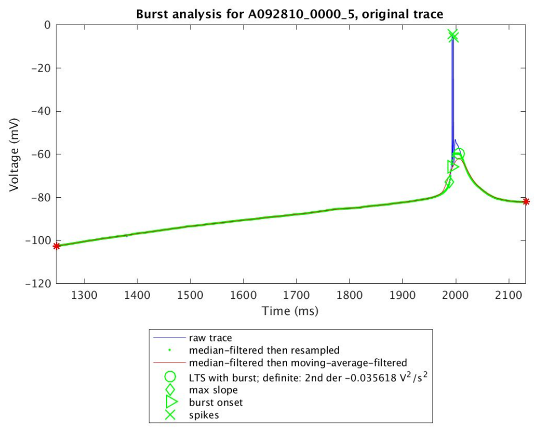


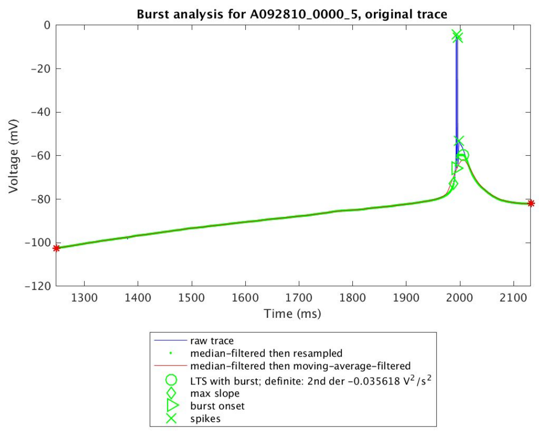
Before:



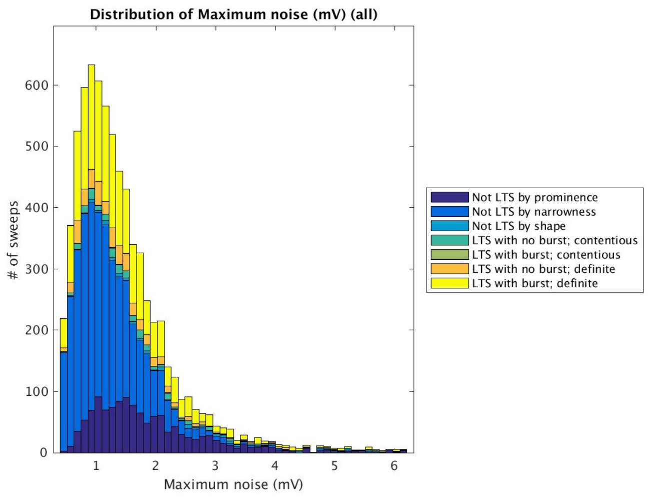
After:





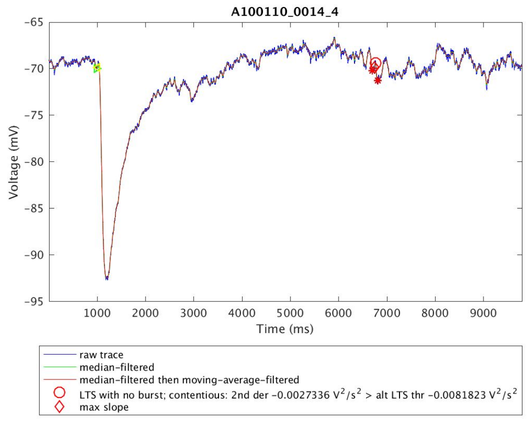


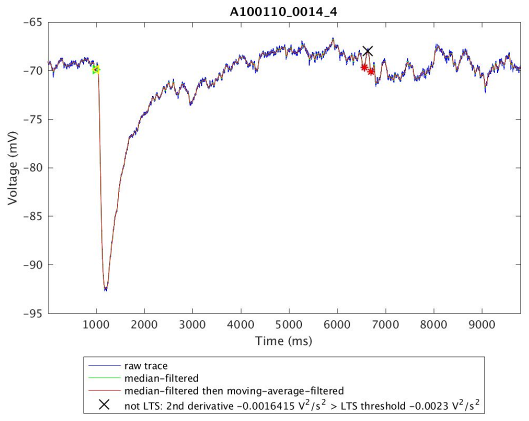
- Changed **maxnoise** so that it's **4*standard deviation** of the baseline data instead of 2*standard deviation



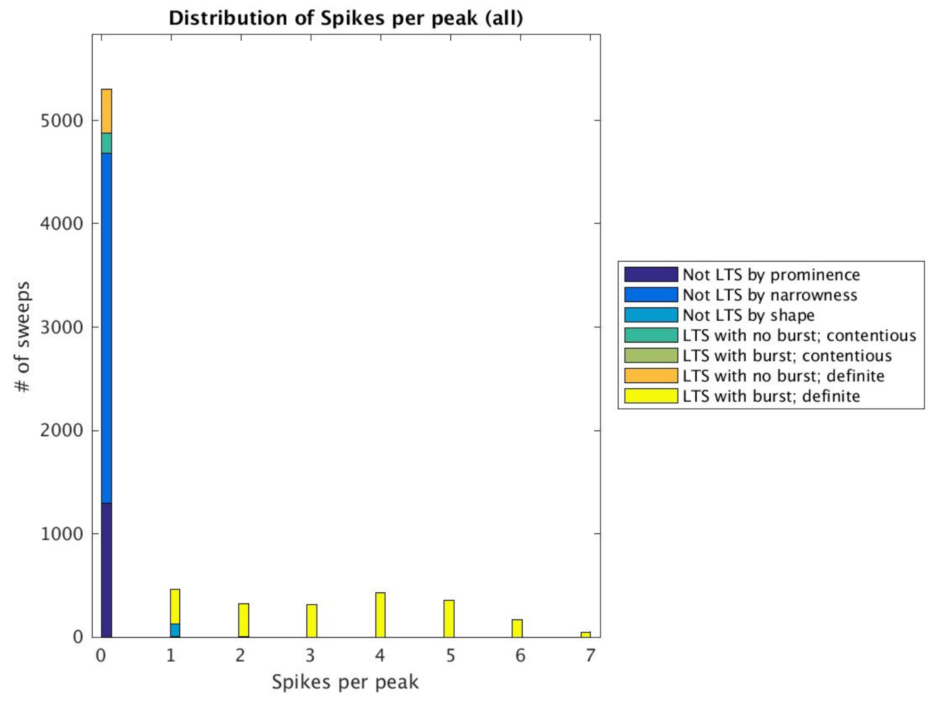
Before:

After:

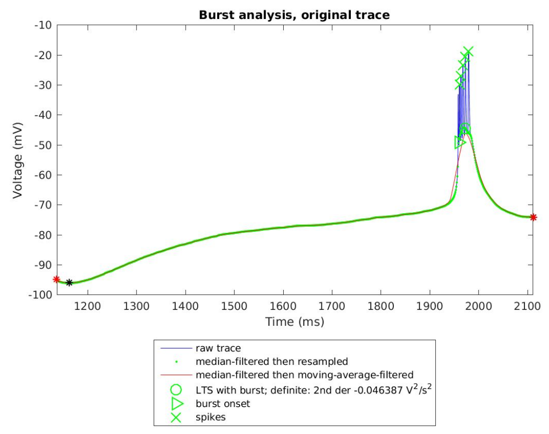




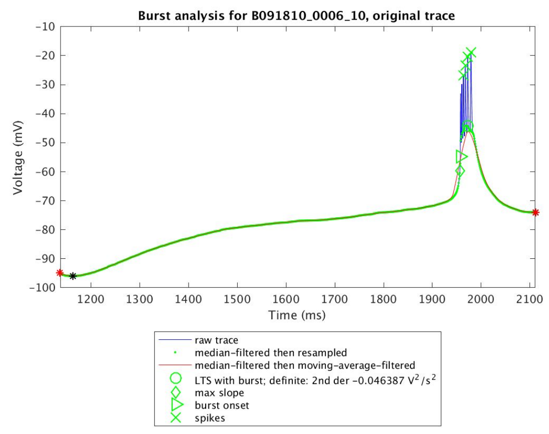
- Changed **actual spike threshold** again to **15 mV above LTS peak** and changed initial spike threshold to **-45 mV**



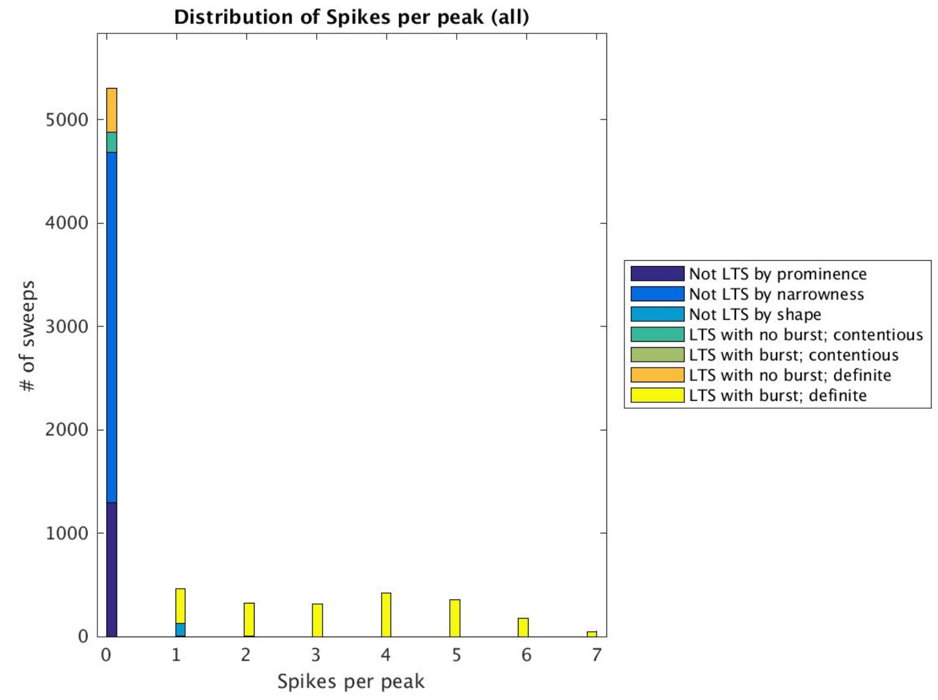
Before:



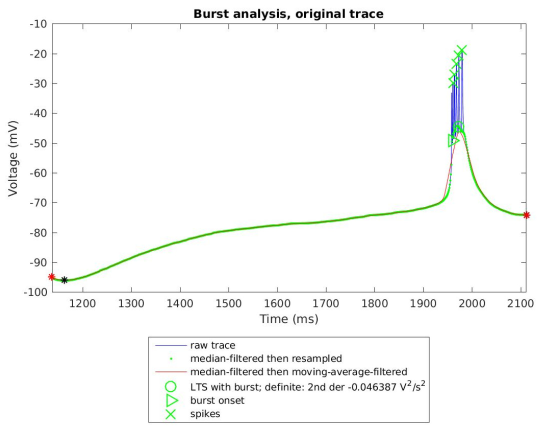
After:



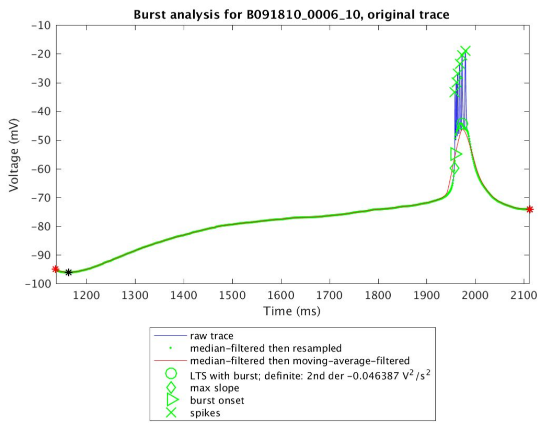
- Changed **actual spike threshold** again to **10 mV above LTS peak**



Before:

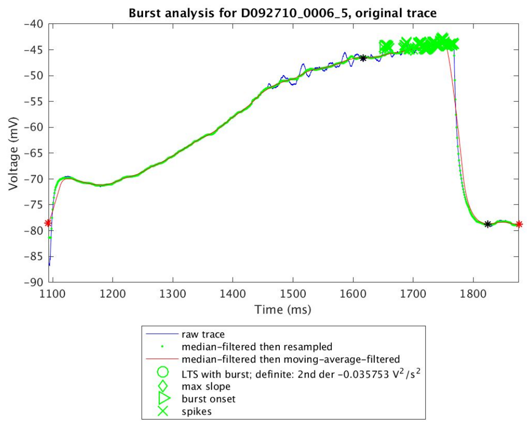


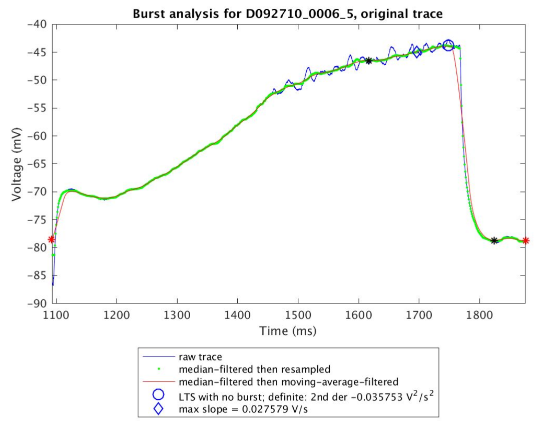
After:



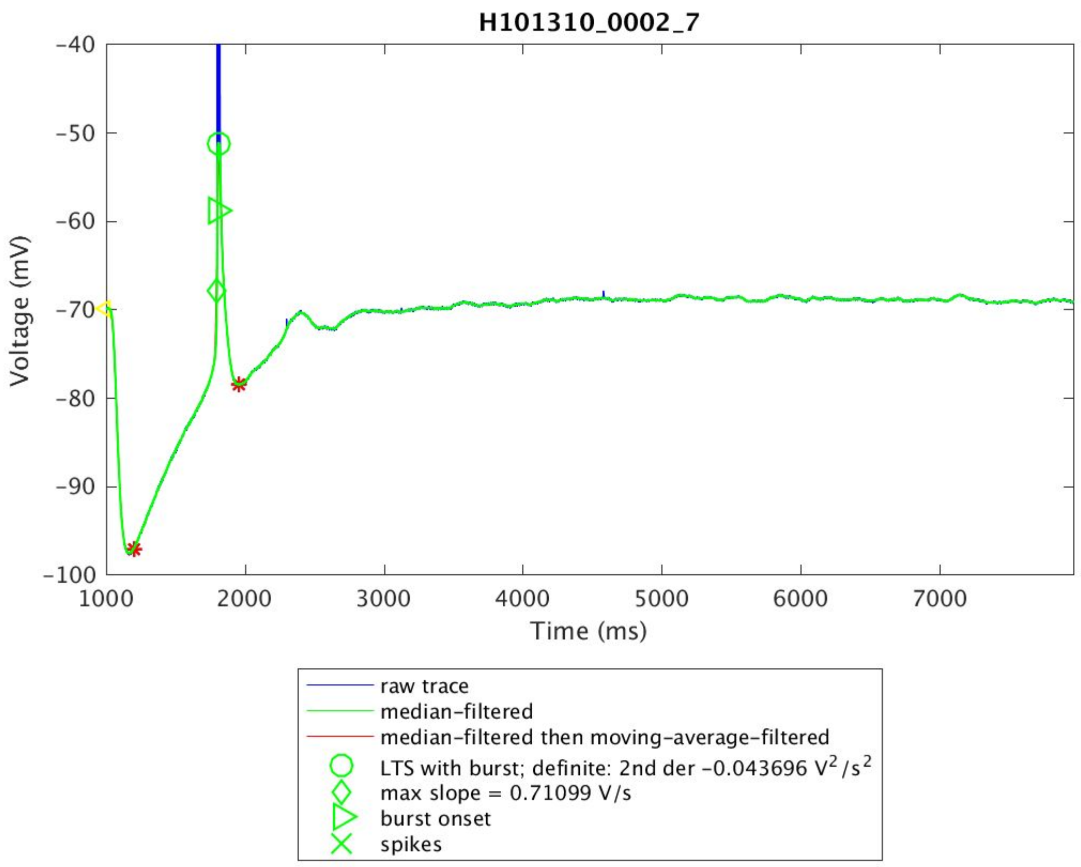
- ○ Now prints "spontaneous spike or noise" for peaks of class 1 & 2 with spikes
- ○ Fixed issue with not updating detected spikes (see D092710_0006_5)

Before: After:





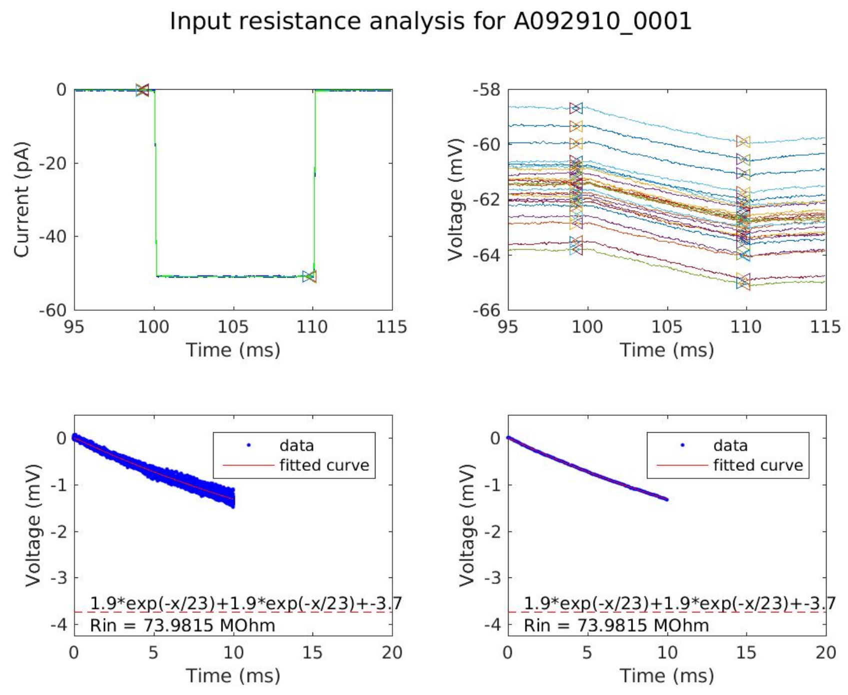
- ○ **max slope** now displays a value when plotted
- ○ Changed the y axis of **vtraces_scaled** and plotted median-filtered trace with **thicker line**

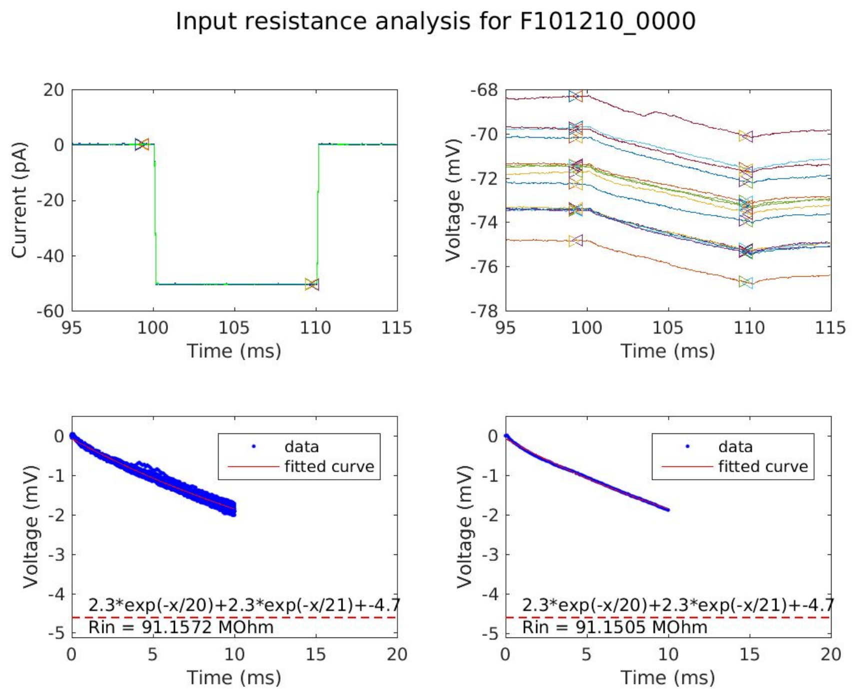


10/26/2016~10/31/2016

Passive fitting (cont'd)

- Pooling all data together and averaging over traces first didn't seem to change the fitting much





Plan for next week

- Patching:
  - Set up Paula’s rig
- EEG:
  - Practice another implant
- Fitting:
  - Continue with double exponential fit
  - Simplex method
- Johnston & Wu:
  - Do Ch 4 Problems
- Data Analysis w/ Brian:
  - Spike features and correlation diagrams
  - Principal component analyses
- SWD detection (w/ Vignesh):
  - Automate the entire process given an input Excel file
  - Change the output into a text file that contains the restricted data for each abf file

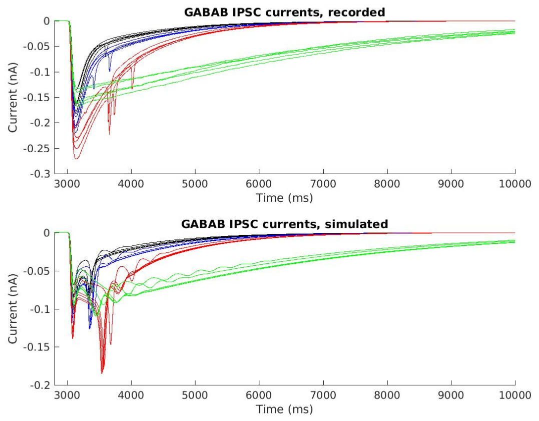
10/3/2016~10/17/2016

Model/Fitting

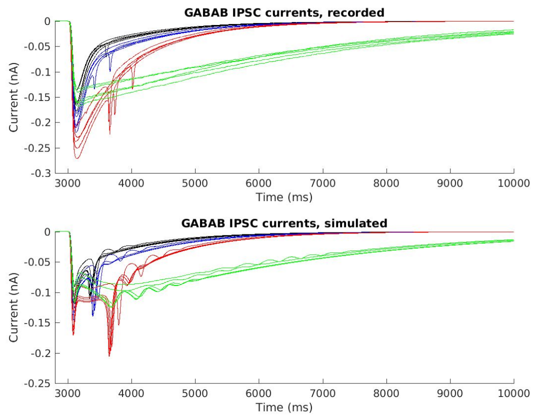
- Updates to **singleneuron4compgabab.hoc**:
  - Multiplied **ghbar** & **pcabar** in the proximal dendrite with the dendritic correction factor
  - Inserted an IClamp called **stim0** for current pulse; added stim0i to record stim0.i
  - Made **tstop** & **stim0.amp** arguments (and removed holdcurrent) to be passed to **sim()**
- Updates to **run_neuron_once_4compgabab.m**:
  - Changed the way NEURON is run so that each sweep can be sent to a different worker, using a **here document** to attach customized commands as you open singleneuron4compgabab.hoc
  - Moved **outparams.lts_to_swp_errratio** to the calculation of total error (instead of total LTS error)
  - Changed the way error calculations are organized; added the function **compute_and_compare_statistics**
  - Correct the 95 % confidence intervals in bar plots
  - Renamed swpreg as **fitreg** (fit region)
  - Renamed figure handles so that they are now all in a structure **hfig** that is passed to and from functions
  - **outparams.currpulse(k)** is now already in **nA**
  - Added cprflag, findltsflag, ltsburststatsflag, ltserrorflag so that these could be suppressed during optimization
  - Updated outputs for find_IPSC_peak & find_LTS
  - Fixed outparams.fitreg to fitreg inside parfor loop
  - Changed from root mean-squared error to **mean-squared error**
- Updates to **optimizer_4compgabab.m**:
  - Reorganized code; moved update_sliderposition to optimizergui_4compgabab.m; **handles** is no longer passed to optimizer_4compgabab.m
  - **xlimits** now uses outparams.fitreg instead of outparams.swpedges
  - **Renumbered** figures systematically to prevent overlap
  - **Renamed figure handles** so that they are now all in a structure **hfig** that is passed to and from functions. And so all figures are made visible only if
  - Plotted current pulse electrode current and responses
  - Wrote **setfieldszero** & **restorefields** functions to set flags to zero before optimization and restore afterwards
- Updates to **optimizergui_4compgabab.m**:
  - Added **current pulse response**
  - Started to reorganize code
  - Renamed figure handles so that they are now all in a structure **hfig** that is passed to and from functions

- Changed E_rev of GABAB from **-105 mV** to **-115 mV**

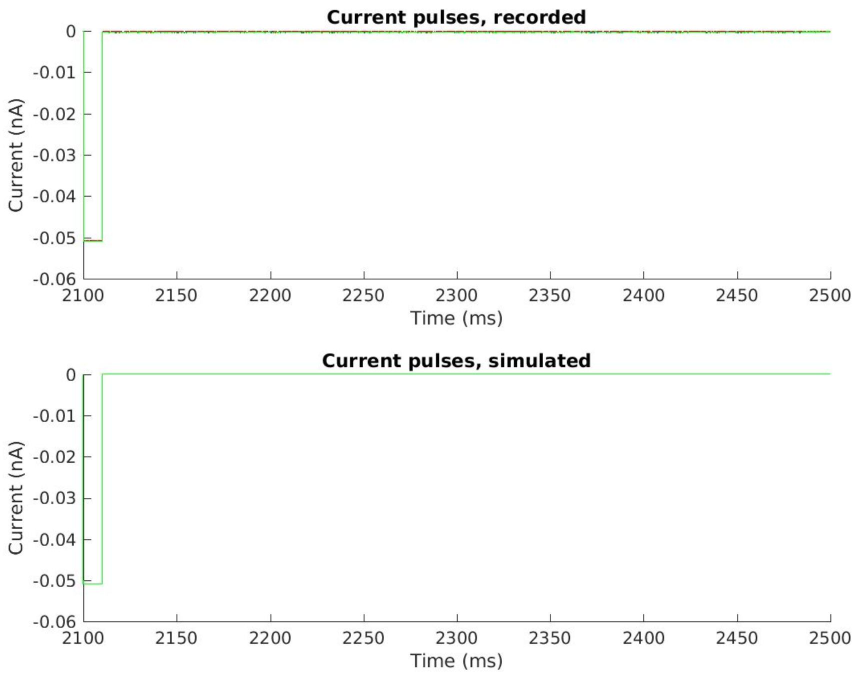
Before:



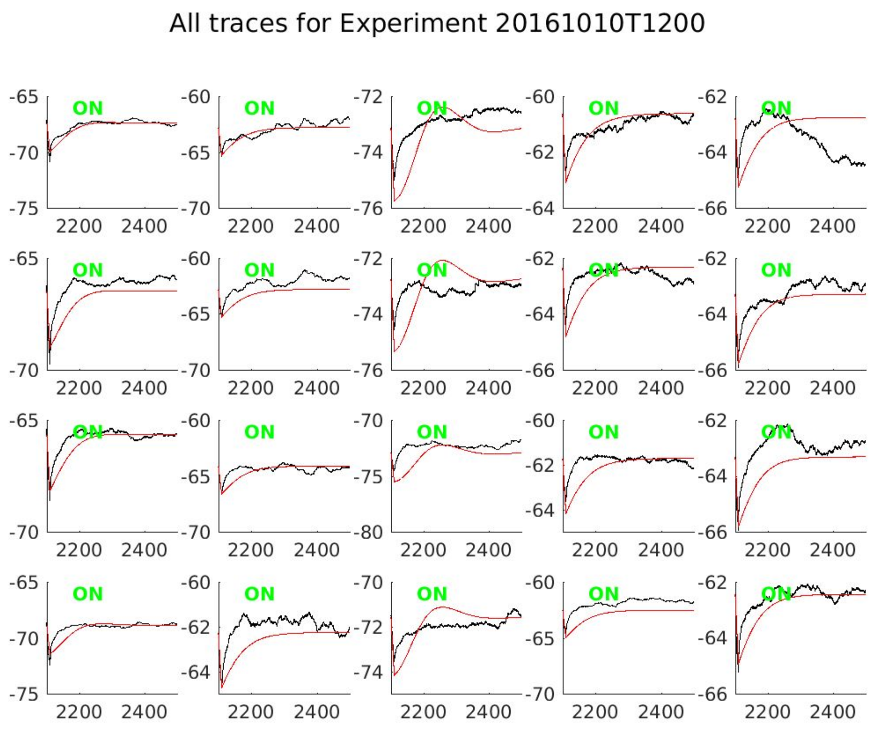
After:



- Current pulse response:
    - Read in **time of current pulse**. Align and patch data so the current pulse lies in **2100-2110 ms**

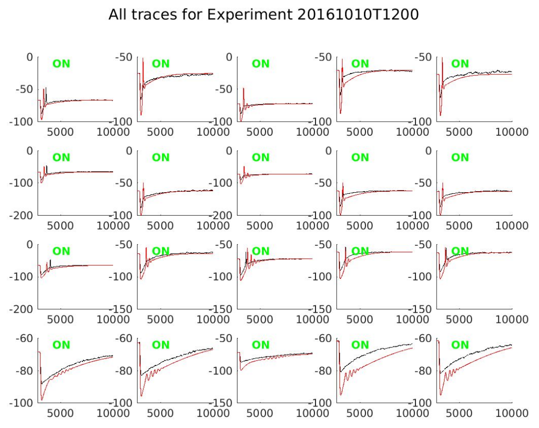


- Read in **holding potential for passive fit**. Simulate without HH.

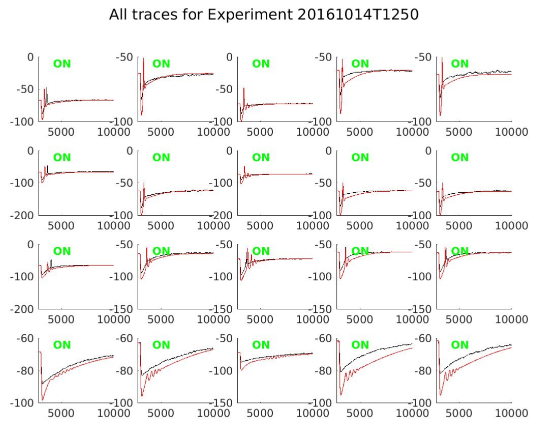


- Tried setting **secondorder = 2** in NEURON (Crank-Nicholson method instead of the default Euler backward integration). But the traces didn't change much, so returned to **secondorder = 0**.

secondorder == 0



secondorder == 2



- Email from John 20161010:

Hi Adam and Mark,

Regarding the fit of the voltage trace to the short current pulse, I have the following thoughts.

- this was meant to be a first step in the fitting process, as it will be very difficult to settle on a unique solution when there are so many unconstrained variables. So we should use the response to a short current pulse to estimate the input conductance and rho, somatic/dendritic conductance ratio. This should help us set the resting state of the neuron in terms of dendrite surface area.

- for this to work well will require averaging, as the voltage responses are rather brief. I propose that we average all the current pulse responses across all conditions for a give RMP (-65, -70, etc), as the response to this pulse will be unaffected by the pharmacological condition as there is zero GABAb conductance during the test pulse.

-the idea would be to fit this averaged trace with the matlab/simplex method to obtain good values for resting state. Probably we should decide how do to this, but one possibility would be to set gT and gH to zero, as the duration of the voltage response may be too brief to have much affect on numbers of voltage gated channels that are opened.

- then, once you have the resting state of each cell, fix those parameters, and then only allow the other ones to vary.

Your thoughts?

Best,
John

- Email to John 20161011 with his responses in green:

Dear John and Mark,

Sorry this is long, but I've underlined my questions in case you want to skip my rambling.

1. I agree that we should only use the current pulse response to fit selected parameters, but I'm not sure which ones.

I know in Johnston and Wu (I've been studying this but has only just begun Chapter 4), they discussed the estimation of **R_in** (input resistance), **rho** (somatic/dendritic conductance ratio), **L** (electrotonic length) and **tau_m** (membrane time constant) from a *current step response* by fitting a **double exponential**. But these are not parameters directly used in our simulations. Do you think we would be able to calculate all the passive membrane parameters from these four values? Maybe it's possible as long as we make assumptions about the **geometry**. I have yet to try out Problem 4.10.14(c). And making such assumptions is basically what we've been doing all along since we do not have any sort of geometrical data to begin with (would it have been possible to do these dynamic clamp experiments along with **biocytin fills**?)

> John: Not necessarily. However, we can calculate L and rho, and even R_in quite easily for any geometric model, and so the parameters we use for the 4 compartment model should have overall
> Good point. We are trying to see if we can come up with a reasonably good estimate that fits well with the data we have. I think Christine did do some biocytin fills, but they were not recovered and traced, so that opportunity is lost now.

Here is what's been done in Destexhe et al 1998: The following parameters were estimated by fitting the *voltage step response* of the **detailed 208-compartment model** to the recording of a *single* TC neuron (you probably did the recording): leak conductance (**g_pas**), leak reversal potential (**e_pas**), axial resistivity (**Ra**), specific membrane capacitance (**cm**) and the electrode series resistance (**Rs**). Then the dendritic correction factor (**corrD**) was estimated for the **3-compartment model** by fitting its voltage step response to the *same data* (with improved accuracy, of course).

So these are all passive parameters that Christine was trying to estimate with the fits, with the exception of **Rs**. Furthermore, because we now have a 4-compartment model, there is an extra geometrical parameter, the relative length of the most distal dendrite (**distdendpercent**), that needs to be fitted. I'd like to ask about each of these in turn:

(a) Specific membrane capacitance (**cm**): My understanding is that for a parallel capacitor, **cm** = C/A = epsilon/d, where d is the **membrane thickness** and epsilon is the dielectric constant, which would depend on the **membrane composition.**

Should **cm** vary among compartments? Currently, it doesn't. Well, membrane thickness should be fairly constant across all lipid bilayers (my impression is that phospholipids in the human body almost always contain fatty acid tails of length 16~20 carbons). And membrane composition should be fairly constant within a cell, since the cell membrane is really fluid. However, if there is a *higher* concentration of **lipid rafts** in the distal dendrite versus the soma, maybe there could be a difference, as lipid rafts look thicker and with more spacing probably has a lower dielectric constant (cholesterol and sphingolipids create lots of kinks), so would probably have a smaller **cm**. But then I haven't found any evidence of that, and these rafts are probably a small proportion of the entire cell membrane anyway.

Should **cm** vary from cell to cell? On the other hand, maybe the membrane composition vary a lot from cell to cell. For example, maybe lipid rafts are more numerous in certain neurons than others, which might offer us a rationale for making **cm** *cell-dependent* in our model. However, according to Gentet et al, 2000, **cm** was about the same across all classes of neurons studied. Therefore, maybe it would be a valid assumption to make **cm** a *constant* in our model, so that other parameters could potentially be estimated more accurately.

> I think we can just use the cm estimated by Destexhe. It would add enormous complexity to try to vary this by compartment or per cell and we have zero data on this point.

(b) Leak reversal potential (**e_pas**): My understanding is that this is a weighted average of the reversal potentials of all *voltage-independent* channels, of which a potassium channel probably contributes the most.

Should **e_pas** vary among compartments? Currently, it doesn't. I cannot find evidence that the ratio of leak potassium channels over leak sodium channels vary between soma and dendrites. And the K+ concentration gradient is probably the same cross the span of a neuron (~200 um according to Figure 1B of Destexhe et al 1998).

> No, again too much complexity would be introduced by doing this. We don't have enough constraints to allow this as a variable.

Should **e_pas** vary from cell to cell? The ratio of leak potassium channels over leak sodium channels might indeed vary from cell to cell. The internal K+ concentration and the local extracellular K+ concentration might also vary, altering the K+ concentration gradient and thus the reversal potential. So I think this is likely. But then, we aren't varying the E_rev of GABA_B channels, which also depends on the K+ gradient, so why should we treat e_pas differently? Therefore, maybe we should also make **e_pas** a *constant* in our model, and just vary **g_pas** instead.

> Yes, because this is the main thing that determines the resting membrane potential.

Should **e_pas** vary from trial to trial? Maybe the K+ gradient is stochastic? But again, we don't do that for E_rev of GABA_B channels.

> Our definition of a stable recording is one in which the resting membrane potential, and input resistance do not change very much. So, operationally, we should not vary either e_pas or g_pas within one recording, i.e. from trial to trial.

(c) Axial resistivity (**Ra**): My understanding is that this has to do with cytoplasmic composition, which includes the cytosolic composition (increased concentration of ions should *decrease* **Ra**) and the presence of organelles and other macromolecules (the local decrease in cross-sectional area would *effectively increase* local **Ra**).

Should **Ra** vary among compartments? Currently, it doesn't. Of course, the cytosol is continuous, and organelles move around freely. However, there is probably a different concentration of organelles in the soma versus dendrites. As Bekkers 2011 shows, one can increase local **Ra** just by stretching a section of a dendrite and altering the cytoskeletal composition. Furthermore, as we are *fixing the relative geometry* of compartments, maybe we should also allow Ra to vary just to account for the variation in cross sectional area.

Should **Ra** vary from cell to cell? Probably yes, as cytoplasmic composition probably depends a lot on the type of cell, the healthiness of the cell, etc. For instance, in Table 3 of Roth & Hauser 2011, cell #3 and cell #4 had non-overlapping ranges of **Ra**, even after accounting for systematic errors. Furthermore, again maybe we should also allow Ra to vary just to account for the variation in cross sectional area.

Should **Ra** vary from trial to trial? Probably not, as cytoplasmic composition probably doesn't vary that quickly.

Does **Ra** affect input resistance? If you look at Figure 4 of Mainen et al 1996, input resistance increases as **Ra** increases, but why? I thought R_input = Rm + Rs.

> We should keep Ra fixed across all conditions, for the reasons stated above.

(d) Leak conductance (**g_pas**): This reflects the leak channels concentration and the proportion of leak channels that are open.

Should **g_pas** vary among compartments? Currently, it doesn't. However, the leak channel distribution might be different between soma and dendrites, just like what's proposed for T channels.

Should **g_pas** vary from cell to cell? Probably yes. The leak channel's surface expression might be modulated by other factors, which may reflect the class/state of each cell. A wide range of membrane resistance values is often reported in literature, as in Major et al 1994.

Should **g_pas** vary from trial to trial? Would any factors that modulate leak channel expression do so with a time constant within 2 minutes (approximate span of the 5 repeating sweeps)?

> Same as for e_pas. we should estimate this one first based on the fit to the fast current pulse, and also at "rest". This is determined by where there was zero current injected. Did Christine keep those values in a database somewhere? They are not necessarily in the clampex files.

(e) Dendritic correction factor (**corrD**): Since the geometry is simplified to cylinders, this is a correction factor to account for the change in effective surface area. This is basically what's reflecting the somatic/dendritic conductance ratio (**rho**) in our model

Should **corrD** vary from cell to cell? Probably yes, because the geometry usually varies from cell to cell.

> Yes.

Should **corrD** vary from trial to trial? Probably not, as TC neurons are mostly immobile.

> No.

(f) Electrode series resistance (**Rs**): This was not even estimated. Did Christine record the access resistance values when she did the experiments? Or is there a typical average series resistance we can set as a *constant*?

> I would use typical. It should not matter much. Maybe set it at 10 Mohm

(g) **distdendpercent**: My understanding is that Christine included this to vary the T channel distribution further. However, wouldn't this be achieved as long as there are two nodes instead of one in what was the distal dendrite of the 3-compartment model? The only difference is that increasing **distdendpercent** would move the dividing line distally. However, since membrane conductance is defined in terms of unit area, and since NEURON only computes at the midpoints (there is currently only 1 node per section), wouldn't this actually increase the proportion of T channels in the more proximal section, thus effectively moving T channels more proximally? Therefore, my question is, do we still want this parameter or maybe just set it to some arbitrary value, e.g., **0.5**?

> You can implement this in any way that you want. I agree that it should be the same. However, make sure that the code doesn't use nseg, as this would allow calculation in subcompartments. I thought that is what she implemented with distdendpercent, but if not, then any way you want to do this will be fine. This is only a factor when fitting the GABAB responses, not the initial characterization of resting properties.

2. As the histogram of actual Vhold shows, the holding potential was almost a continuum between -60 mV and -70 mV. Wouldn't it be unnatural to group them by holding potentials?

> I think it is ok. it is simply a form of binning, and it does help, because binning allows you to signal average, which reduces the S/N.

However, I do agree that all sweeps of all pharmacological conditions should be pooled together for a single neuron. Even though you can't average the sweeps, there are still at least three possible ways of averaging out stochasticity:

(a) Compile a **total sum of squares error** across all sweeps just as Christine was doing, and any poor fits should be averaged out. (By the way, is there a point of taking the square root of the least squares as Christine did? Leaving the cost function in sum of squares form probably allows the optimization to converge faster, as the gradient would be much more easily calculated)

> I'm happy with leaving it as ss, but if we do the sequential approach of just fitting the region of the trace around the short pulse response first, then averaging is fast, and then there is only one resultant trace to fit, which will speed things considerably.

(b) Compile a **total sum of squares error** across all sweeps *weighted* by some **baseline noise**, which could be equal to **4*standard deviation** (or some rms measure) of the values over a certain baseline region.

> Not sure

(c) Fit each sweep individually, then apply some **maximum likelihood estimate** to figure out the best estimate and ranges for each parameter. This method also offers a measure of each parameter's **sloppiness**, and was was basically what they did in Figure 6 of Nogaret et al 2016.

Which method do you think would be best?

> I like a, again restricted to just a portion of the trace, and then on only the averaged trace. The Nogaret method might be good for the next step, which is the fitting to the more complicated GABAb/rebound LTS part.

3. My intuition would be to leave all channels as is while doing the passive fit, since we **didn't apply any pharmacological blockade** during the current pulse. Furthermore, the simulations I showed on Friday showed that **T currents** were indeed activated in some cases (especially when the holding potential was hyperpolarized). However, if we choose to leave the active channels as is, we would then have to arbitrarily set the value of the parameters. Unless we **iteratively** estimate the passive parameters, using that to estimate the active parameters, then using the result to estimate passive parameters again, and so on. Would this be a good idea?

> That is the catch 22. The rebound LTSs you saw with the fits reflect the initial conditions you have in the model, which may not be realistic. We are trying to estimate resting conditions, and don't know how much ih/it/ etc contribute to this. I would start with leaving them out. Then once we go back and fit It etc to get the overall response, then go back and see how much this influences our estimates of resting g and e.

In Destexhe et al 1998, passive fitting was performed with and without T currents. The estimated parameter values were similar "except for the **leak reversal potential** that needed to be readjusted to compensate for the window current." Perhaps we should do the same for all channels? (Remove one channel at a time and determine any difference in parameter estimates)

4. I agree that the fitted passive parameters would be fixed when estimating the active parameters. However, we might need to do it iteratively.

> Exactly!

As for the method of optimization, I also have several questions:

(a) Is the **integration method** accurate enough? Currently, our model uses CVODE, which uses the **backward Euler method** (1st order) as the default method for forward integration of voltage values. We could potentially increase this to 2nd order accuracy by setting **second_order == 2** and use the **Crank-Nicholson method**. According to the NEURON book, The gating variables are integrated with 1st order accuracy if **derivimplicit** is used, and with 2nd order accuracy if **cnexp** is used, even though I can't figure out what methods they stand for. However, Nogaret et al 2016, with all its pretty fits, uses their own simulator integrating with the **Runge-Kutta 5 method**, which has 5th order accuracy. So maybe we should at least apply second_order == 2?

> cn-exp is Crank-Nicholson, while derivimplicit is I think first order, I think. If computational time is not an issue, then cn is fine with me.

(b) Is the **time step** small enough? Another way of improving accuracy is to decrease the time step. In fact, Nogaret et al 2016 made sure there are 100 time points for each action potential! With CVODE, the largest output file in the last simulation I did was only 609 time points for a total of 10000 ms. However, maybe this would be good enough for all purposes.

> Since It, Ih, and GABAB are all very very slow, then we don't need such a short time step, unlike if we wanted to do action potentials.

(c) Should the **initial conditions of the state variables** vary from trial to trial? Currently, the initial condition of all state variables (such as the gating variables) are the steady state value as a function of the **holding potential**, in agreement with what was used in Nogaret et al 2016. But the initial state of the cell might be altered if there were synaptic events or spontaneous spikes in the baseline region before the current pulse. Therefore, we could potentially make some of the initial conditions (e.g., m & h of T currents) vary from trial to trial. Or we could simply try taking out the traces with spontaneous spikes to see whether it makes a difference in fitting.

> This would be interesting to try, to see if you converge on the same solution with different initial conditions. This would provide a test of robustness of the method.

(d) Should **initial values of the parameters** be varied? The initial values of parameters during the search might influence the final estimation if there is a local minimum. If this were the case, a large range of estimates would be obtained when disparate sets of initial parameters were used. This would possibly help us find the global minimum more reassuredly. In Nogaret et al 2016, they somehow formulated the problem so that it's convex, so that the parameter search always converges. How do I determine if my problem is **convex** or not?

That is a good question, but unfortunately beyond me.

(e) Should we include a **control variable** in our cost function? On page 2 of Nogaret et al 2016, a term u(t) is added to the least squares error. They claim the term smoothes convergence and vanishes as the parameter search reaches a global minimum What exactly is the form of such a function?

Sorry, don't know.

(f) What **optimization method** should I choose? I haven't looked into fminsearch3 in detail, but Roth & Hauser 2011 uses a built-in algorithm of NEURON called **PRASIX**, whereas Nogaret et al 2016 uses IPOPT

I don't think it matters too much.

Sorry again for the long response. I'm in no hurry to get a reply, as I still have a lot of code cleaning to do. But let me know what you think!

Thanks, Adam

10/13/2016~10/16/2016

__Rivanna__

- Notes from workshops:
    - **economy** queue is used if only **one core** is needed
    - **serial** queue is used if only **one node** is needed (up to **20** cores available on each node)
    - **parallel** queue is used if more than one node is needed
    - Only charged (in **core-hours**) for the amount of time actually run, not the amount of time specified. Should have time resolution to at least minutes (if not seconds).
    - **sampleTop2.sh** is a shell script written by Ed Hall that can automatically print out the usage from Top into a file named **Top.out**

- Updated code so that file paths can all be set in dclampDataExtractor.m
    - Updates to **dclampDataExtractor.m**:
        - Added functionsdirectory, homedirectory and **addpath**
    - Updates to **PlotHistogramsRefineThreshold.m**:
        - Made infolder and outfolder optional arguments
    - Updates to **dclampdatalog_analyze.m**:
        - Made infolder and outfolder optional arguments
    - Updates to **FindIndToFit.m**:
        - Made infolder an optional argument

- Attempt #1:
    - Ran **dclampDataExtractor.m** with the following usage of flags:

debugflag = 0;
resavedataflag = 0;
plotRinputflag = 0;
plotIPSCoffsetoldflag = 0;
plotIPSCpeakflag = 1;
plotLTSflag = 1;
saveswpinfoflag = 1;
plothistogramsflag = 1;
plotcorrelationsflag = 0;
plotbargraphsflag = 1;
preallocateflag = 1;

    - Result: Sweep analyses stopped at **A100810_0006**. Apparently, one of the cores ran out of memory.
    - Efficiency (from **jobe -v**):

```
JobID   Start                Queue    Size(cores)  Time(hours)  Utilization  Sampled  Node(s)
2630838 2016-10-16T11:24:13 serial   20           1.37         12%          yes      udc-ba38-10f
```

    - **dclampDataExtractor1.slurm**:

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20          # how many processes I will run per node
#SBATCH --time=3:00:00                # amount of time for the whole job d-hh:mm:ss
```

```
#SBATCH --partition=serial                  # the queue/partition I will run on (economy/parallel/serial)
#SBATCH --output=dclampDataExtractor1.out
#SBATCH --error=dclampDataExtractor1.err
#SBATCH --account=netlinks                  # the account/allocation I am using
#SBATCH --mail-user=al4ng@virginia.edu      # address to mail
#SBATCH --mail-type=end                     # send mail on certain events; type can be BEGIN, END, FAIL,
REQUEUE, and ALL
#SBATCH --job-name=dclampDataExtractor1              # job arrays should be named

module load matlab
bash sampleTop2.sh al4ng MATLAB 5 &

# Run matlab
matlab -nodisplay -nosplash \
-r "parpool('local', 19); dclampDataExtractor; exit;"
```

  - **dclampDataExtractor1.err**:

```
{Error using parallel_function (line 604)
All workers aborted during execution of the parfor loop.

Error in dclampDataExtractor (line 809)
                                        parfor swp = 1:nswps
                      % FOR each sweep
}
{The client lost connection to worker 11. This might be due to network problems,
or the interactive communicating job might have errored.
}
slurmstepd: Exceeded job memory limit at some point.
```

  - Excerpt from **dclampDataExtractor1.out**:

```
loading .mat files for the set A100810_0006 ...
Analyzing input resistance ...
Finding IPSC offsets (obsolete) ...
Finding and plotting IPSC peaks ...
Finding and plotting LTSs ...
[Warning: A worker aborted during execution of the parfor loop. The parfor loop
will now run again on the remaining workers.]
[> In parallel_function (line 596)
  In dclampDataExtractor (line 809)]
```

- Attempt #2:
  - Changed the number of workers to **15**.
  - Result: Sweep analyses stopped at **A092110_0005**. Apparently, one of the cores ran out of memory.
  - Efficiency (from **jobe -v**):

```
JobID     Start              Queue       Size(cores)   Time(hours)   Utilization   Sampled          Node(s)
2630891   2016-10-16T14:34:23 serial        16            0.28          56%        no (too short)    udc-ba38-4e
```

  - **dclampDataExtractor2.slurm**:

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16                # how many processes I will run per node; using 20 sometimes eat out
memory
```

```
#SBATCH --time=3:00:00                      # amount of time for the whole job d-hh:mm:ss
#SBATCH --partition=serial                  # the queue/partition I will run on (economy/parallel/serial)
#SBATCH --output=dclampDataExtractor2.out
#SBATCH --error=dclampDataExtractor2.err
#SBATCH --account=netlinks                  # the account/allocation I am using
#SBATCH --mail-user=al4ng@virginia.edu      # address to mail
#SBATCH --mail-type=end                     # send mail on certain events; type can be BEGIN, END, FAIL,
REQUEUE, and ALL
#SBATCH --job-name=dclampDataExtractor2             # job arrays should be named

module load matlab
bash sampleTop2.sh al4ng MATLAB 5 &

# Run matlab
matlab -nodisplay -nosplash \
-r "parpool('local', 15); dclampDataExtractor; exit;"       # Each set has either 15, 20 or 25 sweeps
```

- **dclampDataExtractor2.err**:

```
/bin/rm: cannot remove `Top.out': No such file or directory
{Error using parallel_function (line 604)
All workers aborted during execution of the parfor loop.

Error in dclampDataExtractor (line 598)
                                            parfor swp = 1:nswps
                        % FOR each sweep
}
{The client lost connection to worker 8. This might be due to network problems,
or the interactive communicating job might have errored.
}
slurmstepd: Exceeded job memory limit at some point.
```

- Excerpt from **dclampDataExtractor2.out**:

```
loading .mat files for the set A092110_0005 ...
Analyzing input resistance ...
[Warning: A worker aborted during execution of the parfor loop. The parfor loop
will now run again on the remaining workers.]
[> In parallel_function (line 596)
  In dclampDataExtractor (line 598)]
```

- Attempt #3:
    - Changed the number of workers to **13**.
    - Result: Sweep analyses stopped at **A092110_0014**. Apparently, one of the cores ran out of memory.
    - Efficiency (from **jobe -v**):

```
JobID     Start              Queue      Size(cores)  Time(hours)  Utilization  Sampled         Node(s)
2633561   2016-10-16T17:43:20 serial       14          0.26          62%         no (too short)   udc-ba34-16j
```

- **dclampDataExtractor3.slurm**:

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=14                # how many processes I will run per node; using 20 sometimes eat out
memory
#SBATCH --time=3:00:00                      # amount of time for the whole job d-hh:mm:ss
```

```
#SBATCH --partition=serial                       # the queue/partition I will run on (economy/parallel/serial)
#SBATCH --output=dclampDataExtractor3.out
#SBATCH --error=dclampDataExtractor3.err
#SBATCH --account=netlinks                       # the account/allocation I am using
#SBATCH --mail-user=al4ng@virginia.edu    # address to mail
#SBATCH --mail-type=end                          # send mail on certain events; type can be BEGIN, END, FAIL,
REQUEUE, and ALL
#SBATCH --job-name=dclampDataExtractor3            # job arrays should be named

# Load newest version of Matlab (2016a)
module load matlab

# Run a bash script that prints out all processes
# containing "MATLAB" by user "al4ng" every 5 seconds
bash sampleTop2.sh al4ng MATLAB 5 &

# Run matlab
matlab -nodisplay -nosplash \
-r "parpool('local', 13); dclampDataExtractor; exit;"       # Each set has either 15, 20 or 25 sweeps
```

- **dclampDataExtractor3.err**:

```
{Error using parallel_function (line 604)
All workers aborted during execution of the parfor loop.

Error in dclampDataExtractor (line 598)
                                                parfor swp = 1:nswps
                           % FOR each sweep
}
{The client lost connection to worker 4. This might be due to network problems,
or the interactive communicating job might have errored.
}
slurmstepd: Exceeded job memory limit at some point.
```

- Excerpt from **dclampDataExtractor3.out**:

```
loading .mat files for the set A092110_0014 ...
Analyzing input resistance ...
[Warning: A worker aborted during execution of the parfor loop. The parfor loop
will now run again on the remaining workers.]
[> In parallel_function (line 596)
  In dclampDataExtractor (line 598)]
>> The command used to run the batch was:
sbatch dclampDataExtractor3.slurm
```

- Attempt #4:
    - Modified **plot_LTS.m** & **plot_IPSC_peak.m** so that it figures are closed after saving (i.e., added **close(h)**)
    - Changed the number of workers back to **19**.
    - Result: Successful completion of sweep analyses, but FindIndToFit was still attempting to find special cases despite fitmode == 1. Also, still gives memory limit exceeded error.

- **dclampDataExtractor4.slurm**:
  (Similar to dclampDataExtractor1.slurm)
- **dclampDataExtractor4.err**:

```
{Undefined function or variable 'Noisy_recording'.
Error in FindIndToFit (line 81)
                    && (ismember(fnrow(k), Noisy_recording) ...
Error in PlotHistogramsRefineThreshold (line 120)
          indtofit = FindIndToFit(fnrow_old, cellidrow_old, prow_old, grow_old,
     fitmode, infolder);
Error in dclampDataExtractor (line 928)
          PlotHistogramsRefineThreshold(1, outfolder, outfolder);
}
slurmstepd: Exceeded job memory limit at some point.
```

- Excerpt from **dclampDataExtractor4.out**:

```
loading .mat files for the set H101310_0003 ...
Analyzing input resistance ...
Finding IPSC offsets (obsolete) ...
Finding and plotting IPSC peaks ...
Finding and plotting LTSs ...
Recording sweep properties ...
Elapsed time is 20.040327 seconds.

The highest LTS peak without bursts has amplitude == -48.0066 mV
Plotting histograms and refining threshold ...
Using fit mode == 0 ...
Using matfile == /scratch/al4ng/m3ha/data_dclamp/take4/dclampdatalog_take4.mat ...
Using max_numComponents == 3 ...
Using lts_thr == -0.0023 ...
Using lts_thr_alt == -0.0081823 ...
Finding new LTS threshold ...
Possible 2nd derivative threshold for bursts is -0.00314733
[Warning: ProbDistUnivParam will be removed in a future release. Use fitdist or
makedist instead.]
[> In ProbDistUnivParam (line 91)
  In fit_gaussians_and_refine_threshold (line 96)
  In PlotHistogramsRefineThreshold (line 202)
  In dclampDataExtractor (line 927)]
New LTS threshold is -0.0019 V^2/s^2
Alternate LTS threshold is -0.00314733 V^2/s^2
Using fit mode == 1 ...
Made directory /scratch/al4ng/m3ha/data_dclamp/take4/histograms_100-400all/
```

- Efficiency (from **jobe -v**):

```
JobID    Start                Queue    Size(cores)  Time(hours)  Utilization  Sampled    Node(s)
2642701  2016-10-16T21:07:31  serial   20           1.64         41%          yes        udc-ba34-16j
```

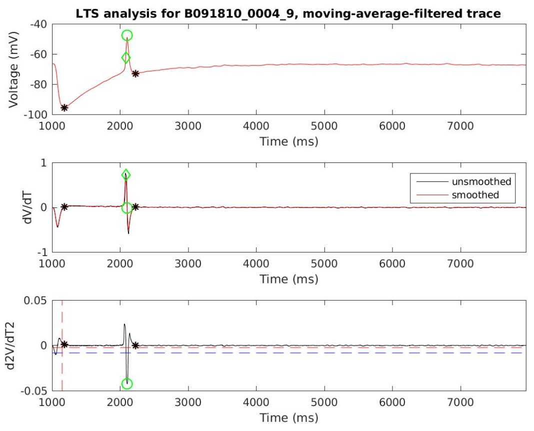
- Conclusions:

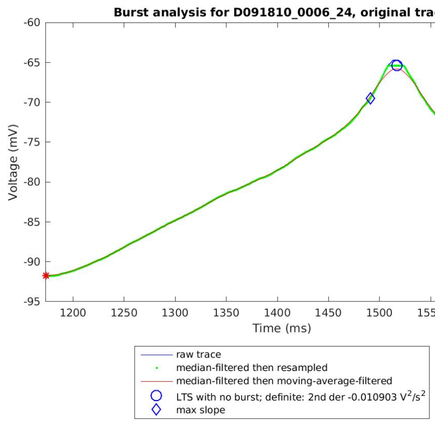
- Attempt #1, using more cores per node, actually crashed *later* than attempts #2 or #3. Therefore, the total RAM used is not an issue, but the **RAM used per core** is probably an issue
- Need to reexamine code to release more memory in the parfor loops
- Need to fix FindIndToFit.m

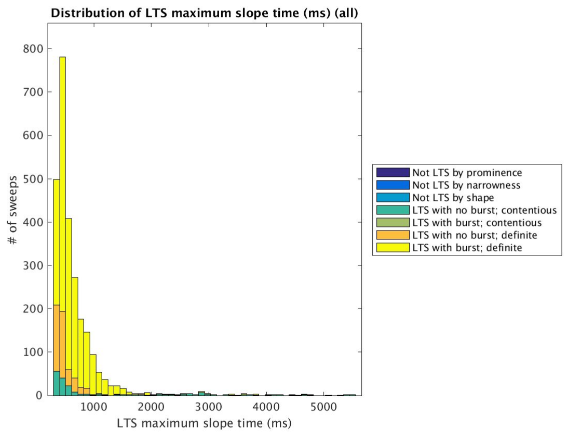
10/13/2016~10/16/2016

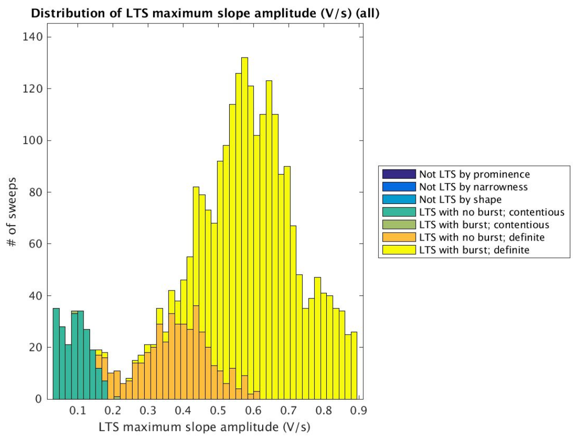
Dynamic clamp data analysis (cont'd)

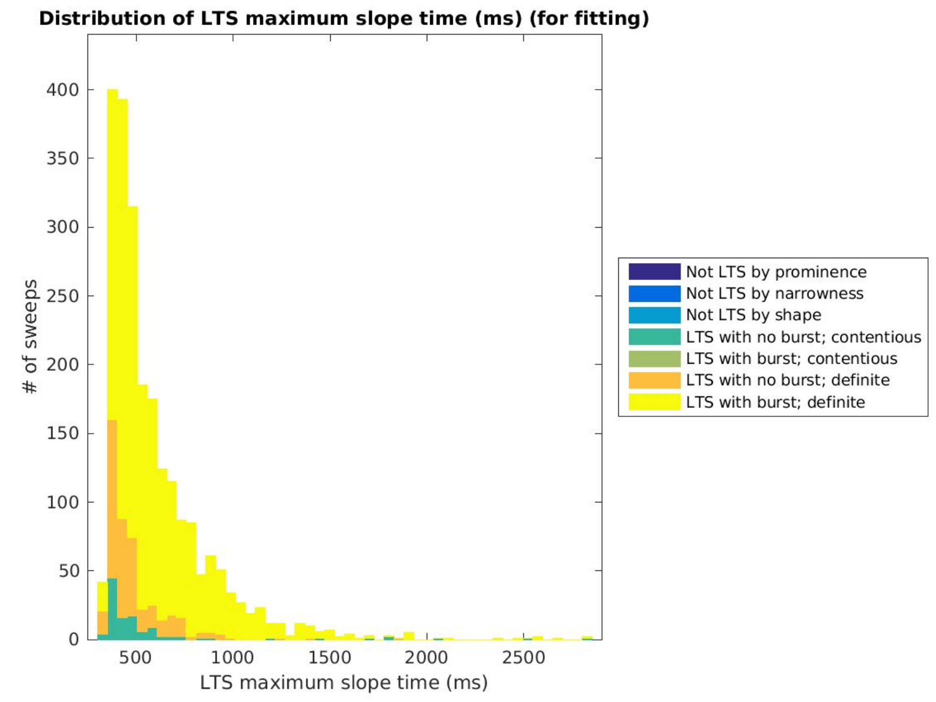
- Updates to **dclampDataExtractor.m**:
  - dclampdatalog_take4.csv now overwrites whenever saveswpinfoflag == 1
  - Added **maxslopetime** & **maxslopeval** and plot them as diamonds

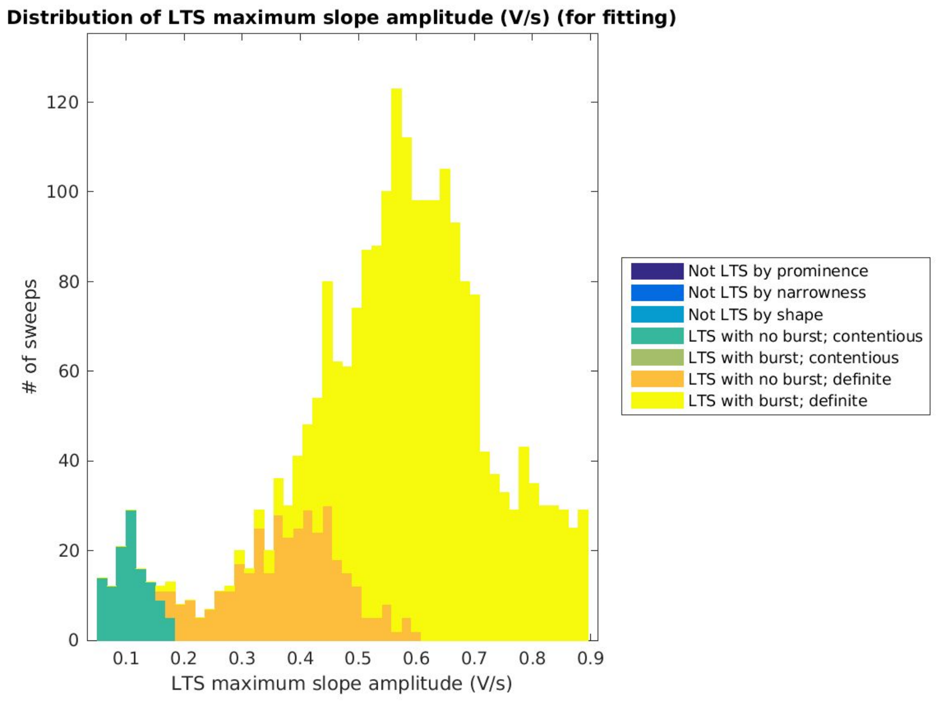






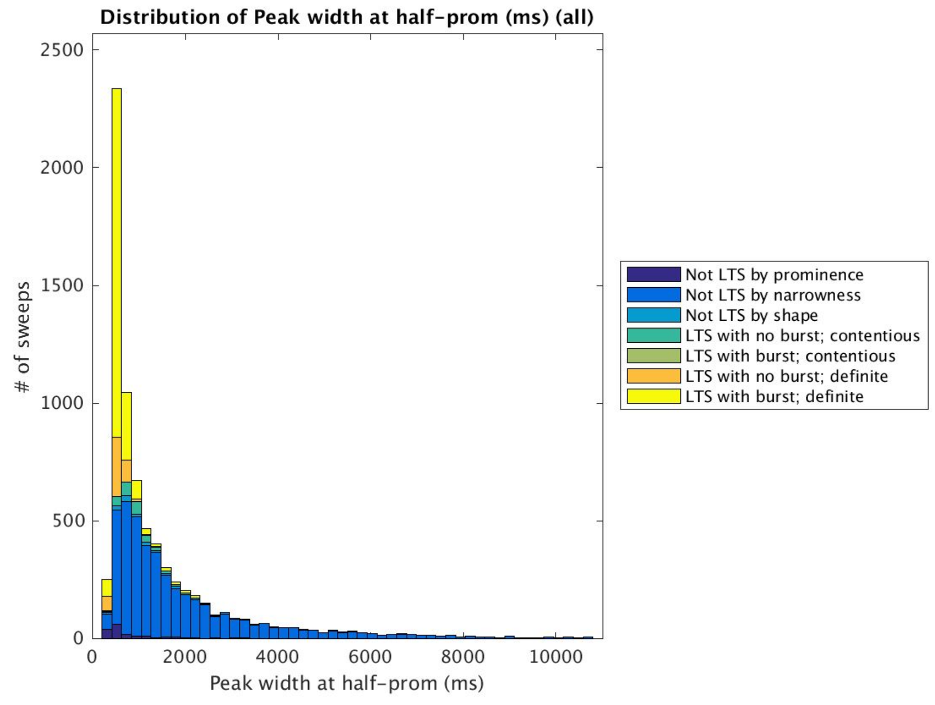


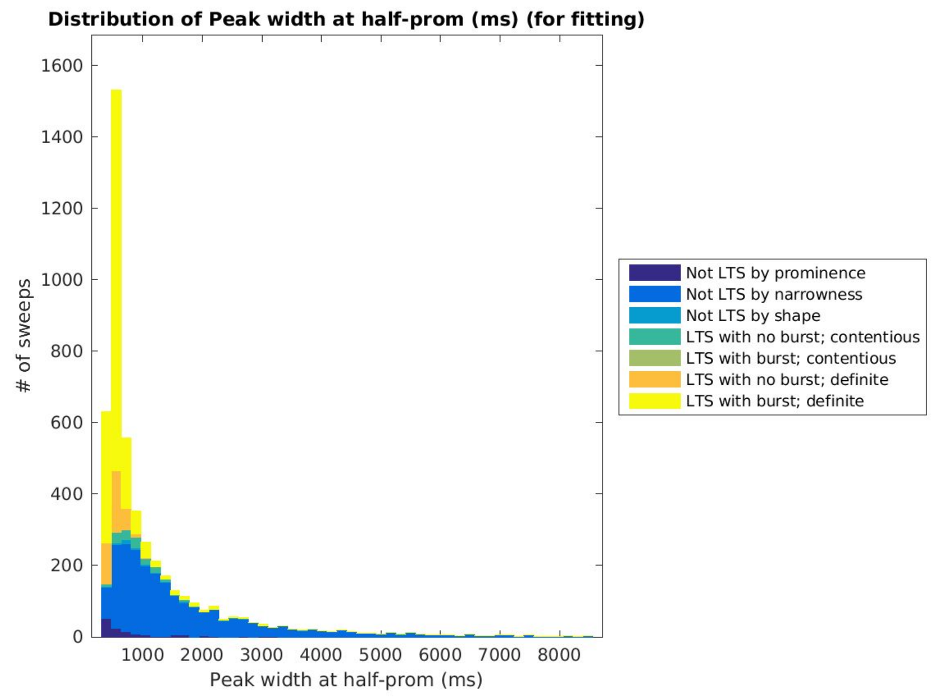




- Combined all the LTS detection to **find_LTS.m** under /home/Matlab/Adams_Functions
- Combined all the IPSC peak detection to **find_IPSC_peak.m** under /home/Matlab/Adams_Functions

- Added **peakwidth** as a statistic to save

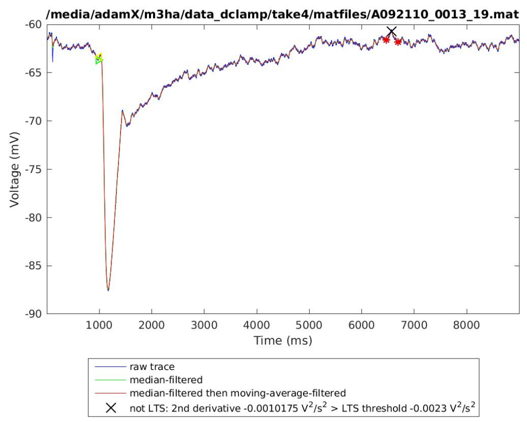




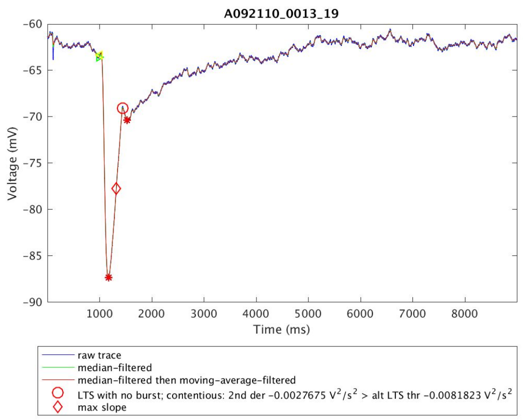
- Updates to **PlotHistogramsRefineThreshold.m**:
    - Changed the way vectors are imported so that it takes an arbitrary number of sweep info vectors; now needs **find_ind_str_in_cell.m** from /home/Matlab/Adams_Functions
    - Added **suffix**; changed the directory name for fitmode == 0 to include suffix '**_all**'
- Updates to **dclampdatalog_analyze.m**:
    - Added **suffix**; changed the directory name for fitmode == 0 to include suffix '**_all**'
    - Made infolder and outfolder optional arguments
- Updates to **find_IPSC_peak.m**:
    - Can now read more than one current vector at a time

- Updates to **find_LTS.m**:
    - Now plots **maxslope** with a diamond
    - Changed **maxnoise** so that it's **2*standard deviation** of the baseline data (68% of noise should be within this range) instead of the maximal range. This changed **LTS onset time** for **45** traces (see **to_compare_lts_old6_against_old5**).

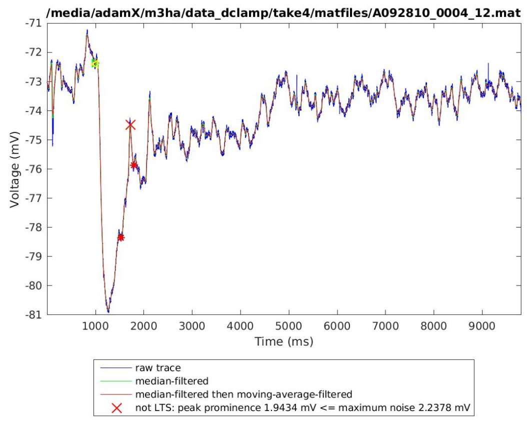
Old:



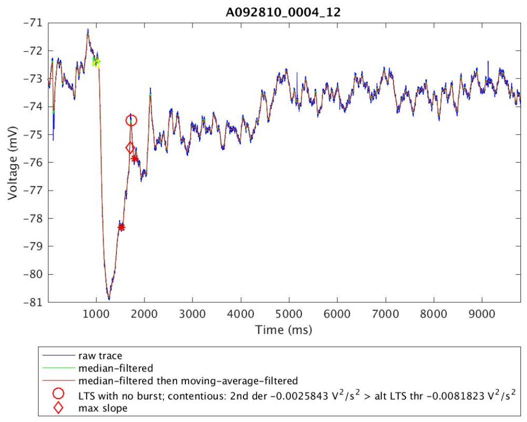
New:



Old:

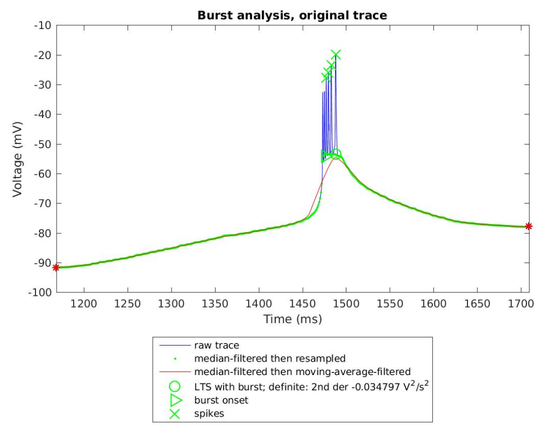


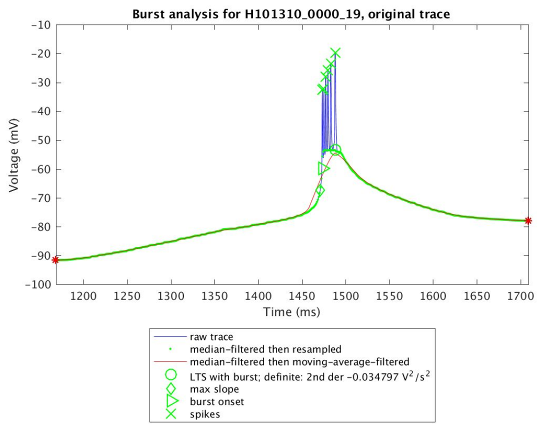
New:



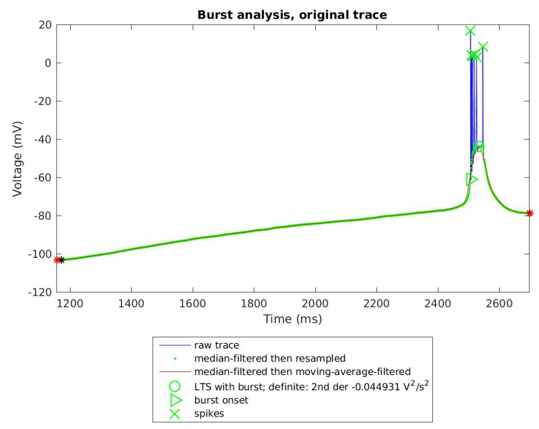
- ○ Changed spike threshold from **-30 mV** to **-43 mV.** This changed **spikes per peak** for **75** traces (see **to_compare_spp_old6_against_old5**).

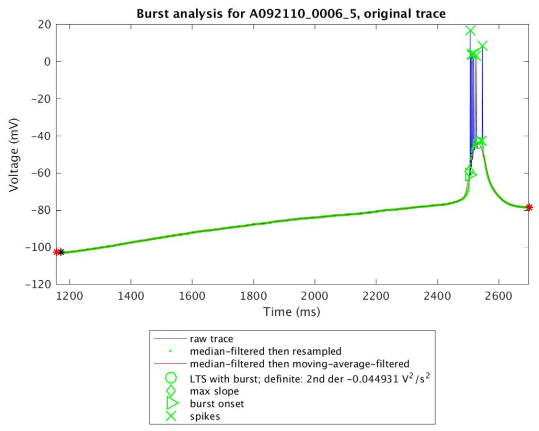
Old: New:





Old: New:





- ○ Made **tvec2, vvec1, vvec2, vvec3** optional arguments for efficiency

- Reran **dclampDataExtractor.m** on Rivanna (**version 6**). Ran **backup_files.sh**
- Ran **find_more_gray_area_traces.sh**: Looked through **special_cases/unclassified** and reclassify

<u>Plan for next week</u>

- EEG:
  - Learn how to make headsets
  - Learn how to implant
  - Practice implants
- Data Analysis - current/conductance traces:
  - Figure out the reason for the remaining discrepancies
- Data Analysis - voltage traces:
  - Fix spike detection?
  - Change **maximum noise** measurement again?.
  - Ran **compare_statistics.m**: Compared with **version 5** (old5); find all traces with altered LTS onset times and reclassify; find all traces with altered spikes per peak and reclassify
  - Run **copy_LTS_figures.sh**, then **backup_figures.sh**, then, then **update_figures.sh**: Examine each special cases folder and look for any classification discrepancies
  - Go through the set “**Not_LTS_by_prom**.”
  - Fix the condition “**Spontaneous_spikes_in_baseline**”
  - Go through the sets “**Wide_LTS_could_be_noise,**” “**Noise_in_trace,**” and “**Looks_like_LTS**” decide whether or not to overrule.
  - Decide how to classify “**Noisy_recording**” (to be taken out of fitting) vs “**Noise_in_trace**” (to be overridden and included in fitting)
  - Do we still take out **Noisy_recording_fixed** from the fitting even though it’s fixed? Should we go through the data and look for other undetected noisy recordings?
  - Compute new histograms, thresholds & bargraphs
  - Rerun dclampDataExtractor.m; find all traces with altered LTS onset times and reclassify; find all traces with altered spikes per peak and reclassify; find all “gray area traces” and reclassify. Compare with old4
  - Look through “gray area traces,” reclassify
  - Reexamine all problematic traces; decide what to do with the remaining false negatives/positives
  - OPTIONAL: Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly

- Brian’s tasks:
  a. Plot correlation diagrams. One way is sufficient. Take out IPSC_offset.
  b. Color code according to peak_class and create legend.

c. Analyze the correlation between **spikes per peak**, **spikes per burst**, **burst probability** and other features of an LTS, such as **LTS onset time**, **LTS amplitude**, **LTS prominence, LTS maximum slope**.
d. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
e. Help analyze special cases

- Fitting:
    - Ask Dr. Meliza about u(t)
    - Make the following parameters a **constant** throughout (same for all cells):
        i. specific membrane capacitance (**cm**) = **0.88 uF/cm^2**
        ii. axial resistivity (**Ra**) = **173 Ω·cm**
        iii. Electrode series resistance (**Rs**) = **10 MΩ**
    - Make the following parameters vary **across cells**:
        i. leak conductance (**g_pas**)
        ii. leak reversal potential (**e_pas**)
        iii. dendritic correction factor (**corrD**)
    - Make the following parameters vary **across trials**:
        i.
    - Try binning the traces for each cell by **Vhold** levels and deriving passive parameters from the double exponential curve fits
        i. Ch 4 of Johnston & Wu
        ii. Problem 4.10.14(c) of Johnston & Wu
    - If (a) is still not helpful, instead of calculating the input resistance and using it, fit the model to the current injection responses **of each cell** and **of each Vhold level** to extract parameters relevant to the input resistance
        i. Try making separate estimates for each binned Vhold level
        ii. Use the sum of squares error over all traces
    - Try varying **initial conditions** for *state variables* and *parameters* to test for **robustness**.
    - Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time; Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data
    - Figure out whether we actually need to separate the third compartment into two. Or fix the relative length of the most distal dendrite (**distdendpercent**) to 0.5.
    - Fit the model to find the best parameters that will be used in network simulations

- SWD detection (w/ Vignesh):
    - Change the input into an Excel file and an output into a text file
    - Figure out a way to read in two channels simultaneously and detect SWDs that are present in BOTH channels
    - Maybe figure out a way to detect behavioral arrests in the third channel

    - Figure out how to use the text file in PClamp to aid the manual SWD detection process.

- Rivanna/Parallelization
    - Fix FindIndToFit.m
    - Need to reexamine code to release more memory in the parfor loops
    - Go through the Parallel Matlab lectures
    - Parallelize the data analyses codes as much as possible
    - Parallelize the optimization codes as much as possible
    - Parallelize the simulation codes as much as possible

- Johnston & Wu:
    - Read Ch 4
    - Do Ch 4 Problems

9/19/2016~9/21/2016

Spike-wave discharge (SWD) detection

- Modified code so that it reads .mat files as inputs
- Fixed the size of the GUI so that all components will show up
- Translated buttons on the GUI
- Modified code so that it outputs Excel files without error and in English
- Modified code so that it shows the initial detection and the refined detection separately
- Parameters used to find SWDs:

Parameter	Default value
Window length (**WL**)	**0.5** [sec]
Signal to noise ratio (**SNR**)	**3^(1/2)**
Minimum duration (**minD**)	**2** [sec]
Baseline variance (**BV**, computed)	[μV²]
Minimum frequency (**minF**)	**6.0** [Hz]
Minimum frequency (**maxF**)	**11** [Hz]
Frequency proximity (**FP**)	**0.7** [Hz]

- Algorithm/operational definition for finding SWDs:
    1. **Variance**: Variance of the signal over each unit window of length **WL** must be *greater than* **SNR² x BV**, where BV is computed by taking the variance of the signal over a baseline region *manually selected*
    2. **Duration**: Duration of the SWD must be *at least* (>=) **MD**
    3. **Spectral power**: One of the following must be true:
        i. Frequency with the highest power (**f1**) is in the interval [**minF**, **maxF**]
        ii. Frequency with the 2nd highest power (**f2**) is in the interval [**minF**, **maxF**] and the frequency with the highest power (**f1**) is within **FP** of **2*f2**
        iii. Frequency with the 3rd highest power (**f3**) is in the interval [**minF**, **maxF**] and the frequency with the 2nd highest power (**f2**) is within **FP** of **2*f3**

    Note: If 1 & 2 are true, it is an **SWD candidate**.

9/22/2016~9/23/2016

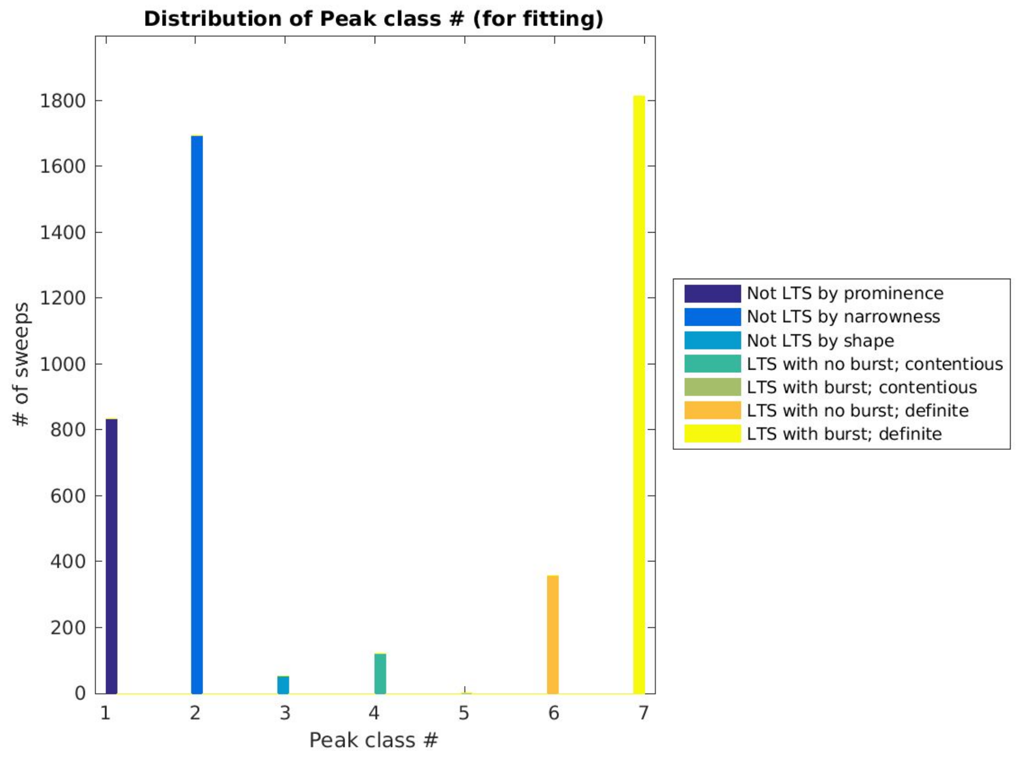
__**Transfer of LTS detection codes**__

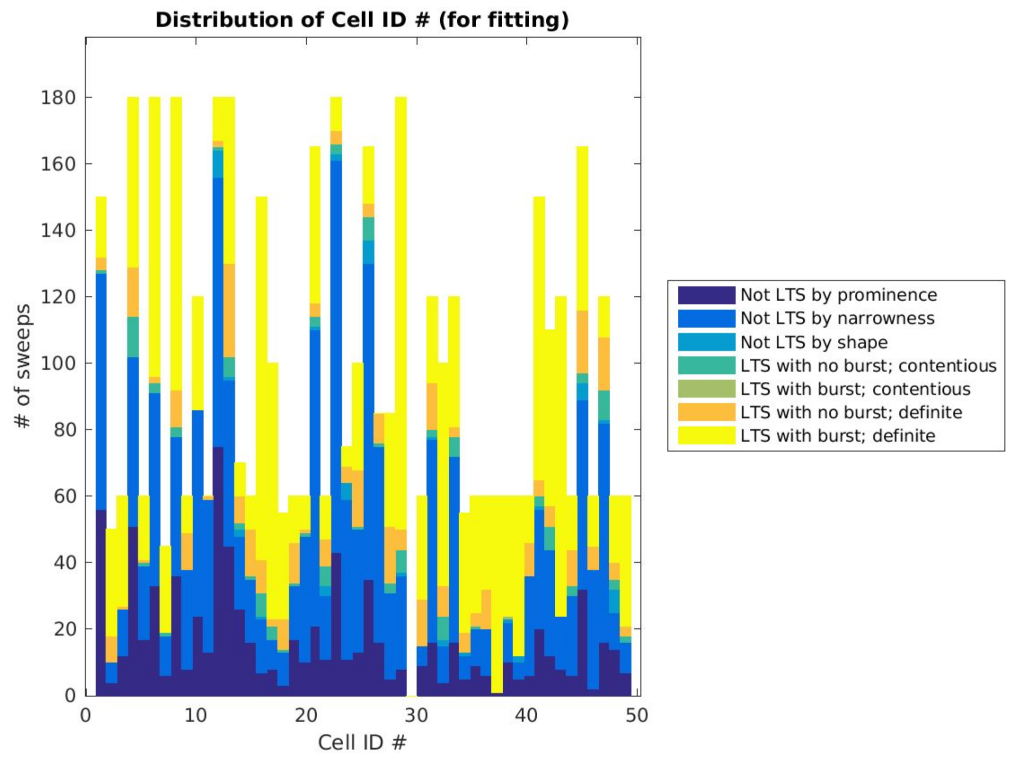
- The current LTS detection codes from dclampDataExtractor.m were transferred into a function called "**find_LTS.m**"
    - To see how to use it, type "**help find_LTS**"
    - This function and all others mentioned below will show up if the following folder is added to the path: **home/Matlab/Adams_Functions**
    - The full detection results can be viewed under the created folder **/vtraces/** or **/vtraces_scaled/**
    - A walk-through of the LTS detection process can be viewed under the created folder **/LTSanalysis/**
    - A close-up view of the selected peak can be viewed under the created folder **/burstanalysis/**
    - Any traces with 2nd derivatives in the gray area are copied under the created folder **/gray_area_traces/**
    - The analysis results are stored in the matfile "**XXX_analysis.mat**"
    - Note that in the code, a "peak" means an LTS candidate, whereas a "spike" refers to an action potential.
- The find_LTS function requires the time of current application (istart) as an input. This can be found using "**find_istart.m**"
    - To see how to use it, type "**help find_istart**"
    - The detection result can be viewed under the created folder **/istart/**
- To load and plot from abf files, use the function "**plot_traces_abf.m**"
    - To see how to use it, type "**help plot_traces_abf**"
    - The plotted figures can be viewed under the created folder **/XXX_traces/**
- To see an example of how to combine these functions for high-throughput analysis, see "**test_sweeps.m**" under /media/shareX/share/Adam/Sample_files_from_Katie
- Two caveats:
    - The detection is pretty good, but far from perfect. As seen in C20160922_0002_7, the actual LTS is not detected, possibly because its prominence is smaller than noise. And the reason that the prominence is so small is that the "follow-up spike" is too close to the LTS.
    - Since in Christine's data, the smallest action potential was shooting above the largest LTS peak, the spike detection part could be simply implemented by a threshold definition. However, this could potentially cause small amplitude spikes to be undetected, so the old spike detection code might need to be integrated in the future.

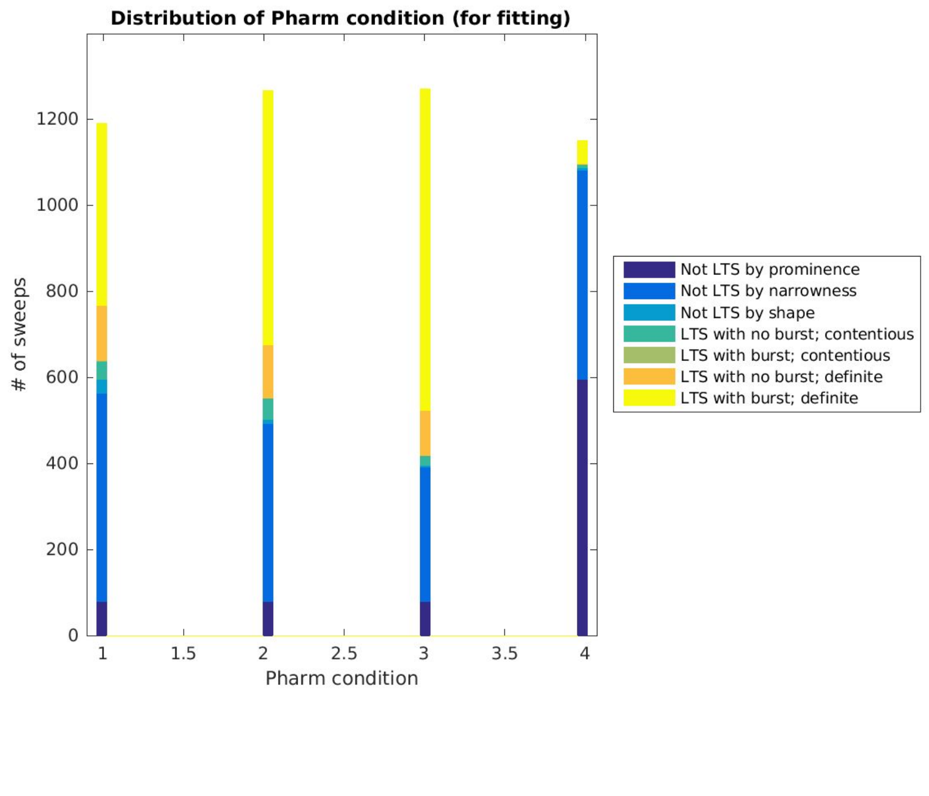
9/23/2016~10/2/2016

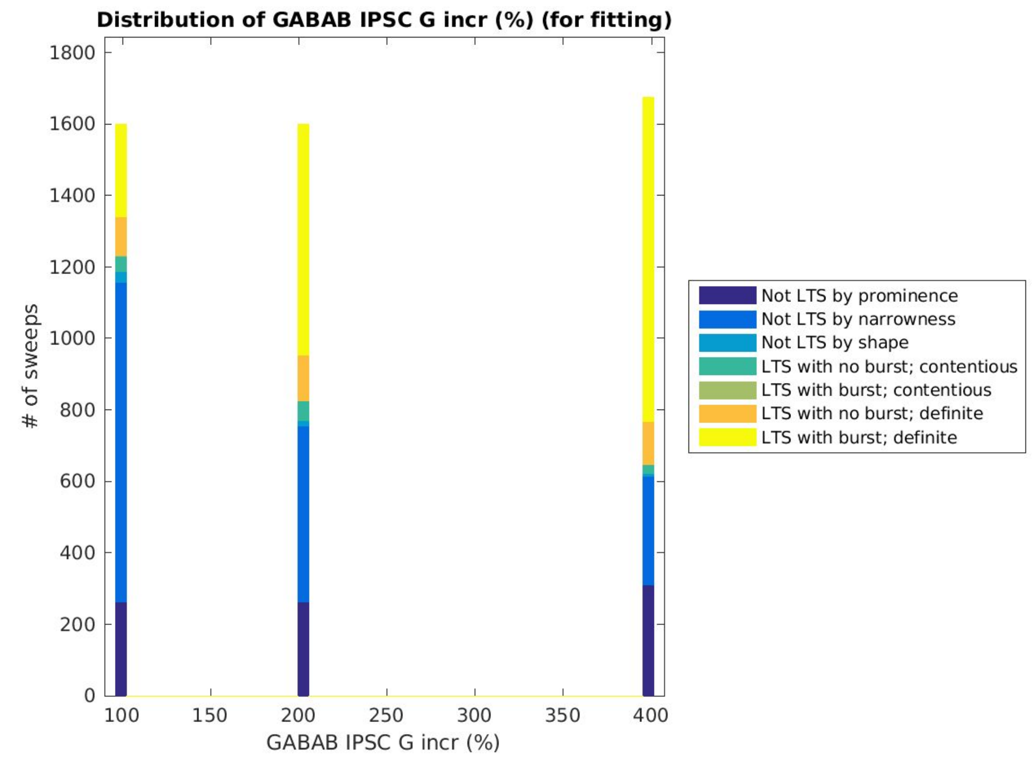
<u>Dynamic clamp data analysis (continued)</u>

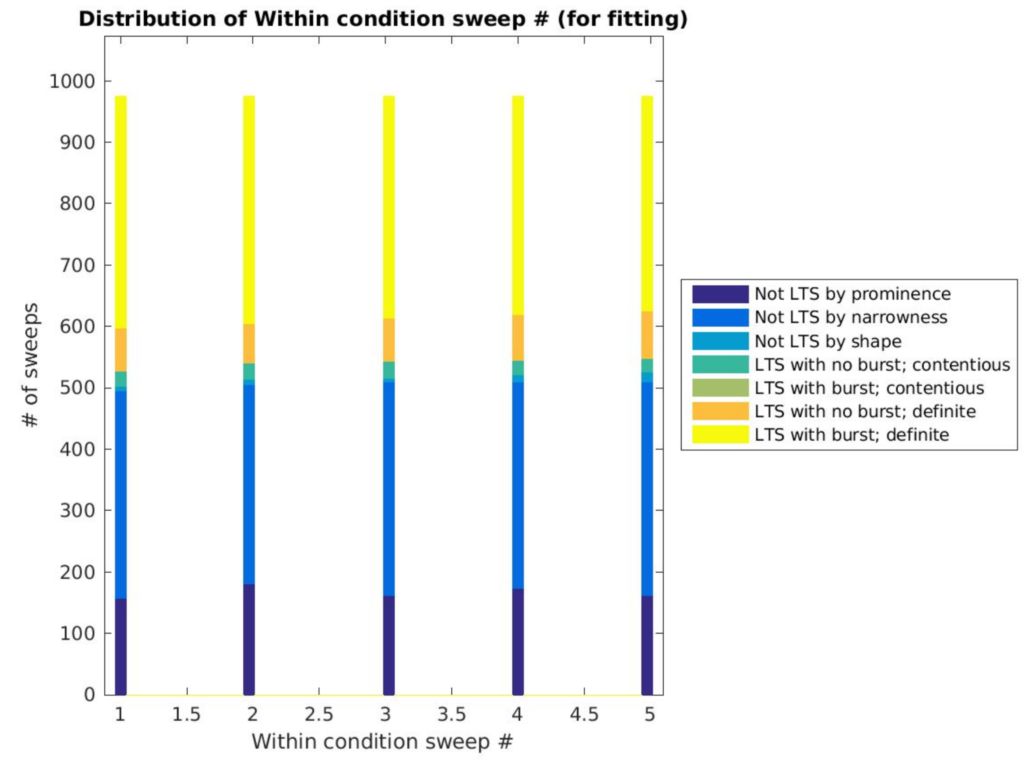
- Created function **plot_traces_mat.m**, **test_sweep.m** to test individual sweeps
- Modified histograms so that it plots each class separately

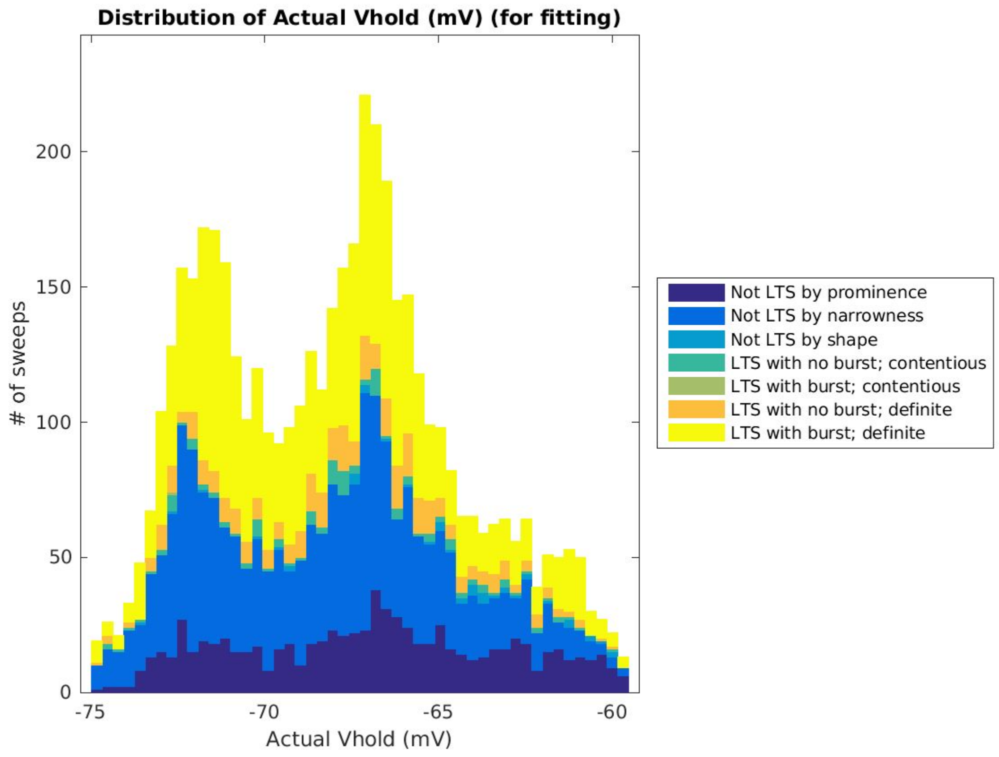


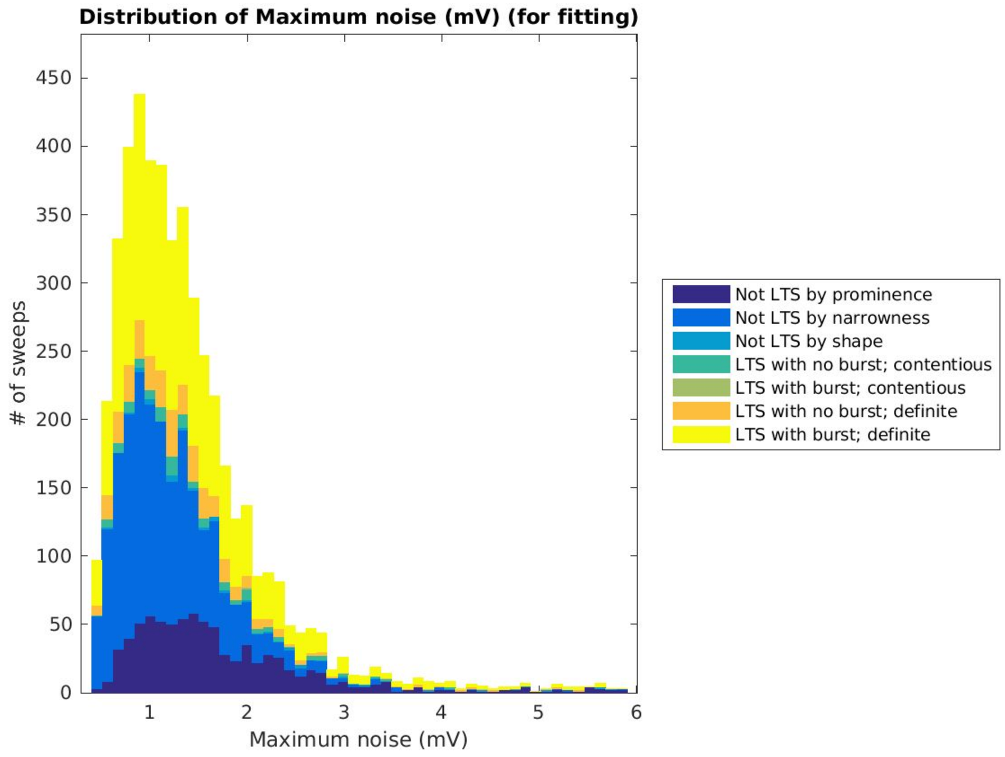

Distribution of Cell ID # (for fitting)
# of sweeps
Cell ID #
0
20
40
60
80
100
120
140
160
180
10
30
50
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

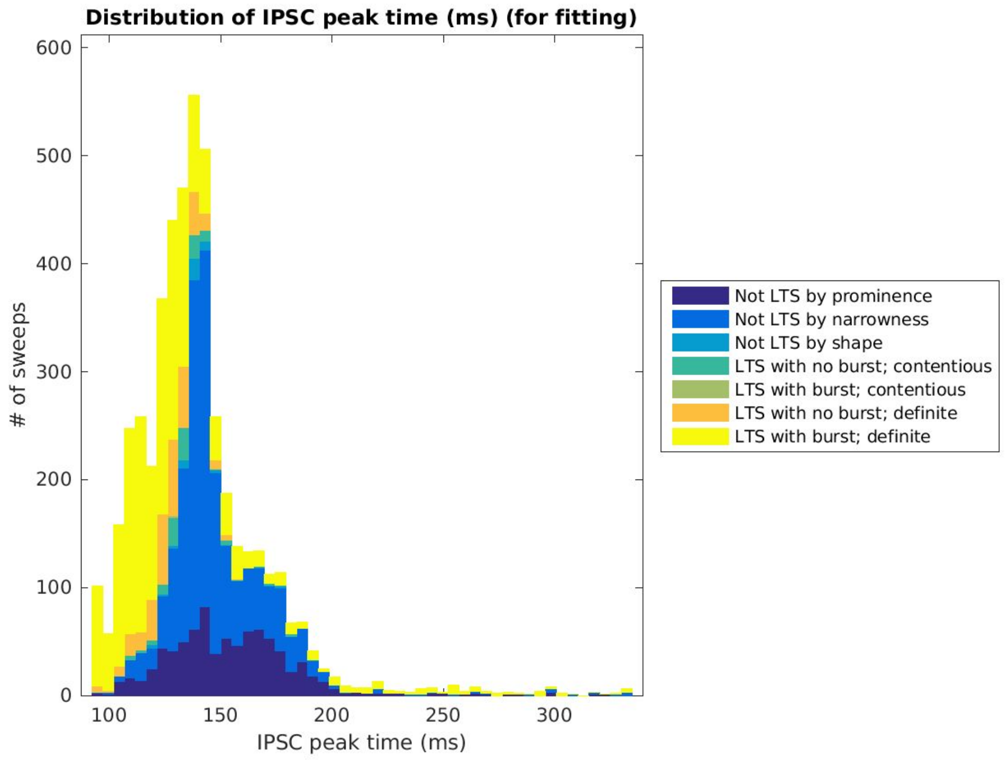

Distribution of Pharm condition (for fitting)
# of sweeps
Pharm condition
0
200
400
600
800
1000
1200
1
1.5
2
2.5
3
3.5
4
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

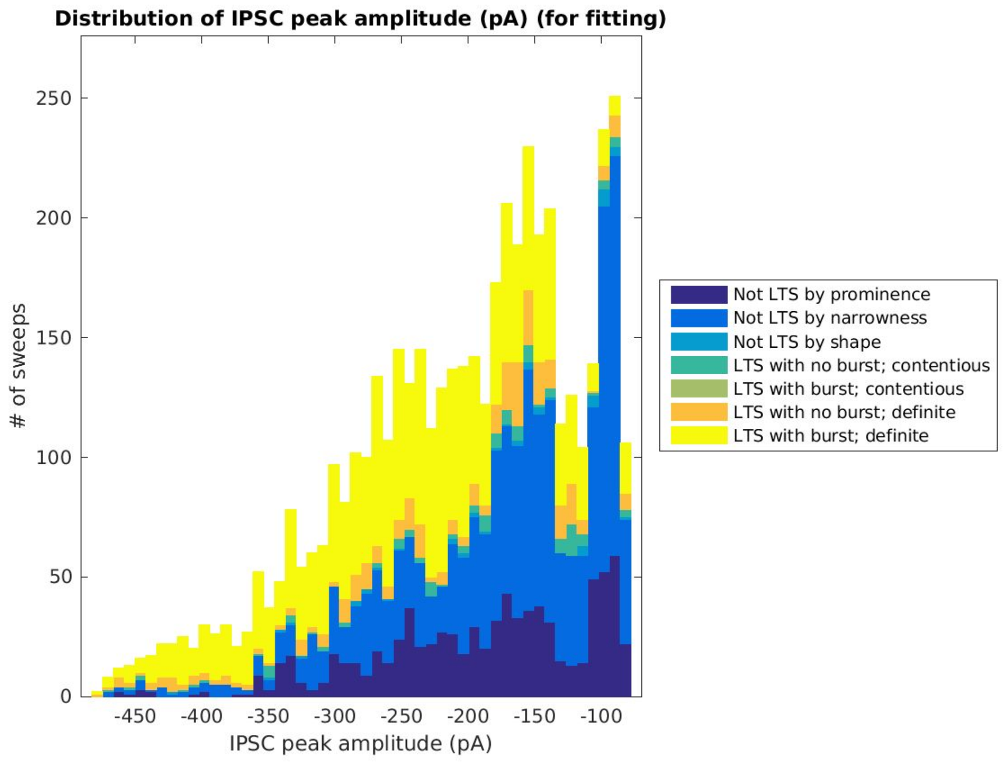

Distribution of GABAB IPSC G incr (%) (for fitting)
# of sweeps
0
200
400
600
800
1000
1200
1400
1600
1800
100
150
200
250
300
350
400
GABAB IPSC G incr (%)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

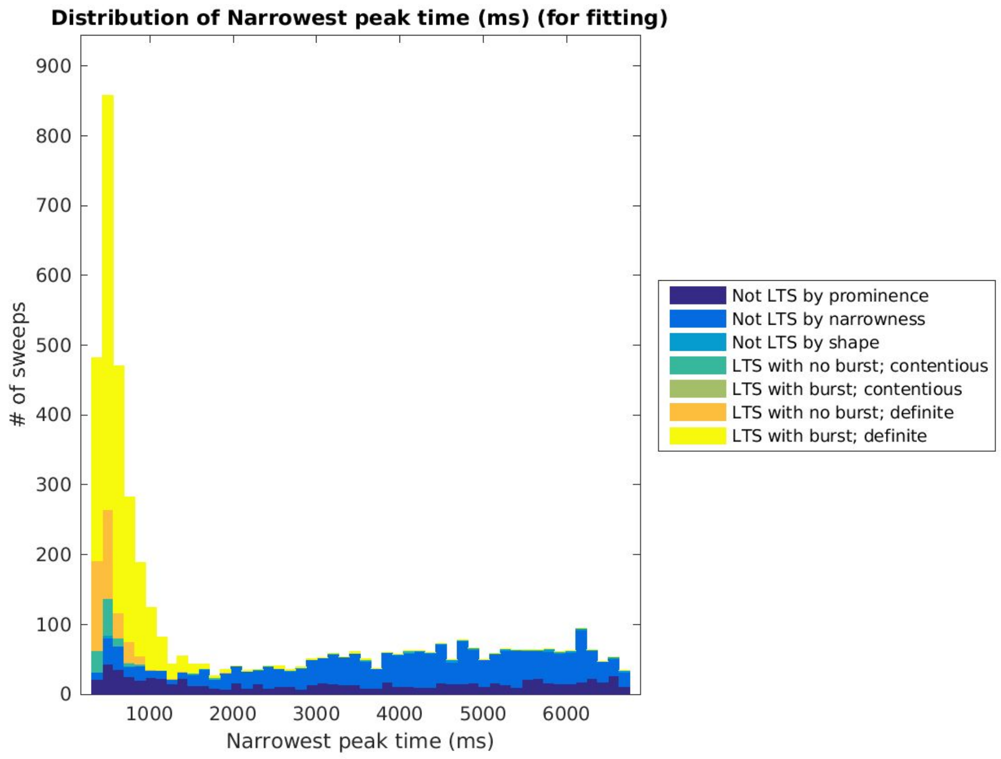

Distribution of Within condition sweep # (for fitting)
# of sweeps
0
100
200
300
400
500
600
700
800
900
1000
1
1.5
2
2.5
3
3.5
4
4.5
5
Within condition sweep #
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite


Distribution of Actual Vhold (mV) (for fitting)
# of sweeps
0
50
100
150
200
-75
-70
-65
-60
Actual Vhold (mV)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

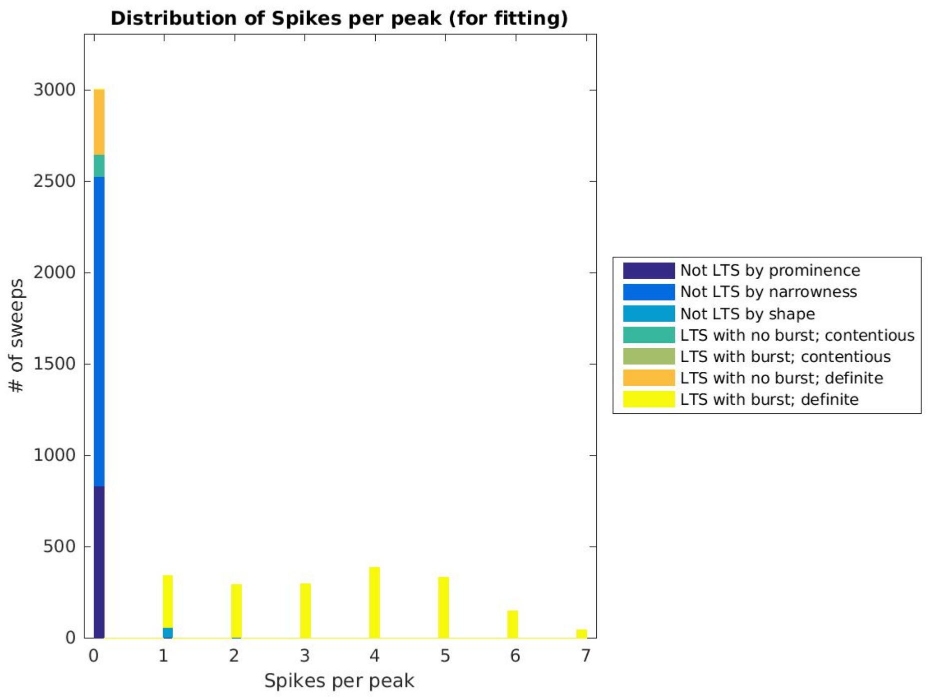

Distribution of Maximum noise (mV) (for fitting)
# of sweeps
0
50
100
150
200
250
300
350
400
450
1
2
3
4
5
6
Maximum noise (mV)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite


Distribution of IPSC peak time (ms) (for fitting)
# of sweeps
0
100
200
300
400
500
600
100
150
200
250
300
IPSC peak time (ms)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

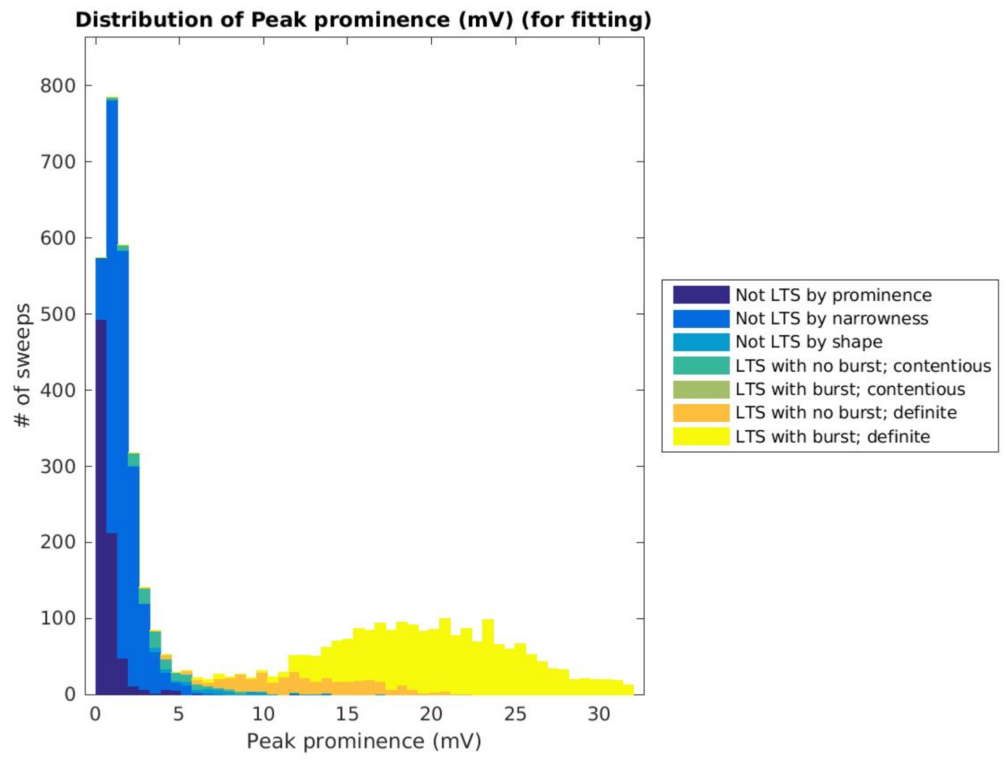

Distribution of IPSC peak amplitude (pA) (for fitting)
# of sweeps
0
50
100
150
200
250
-450
-400
-350
-300
-250
-200
-150
-100
IPSC peak amplitude (pA)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

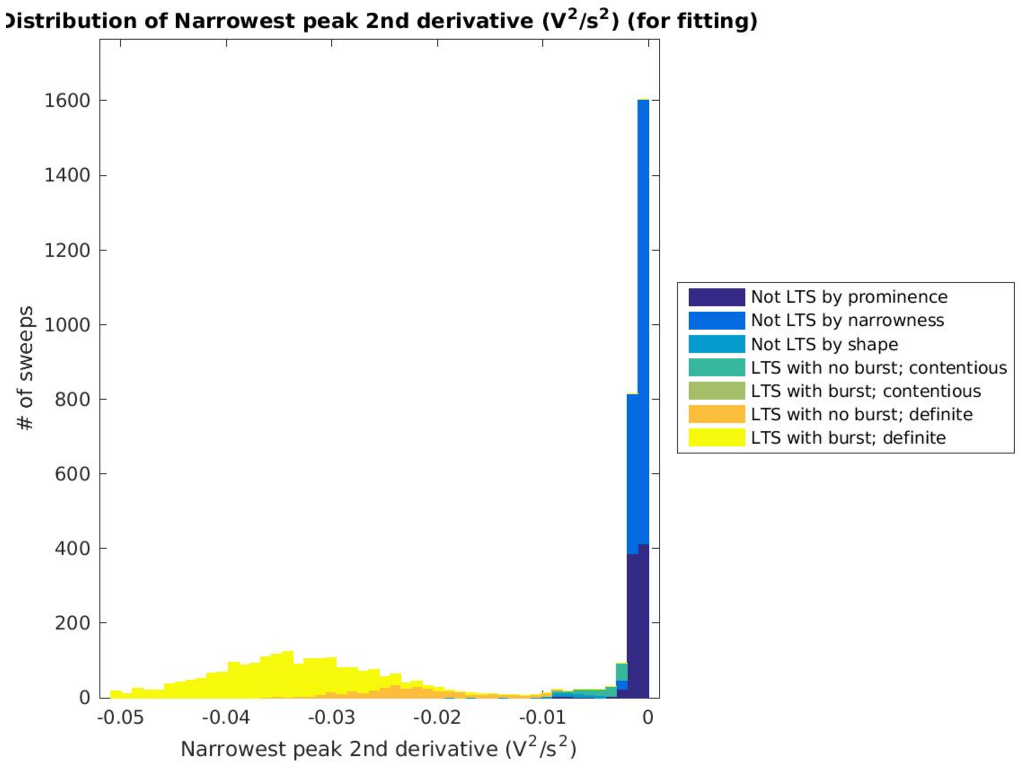


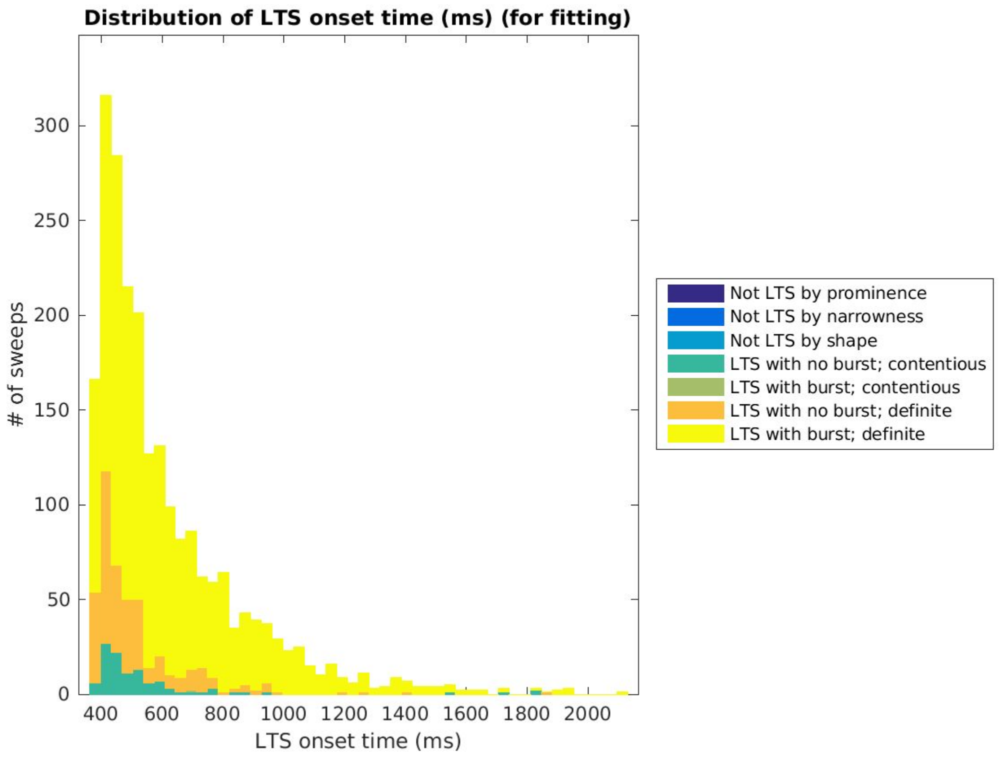
Distribution of Narrowest peak time (ms) (for fitting)
# of sweeps
Narrowest peak time (ms)
0
100
200
300
400
500
600
700
800
900
1000
2000
3000
4000
5000
6000
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

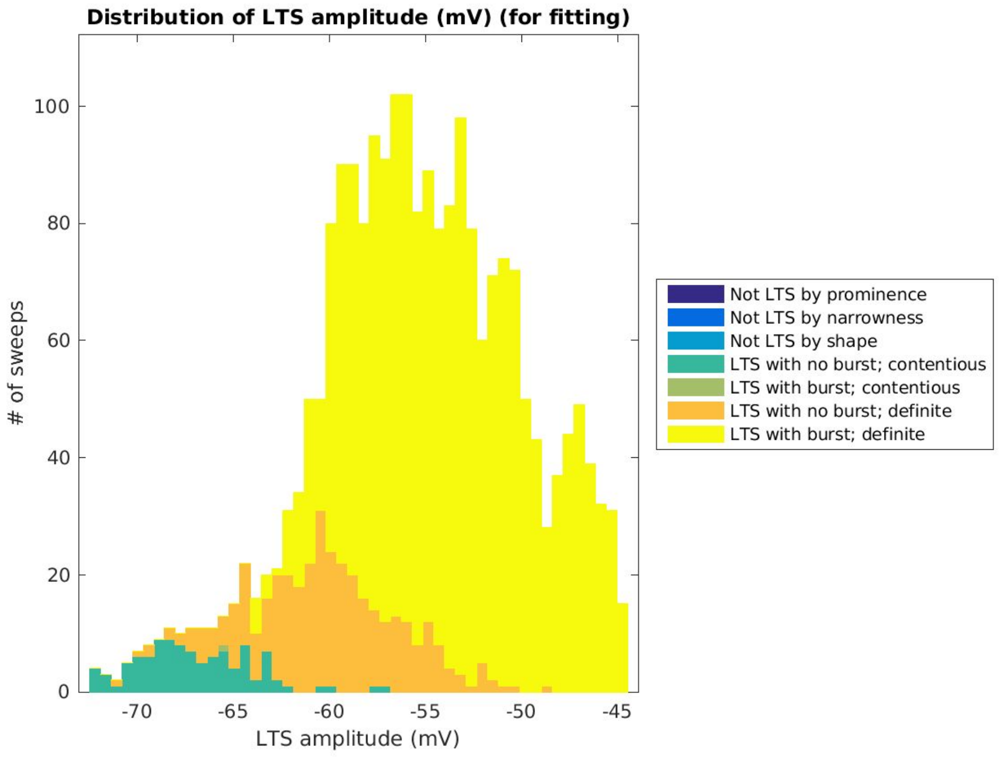


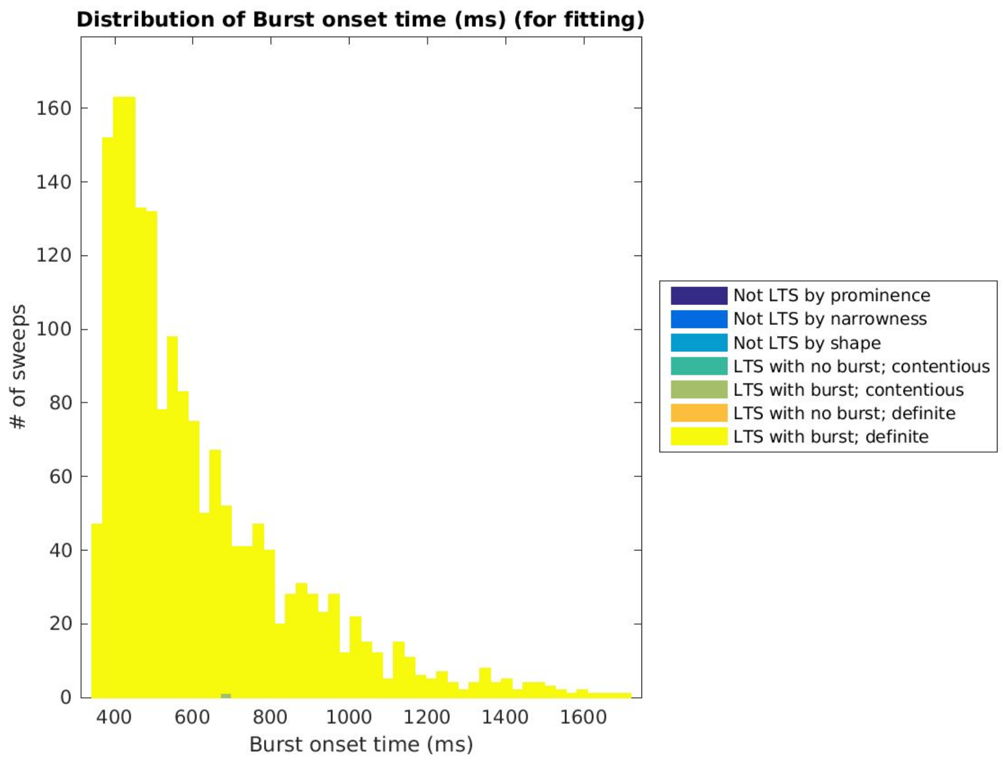
Distribution of Spikes per peak (for fitting)
# of sweeps
Spikes per peak
0
500
1000
1500
2000
2500
3000
0
1
2
3
4
5
6
7
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

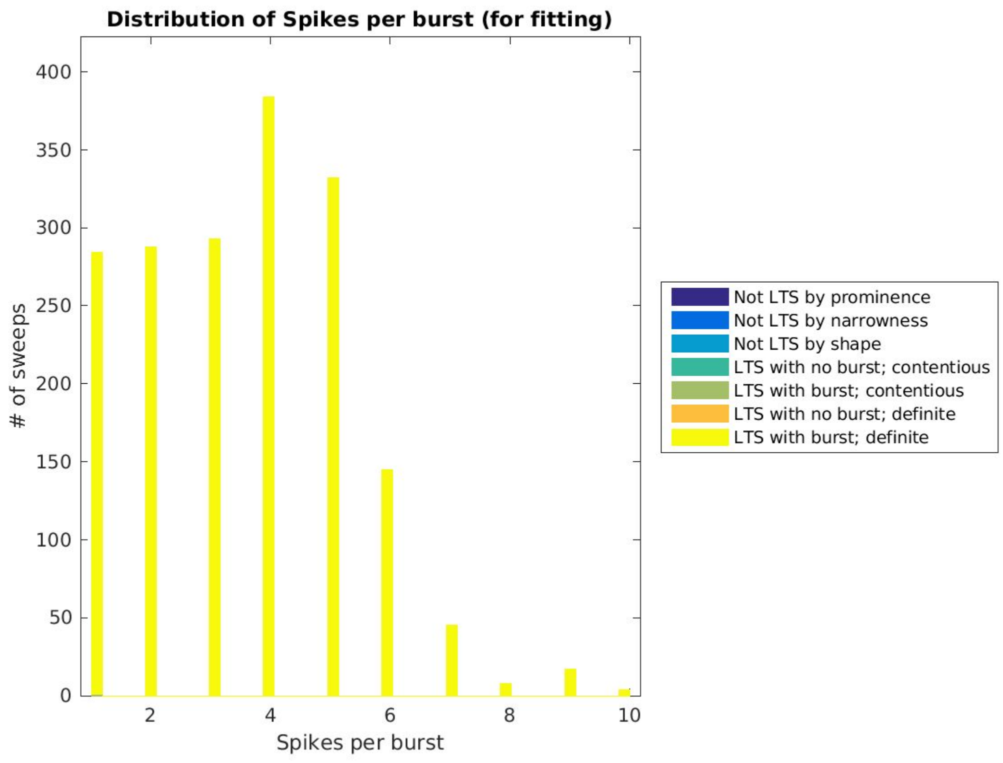

Distribution of Peak prominence (mV) (for fitting)
# of sweeps
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30
Peak prominence (mV)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite


Distribution of Narrowest peak 2nd derivative (V^2/s^2) (for fitting)
# of sweeps
1600
1400
1200
1000
800
600
400
200
0
-0.05
-0.04
-0.03
-0.02
-0.01
0
Narrowest peak 2nd derivative (V^2/s^2)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

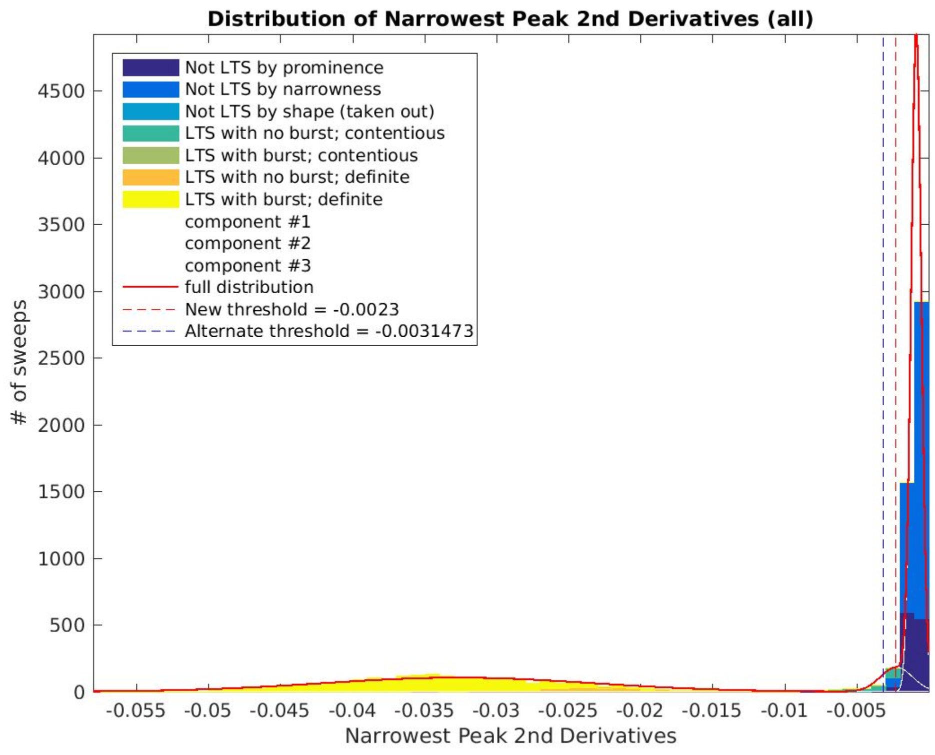

Distribution of LTS onset time (ms) (for fitting)
# of sweeps
0
50
100
150
200
250
300
400
600
800
1000
1200
1400
1600
1800
2000
LTS onset time (ms)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

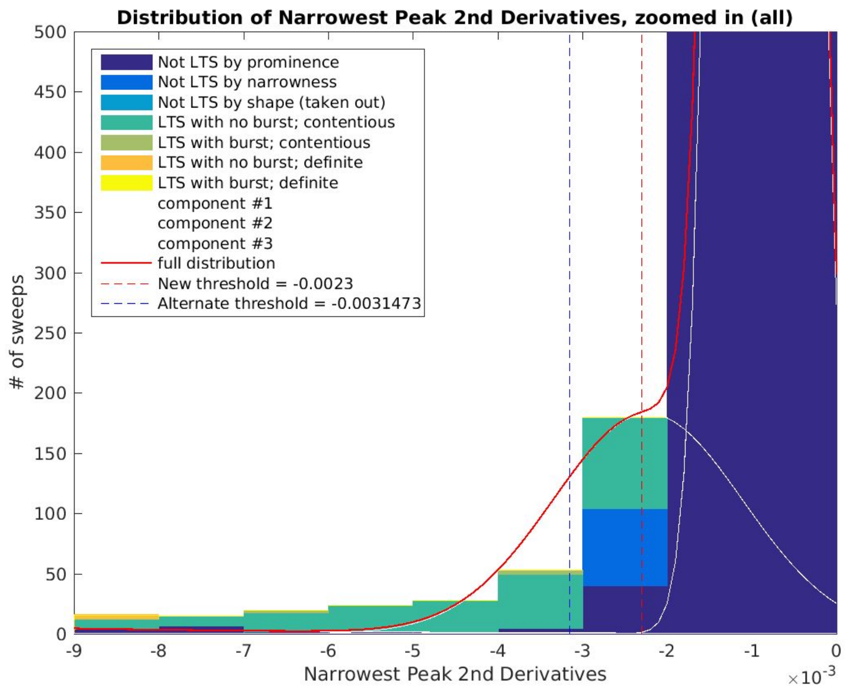

Distribution of LTS amplitude (mV) (for fitting)
# of sweeps
0
20
40
60
80
100
-70
-65
-60
-55
-50
-45
LTS amplitude (mV)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

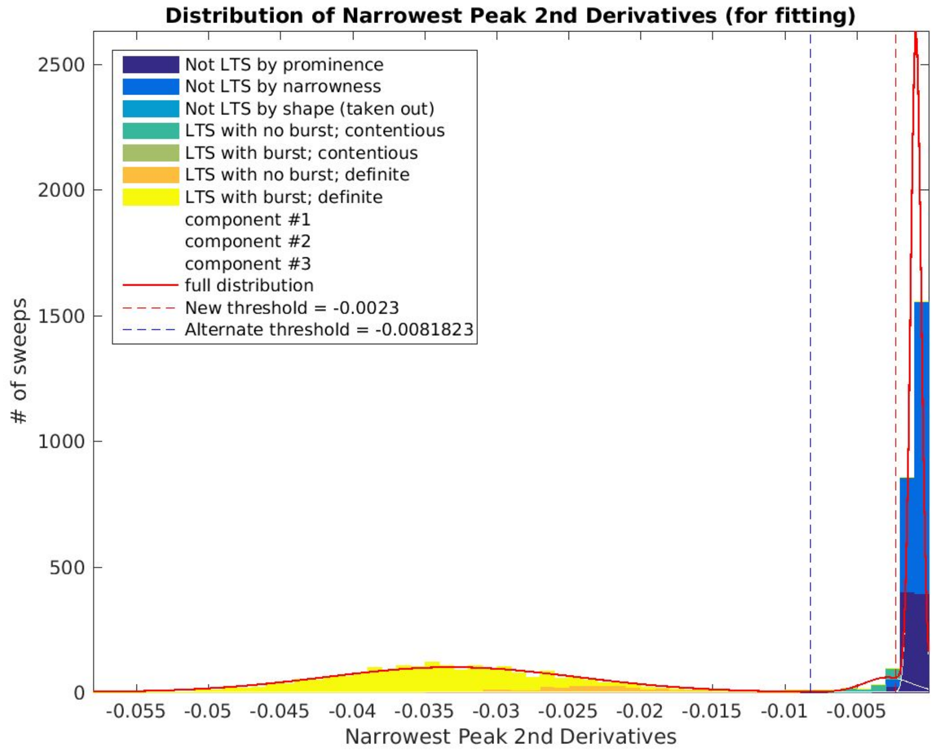

Distribution of Burst onset time (ms) (for fitting)
# of sweeps
0
20
40
60
80
100
120
140
160
400
600
800
1000
1200
1400
1600
Burst onset time (ms)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

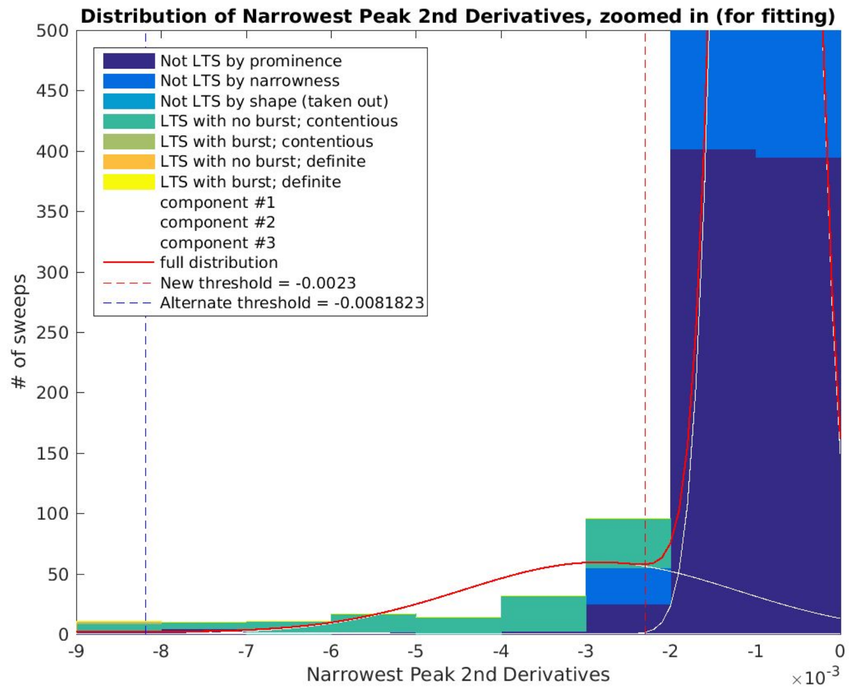

Distribution of Spikes per burst (for fitting)
# of sweeps
0
50
100
150
200
250
300
350
400
2
4
6
8
10
Spikes per burst
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite

- Recalculate LTS threshold with spontaneous spikes taken out of the histogram (for all fitmodes). Rationale: We are trying to differentiate *LTS* from *noise*, so we don't care about where a spontaneous spike falls in a histogram






Distribution of Narrowest Peak 2nd Derivatives (for fitting)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape (taken out)
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite
component #1
component #2
component #3
full distribution
New threshold = -0.0023
Alternate threshold = -0.0081823
# of sweeps
Narrowest Peak 2nd Derivatives
2500
2000
1500
1000
500
0
-0.055
-0.05
-0.045
-0.04
-0.035
-0.03
-0.025
-0.02
-0.015
-0.01
-0.005


Distribution of Narrowest Peak 2nd Derivatives, zoomed in (for fitting)
Not LTS by prominence
Not LTS by narrowness
Not LTS by shape (taken out)
LTS with no burst; contentious
LTS with burst; contentious
LTS with no burst; definite
LTS with burst; definite
component #1
component #2
component #3
full distribution
New threshold = -0.0023
Alternate threshold = -0.0081823
# of sweeps
Narrowest Peak 2nd Derivatives
×10-3
500
450
400
350
300
250
200
150
100
50
0
-9
-8
-7
-6
-5
-4
-3
-2
-1
0

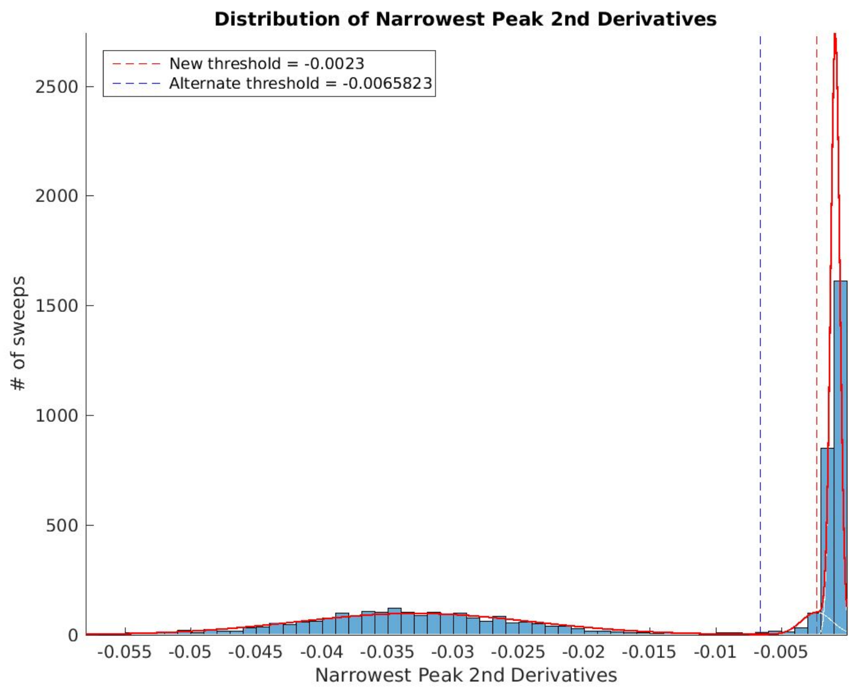
Plan for next week

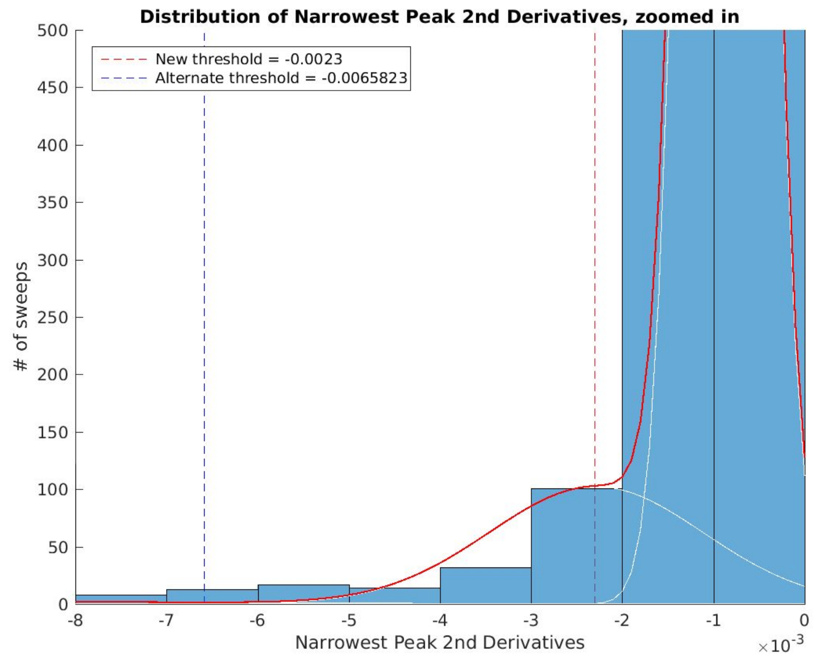
- Data Analysis - current/conductance traces:
    a. Waiting for John's input on the discrepancies
- Data Analysis - voltage traces:
    a. Go through the set "**Not_LTS_by_prom**." Is there a better way of measuring **maximum noise**? Fix the condition "**Spontaneous_spikes_in_baseline**"
    b. Go through the sets "**Wide_LTS_could_be_noise,**" "**Noise_in_trace,**" and "**Looks_like_LTS**" decide whether or not to overrule.
    c. Decide how to classify "**Noisy_recording**" (to be taken out of fitting) vs "**Noise_in_trace**" (to be overridden and included in fitting)
    d. Do we still take out **Noisy_recording_fixed** from the fitting even though it's fixed? Should we go through the data and look for other undetected noisy recordings?
    e. Compute new histograms, thresholds & bargraphs
    f. Rerun dclampDataExtractor.m; find all traces with altered LTS onset times and reclassify; find all traces with altered spikes per peak and reclassify; find all "gray area traces" and reclassify. Compare with old4
    g. Look through "gray area traces," reclassify
    h. Reexamine all problematic traces; decide what to do with the remaining false negatives/positives
    i. Analyze the correlation between the burst probability, spikes per peak, spikes per burst, etc. and the slope or other features of an LTS. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
    j. OPTIONAL: Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly
- Fitting:
    a. Instead of calculating the input resistance and using it, fit the model to the current injection responses to extract parameters relevant to the input resistance
    b. Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time; Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data
    c. Fit the model to find the best parameters that will be used in network simulations
- SWD detection (w/ Vignesh):
    a. Figure out a way to read in two channels simultaneously and detect SWDs that are present in BOTH channels
    b. Maybe figure out a way to detect behavioral arrests in the third channel
    c. Change the input into an Excel file and an output into a text file
    d. Figure out how to use the text file in PClamp to aid the manual SWD detection process.
- Johnston & Wu:
    a. Read Ch 4

9/13/2016~9/18/2016

Dynamic clamp data analysis (continued)

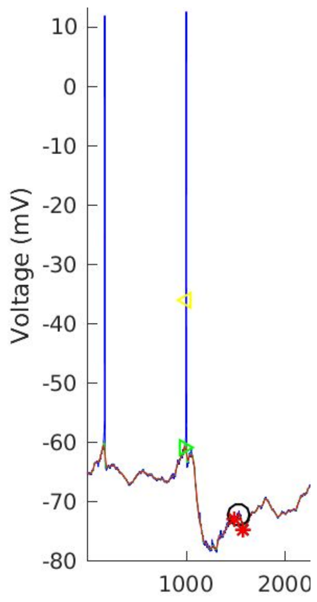
- Computed an alternative LTS threshold (**lts_thr_alt**) in the Gaussian mixture fitting process. This is to define a "**gray area**" in which **false positives** might lie. The following histograms includes g incr = 100%, 200%, 400% only and excludes problematic traces



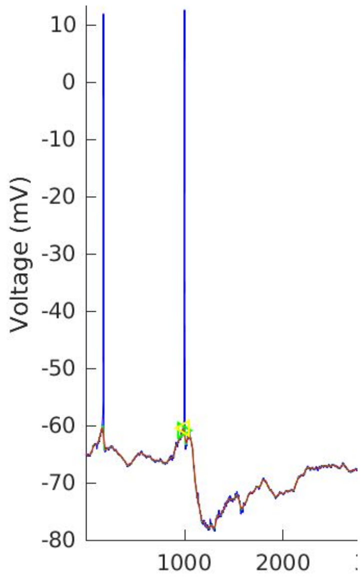


- Fixed “Wrong_holding_potential” due to spontaneous spikes messing up result, see E100810_0006_13 for example
    - Changed trace for measuring holding potential from the original trace to the **median-filtered trace**.

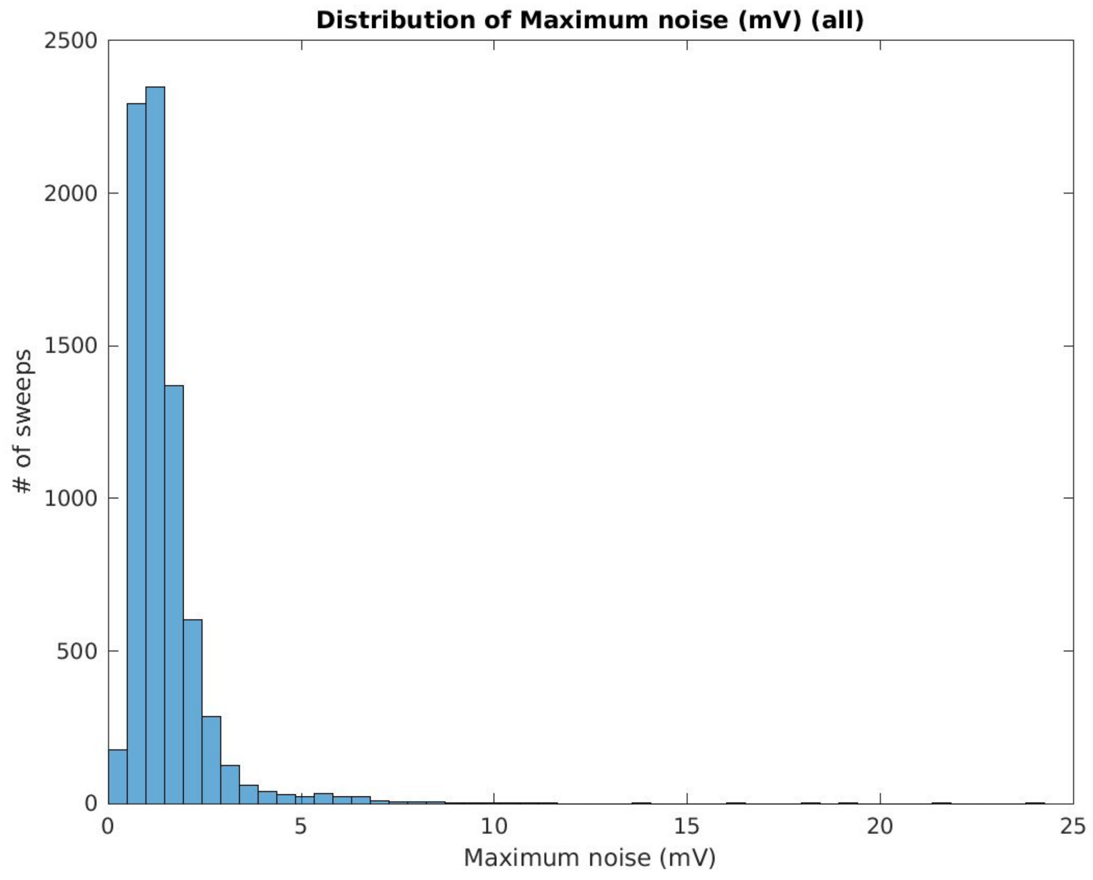
Before:



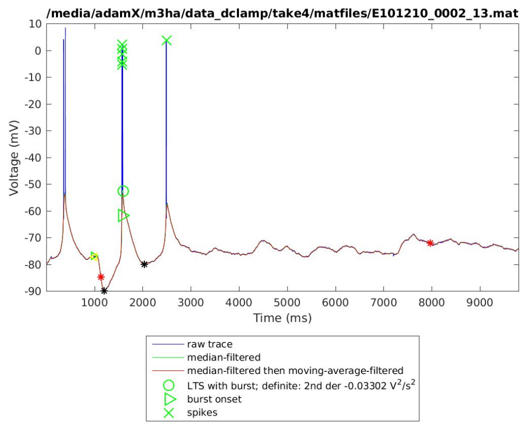
After:



- Attempted to improve noise detection:
    - Changed **order** of LTS determination: now it finds the first spike crossing 2nd derivative threshold before checking whether it’s a spontaneous spike (the reverse happened previously). This should eliminate most of the “Spontaneous_spike_that_did_not_fire”
    - Made the prominence threshold for an LTS equal to maximum noise (new statistic to save **maxnoise**) instead of an arbitrary 1 mV. Maximum noise (in mV) is calculated as the range of median-filtered then moving-average-filtered voltage values between **200~1000 ms**.
    - In summary, the new **operational definition** (by the order of detection in the algorithm) for an LTS is:
        (1) **Peak**: Must be a *local maximum*
        (2) **Prominence**: Prominence must be greater than *maximum noise*
        (3) **Narrowness**: Most negative 2nd derivative within peak must be less than or equal to *LTS threshold*
        (4) **Shape**: If an action potential exists, the first spike must be *behind* the LTS peak

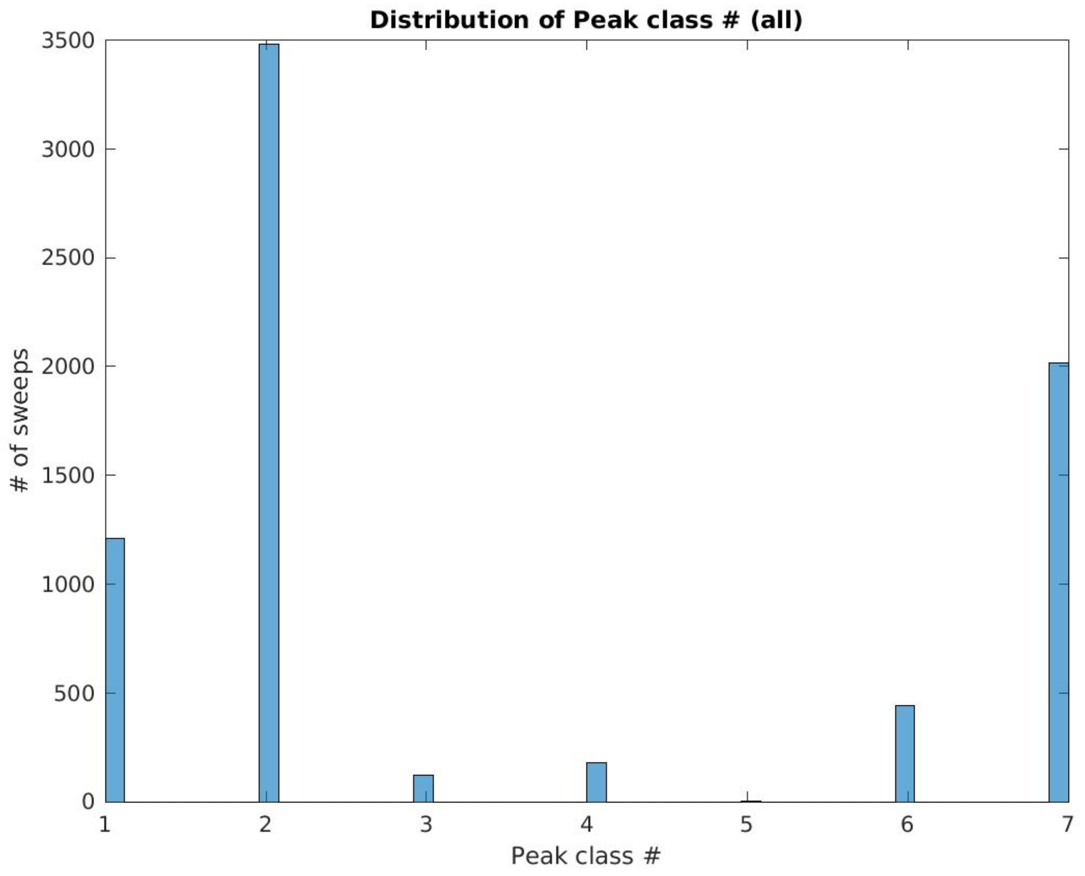


- For the sake of simplification, changed **maxnoiseprom2** (the threshold for detection peak boundaries when counting spikes per peak) to be equal to **maxnoise**.
    - New problem: this will cause **one** trace to overdetect spikes (now in "**Spikes_per_burst_incorrect**"):

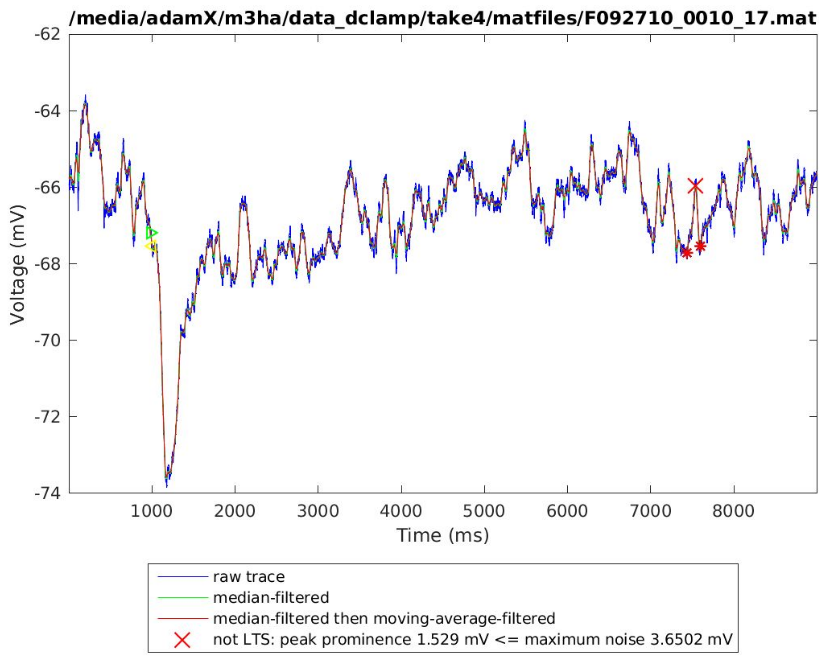


    - Should we just override this (and other traces with spontaneous spikes in baseline) as it's a single trace? Or should we consider this a "Noisy_recording" and take it out of the fitting process?

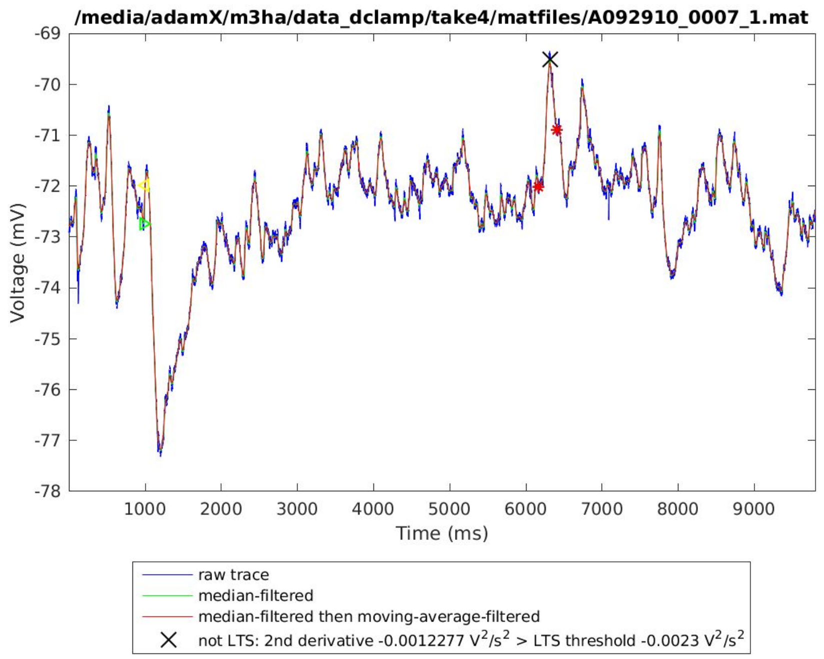
- Added the alternative LTS threshold (**lts_thr_alt**) to dclampDataExtractor.m
  - Marked “gray area peaks” with **red circle** or a **red cross**, depending on whether it was classified as an LTS. Definite LTSs without burst are now marked with a **blue circle**; whereas LTSs with bursts are still marked with a **green circle**.
  - Added the statistic **peakclass**, which saves the result of peak classification for each trace. The classes are defined in **peakclass_labels** and the values (a number from 1~7) set in **pk_class**. The legend of vtrace now prints the pertinent values according to the classification result.
  - Reran dclampDataExtractor.m; found all traces with altered LTS onset times and reclassified; found all traces with altered spikes per peak and reclassified; found all “gray area traces” and reclassified



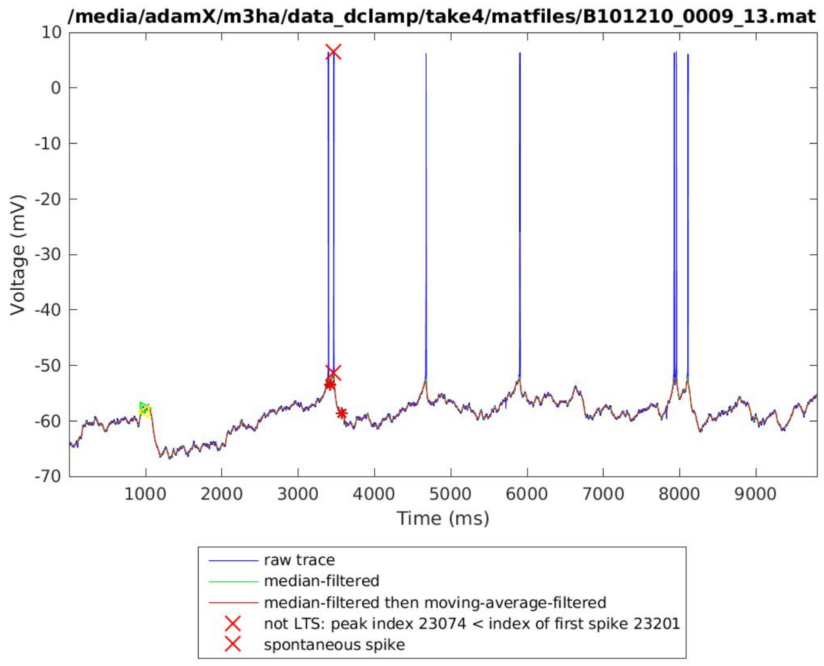
Peak Class #1 - 'Not LTS by prominence':



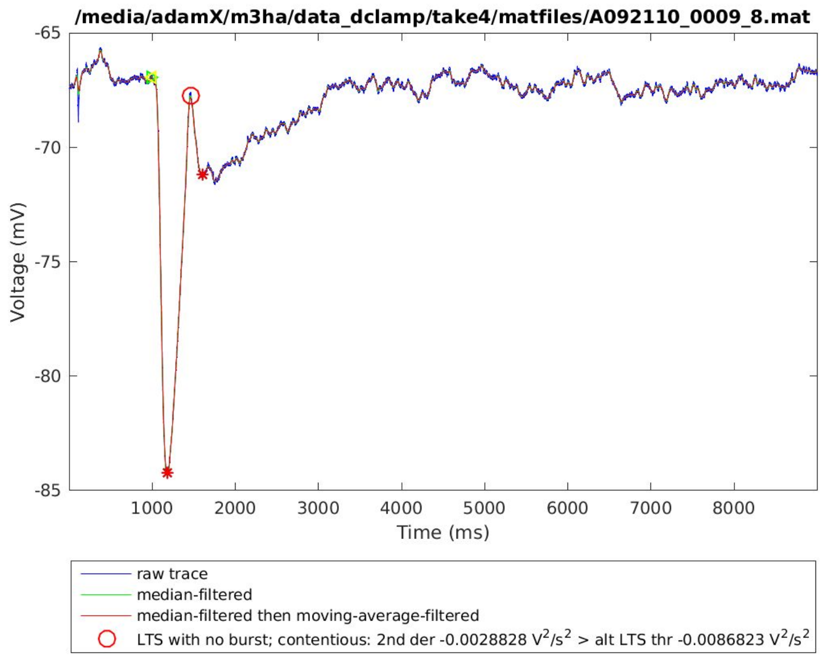
Peak Class #2 - 'Not LTS by narrowness':



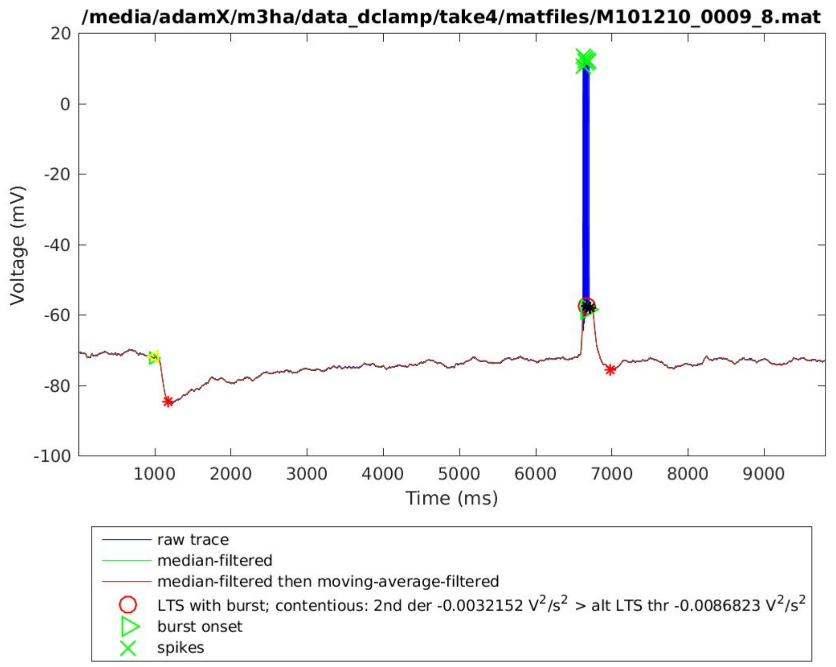
Peak Class #3 - 'Not LTS by shape':



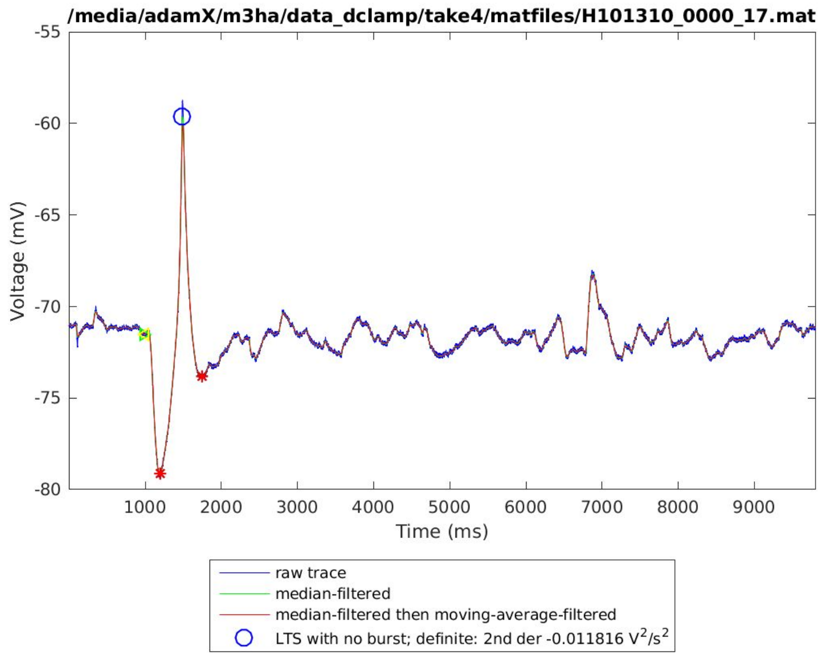
Peak Class #4 - 'LTS with no burst; contentious':



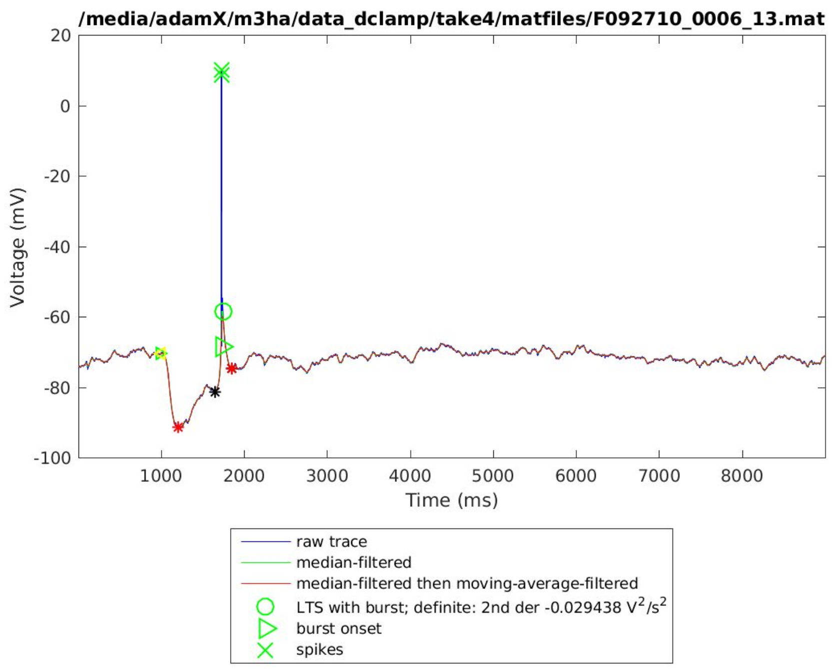
Peak Class #5 - 'LTS with burst; contentious':



Peak Class #6 - 'LTS with no burst; definite':

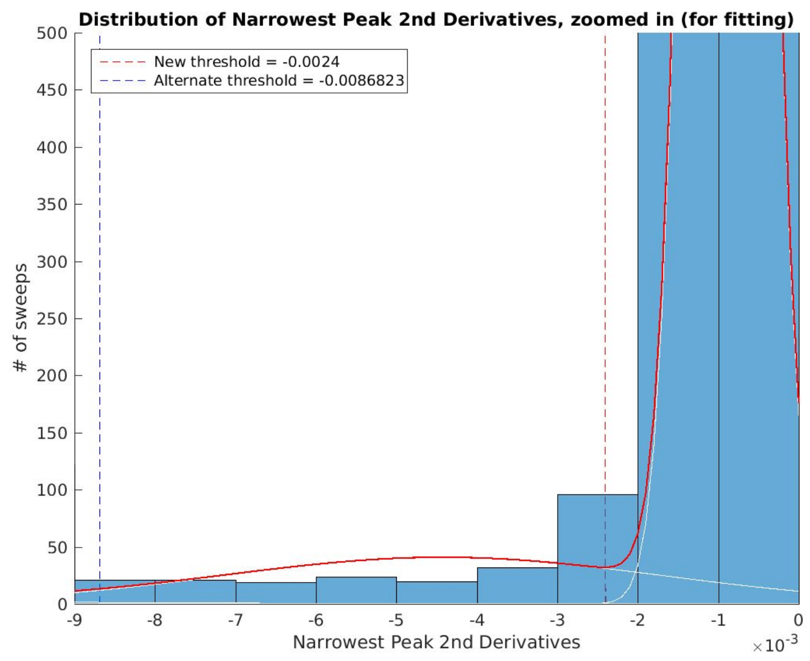


Peak Class #7 - 'LTS with burst; definite':

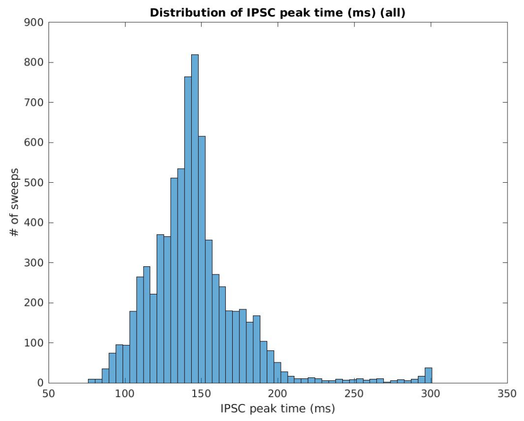


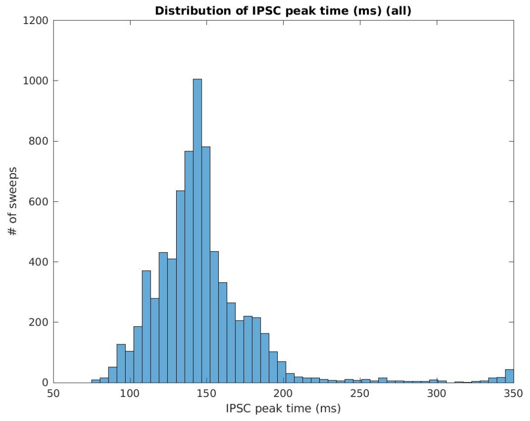
- The newest algorithmic change had an effect in parts of these sets: “Missed_LTS_by_order,” “Noise_in_trace,” “Spontaneous_spike_that_did_not_fire,” “Small_LTS_could_be_noise”, “Noisy_recording,” “Spontaneous_LTSs_or_bursts”

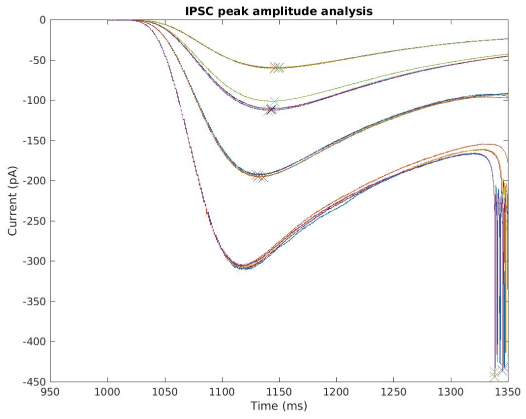
- Changed alternative threshold from … to … to increase “gray area”



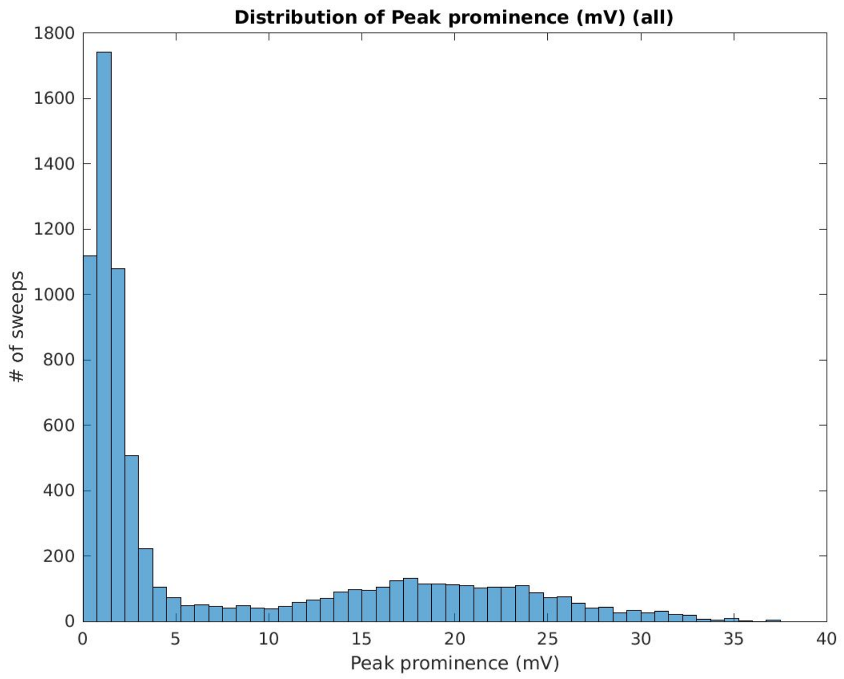
- Attempted to make the IPSC peak window less arbitrary because some peak times were at the upper boundary of the window. However, this was not helpful, as some LTSs influenced the current trace after 1300 ms







- Save the prominence of the narrowest peak as a new statistic **peakprom**


Distribution of Peak prominence (mV) (all)
# of sweeps
Peak prominence (mV)
0
200
400
600
800
1000
1200
1400
1600
1800
0
5
10
15
20
25
30
35
40

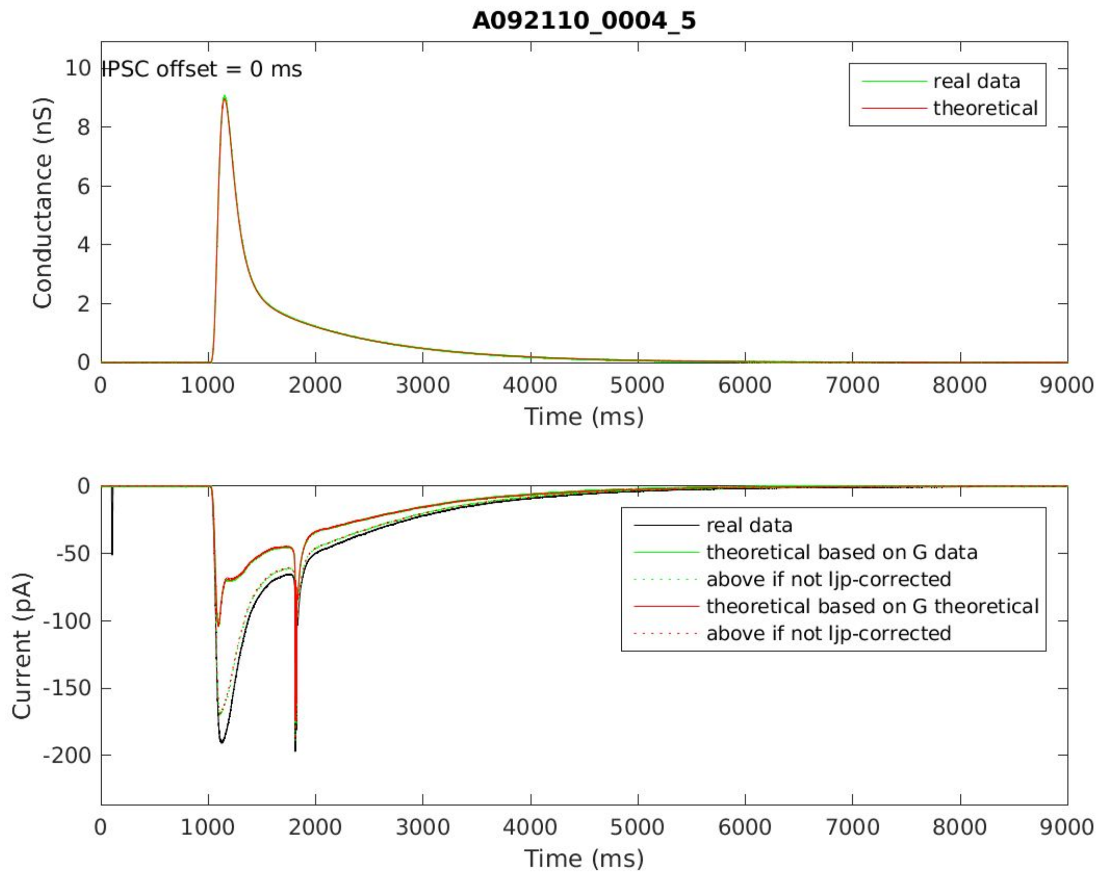
<u>Plan for next week</u>

- Data Analysis - current/conductance traces:
    a. Waiting for John's input on the discrepancies
- Data Analysis - voltage traces:
    a. Recalculate LTS threshold with spontaneous spikes taken out of the histogram (for all fitmodes). Rationale: We are trying to differentiate *LTS* from *noise*, so we don't care about where a spontaneous spike falls in a histogram
    b. Go through the set "**Not_LTS_by_prom**." Is there a better way of measuring **maximum noise**?
    c. Fix the condition "**Spontaneous_spikes_in_baseline**"
    d. Decide how to classify "**Noisy_recording**" (to be taken out of fitting) vs "**Noise_in_trace**" (to be overridden and included in fitting)
    e. Do we still take out **Noisy_recording_fixed** from the fitting even though it's fixed? Should we go through the data and look for other undetected noisy recordings?
    f. Compute new histograms, thresholds & bargraphs
    g. Rerun dclampDataExtractor.m; find all traces with altered LTS onset times and reclassify; find all traces with altered spikes per peak and reclassify; find all "gray area traces" and reclassify. Compare with old4
    h. Look through "gray area traces," reclassify
    i. Reexamine all problematic traces; decide what to do with the remaining false negatives/positives
    j. Analyze the correlation between the burst probability, spikes per peak, spikes per burst, etc. and the slope or other features of an LTS. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
    k. OPTIONAL: Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly
- Fitting:
    a. Instead of calculating the input resistance and using it, fit the model to the current injection responses to extract parameters relevant to the input resistance
    b. Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time; Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data
    c. Fit the model to find the best parameters that will be used in network simulations
- Johnston & Wu:
    a. Finish Ch 2 Problems
    b. Discuss Ch 2
    c. Read Ch 3

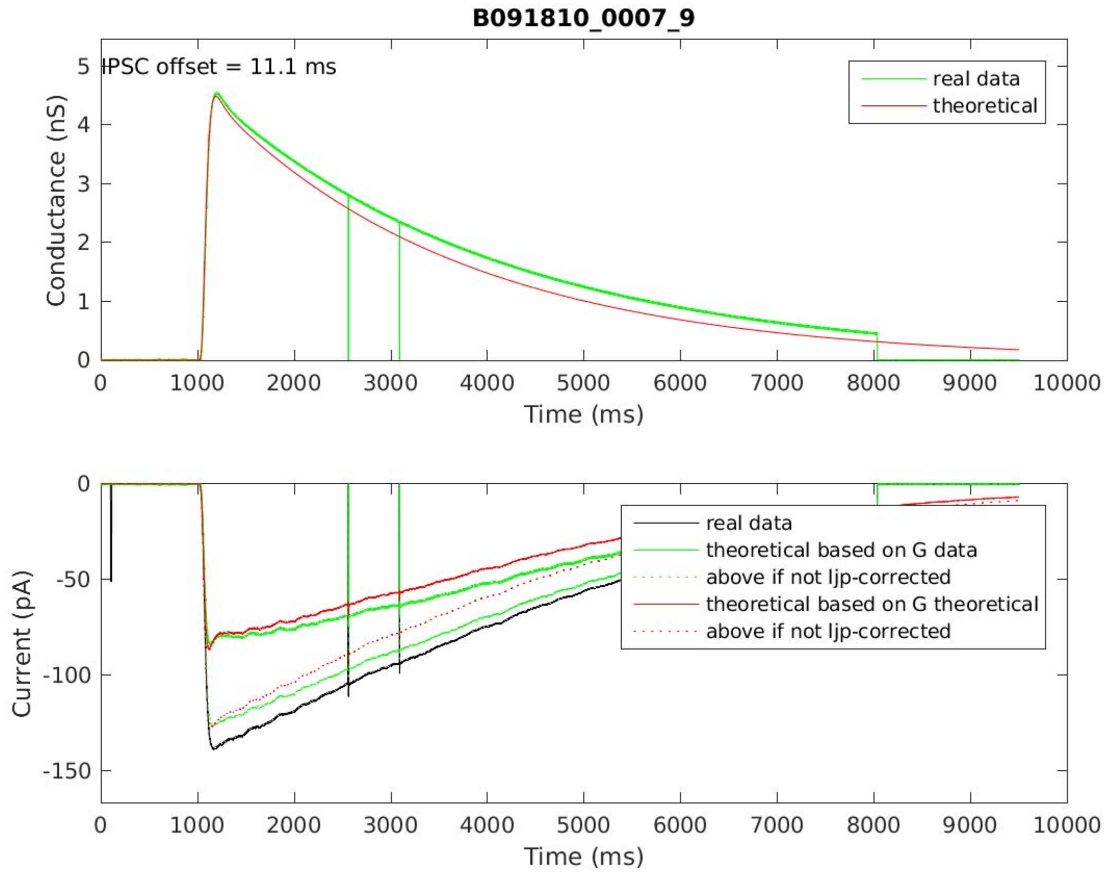
9/4/2016~9/12/2016

<u>Dynamic clamp data analysis (continued)</u>

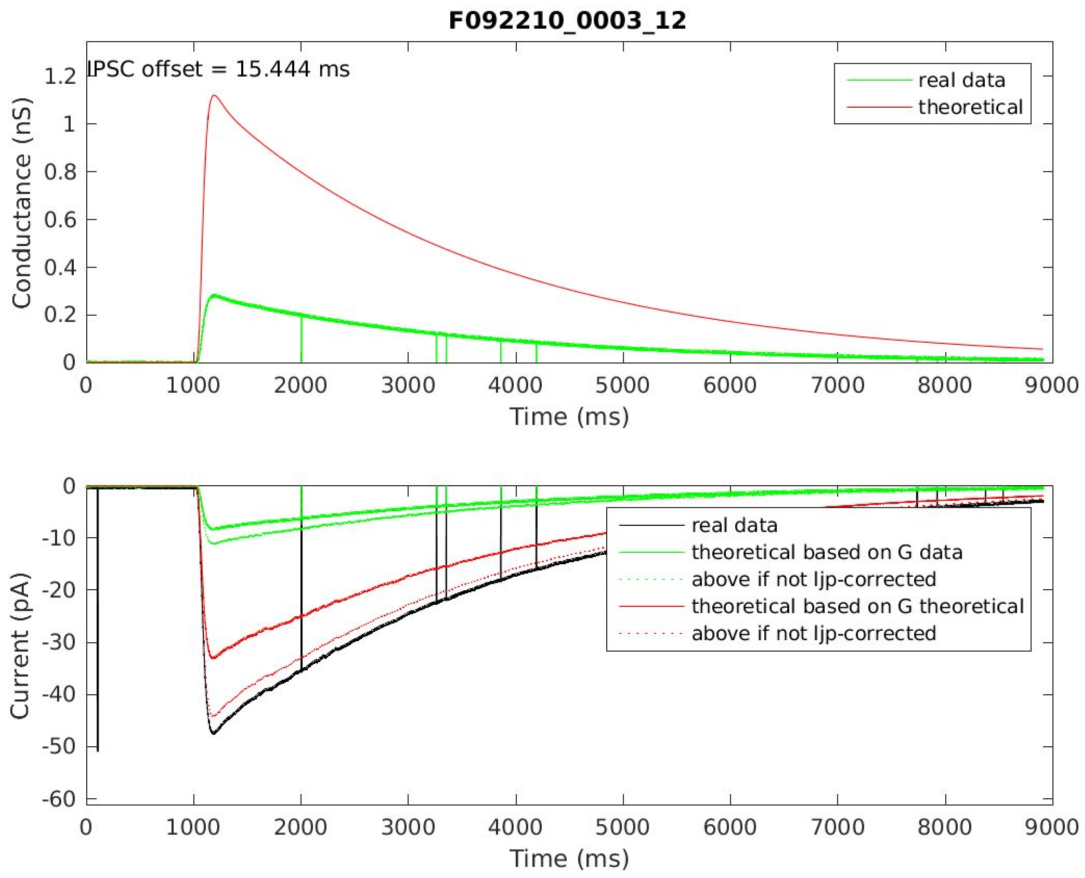
- Compared recorded conductance & current traces with the corresponding theoretical traces
    - The conductance curve recorded in the abf file (scaling-factor-corrected) (green) vs. The conductance curve that was supposed to be applied, based on the pharmacological condition & conductance amplitude scaling that Christine associated with the file (red)
    - This showed discrepancies for some conductance curves but not others, as previously found. However, the peaks do seem to mostly match, and a histogram of the peak differences (which I labeled IPSC offset) on p. 6 shows that there might not be a real offset after all (you were right).
    - When I tried to investigate further whether the current trace deduced from the recorded or theoretical conductance curve (using I = -G(V - E_rev), green and red lines, respectively) matched up with the recorded current curve (black) better, I found that neither matched well!
    - One possibility for this discrepancy is that Christine didn't account for the liquid junction potential when she used E_rev = -105 mV to calculate the current in dynamic clamp. Therefore, I've plotted the same deduced curves (dotted line) using E_rev = -115 mV, and indeed, the current trace deduced from the conductance curve gets a lot closer!
    - However, in many cases, they still don't match up perfectly. So I have the following questions:
    - 1. Why do you think there is still a discrepancy? I've noted the arbitrary scaling factor detected under each figure (The original green trace on the top and black trace on the bottom in the .abf file would be multiplied by this number), but it doesn't seem to explain the discrepancy. What other details might have gone wrong?
    - 2. Most of the times, the red dotted line is closer to the black line than the green dotted line is. However, it's reversed in some cases, such as p.2. It seems like the recorded conductance trace isn't very consistent with the theoretical value. How can we be sure that the recorded current trace is more accurate?
    - 3. In the end, what should we use in the simulations? Should we just inject the same currents as those recorded from PClamp?



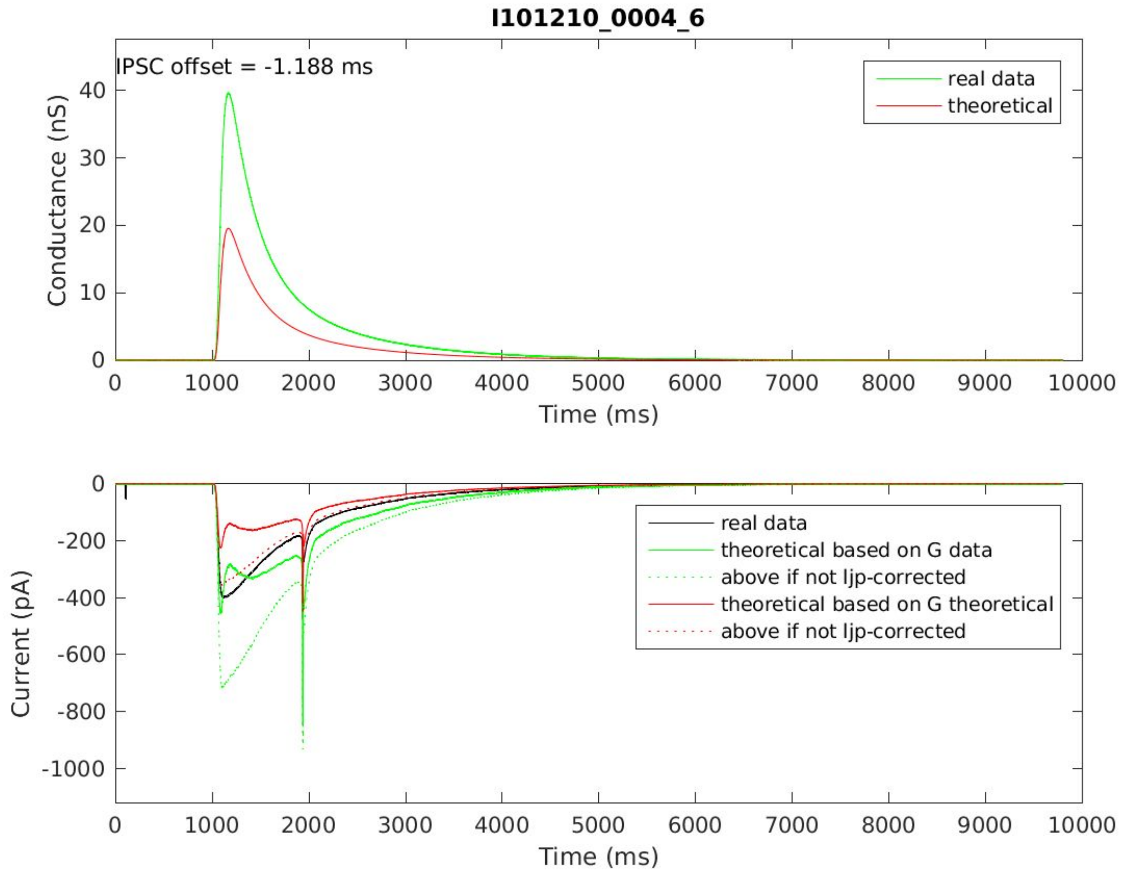
Arbitrary scaling factor in .abf file = **5**



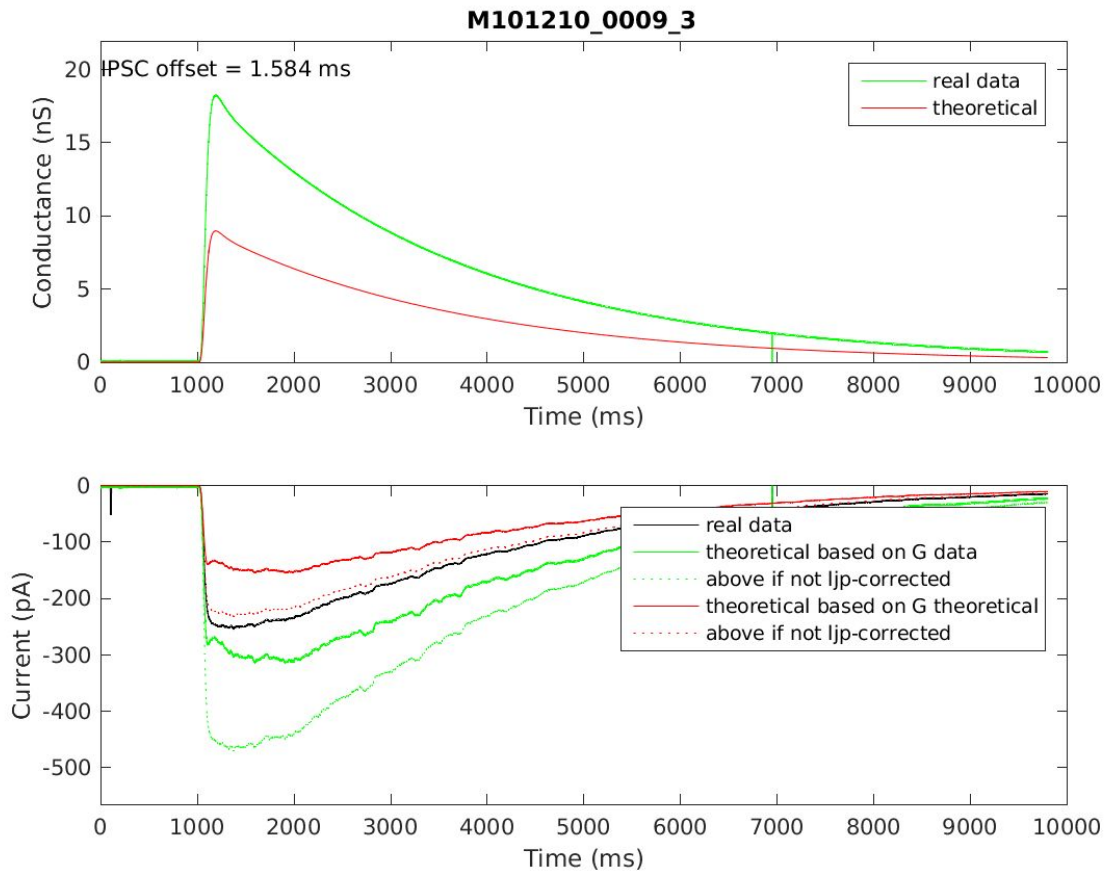
Arbitrary scaling factor in .abf file = **1**



Arbitrary scaling factor in .abf file = **1**

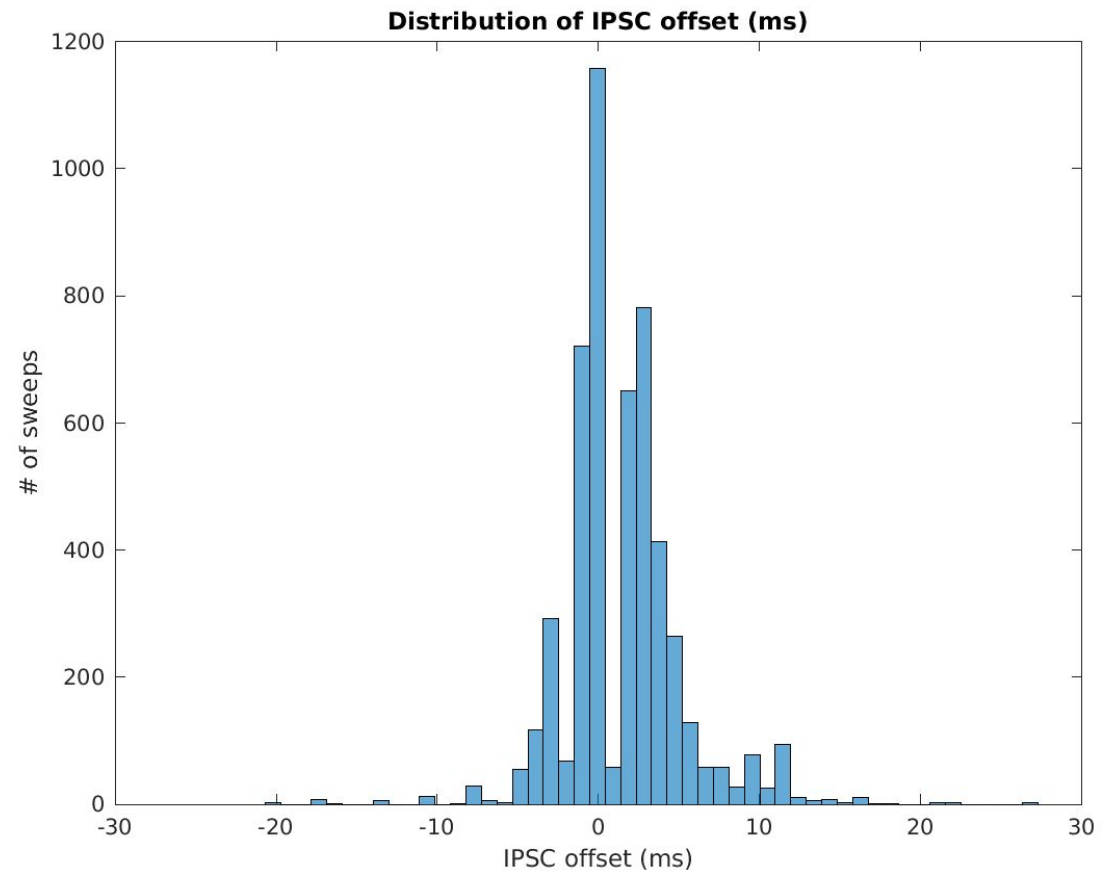


Arbitrary scaling factor in .abf file = **2.5**



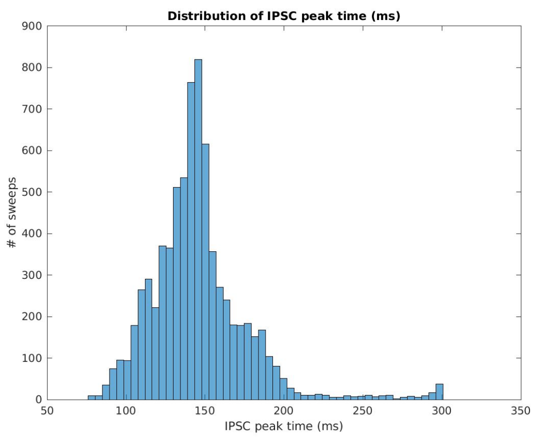
Arbitrary scaling factor in .abf file = **2.5**

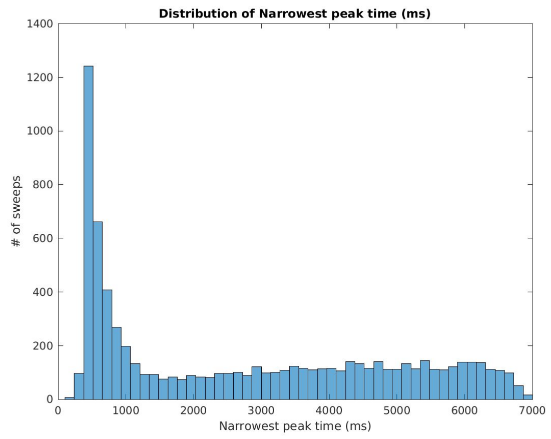
- Compared conductance peak with theoretical peak to find IPSC offset

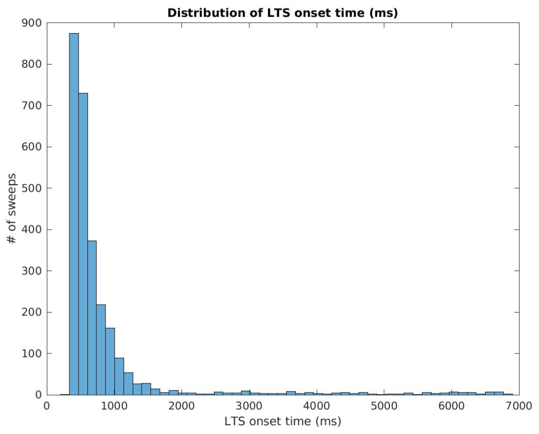


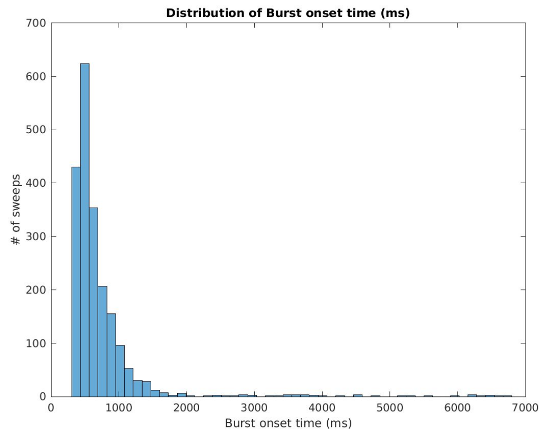
- Recorded the reason for some abf files to be declared broken
    - **Was**: broken_files = {'A100110_0015', 'B100110_0000', 'E100110_0001', 'G091810_0000', 'F092210_0007'};
    - **Now**: broken_files = {'B100110_0000', …   % Cannot open with abf2load
      'E100110_0001', …   % Only one sweep was recorded
      'G091810_0000'}   % Channels were mixed up for at least one trace
    - 'A100110_0015' and 'F092210_0007' were actually not broken but had problematic traces. These will be classified as problematic later, but need to be processed for completion.
    - **TO DO**: Rerun dclampDataExtractor.m with these 45 traces included

- Removed IPSC offset, rerun dclampDataExtractor.m since onset times will change.

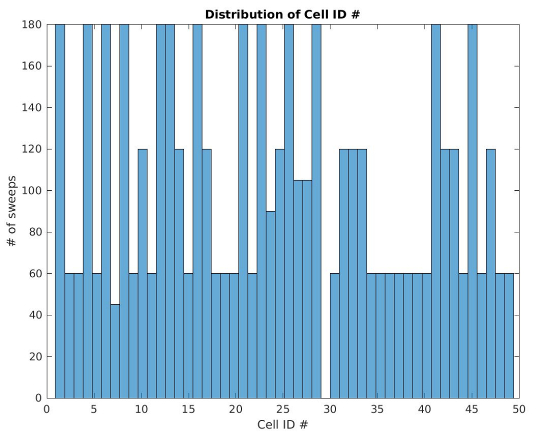


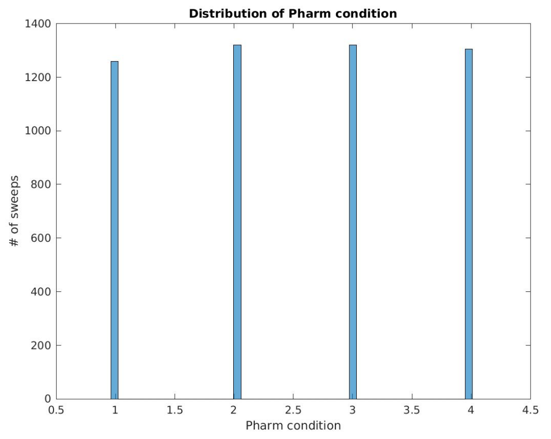


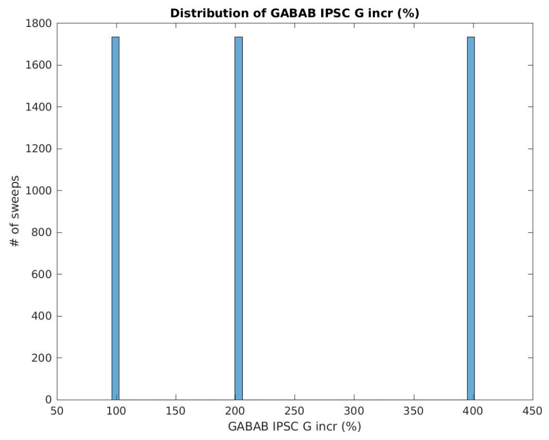


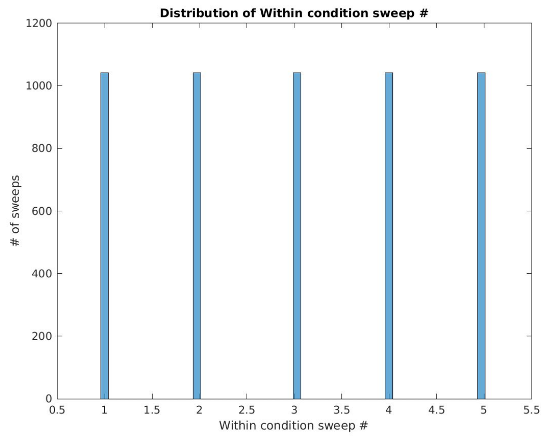


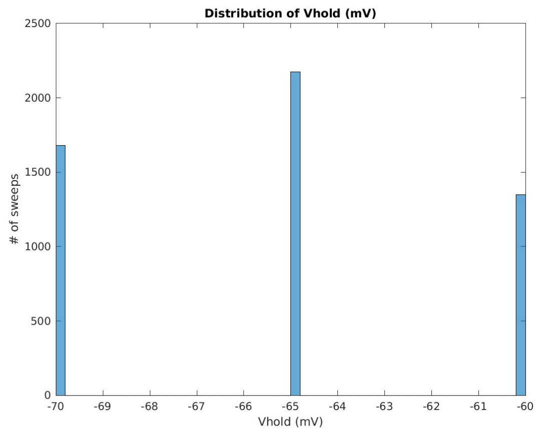
- Restricted to only 100%, 200% & 400% g incr values. Then recalculated threshold & LTS/burst statistics, now with ANOVA for determining whether there are significant changes

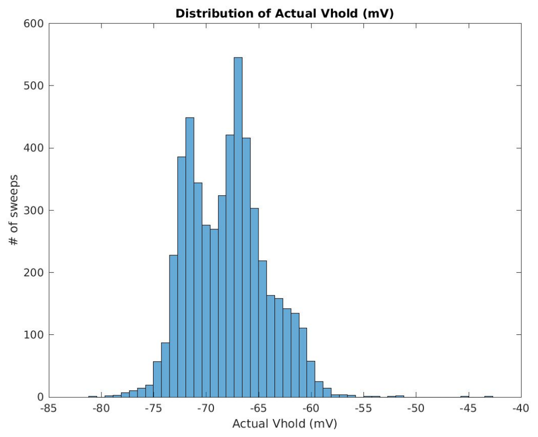


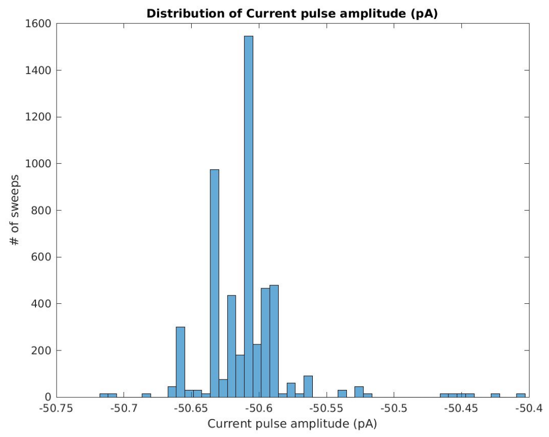


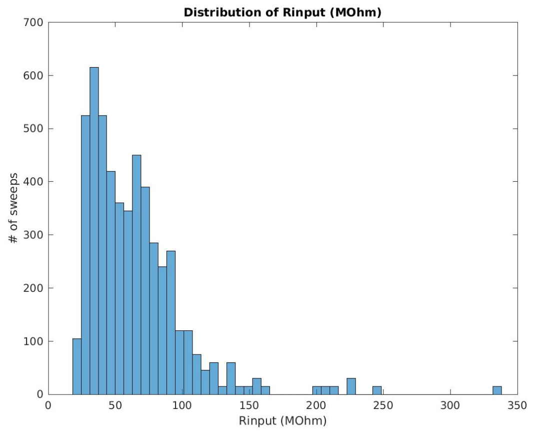

Distribution of GABAB IPSC G incr (%)
# of sweeps
GABAB IPSC G incr (%)

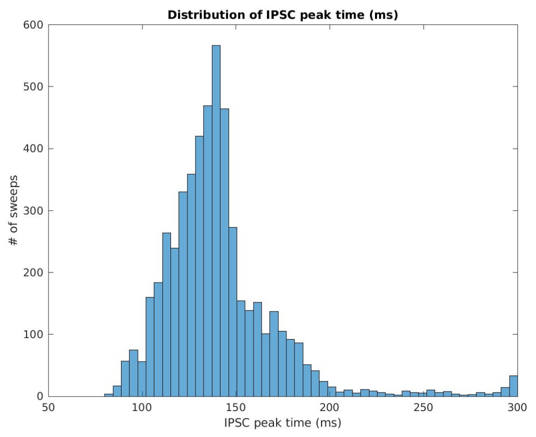

Distribution of Within condition sweep #
# of sweeps
Within condition sweep #

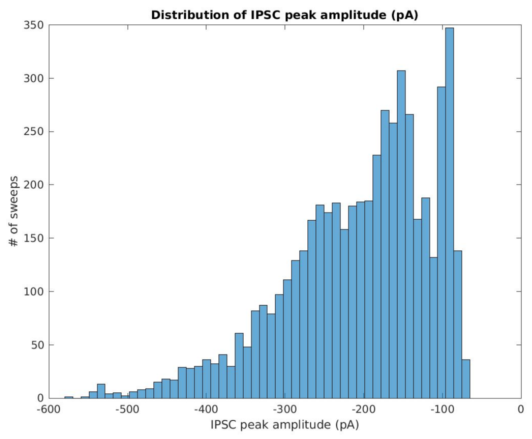

Distribution of Vhold (mV)
# of sweeps
Vhold (mV)

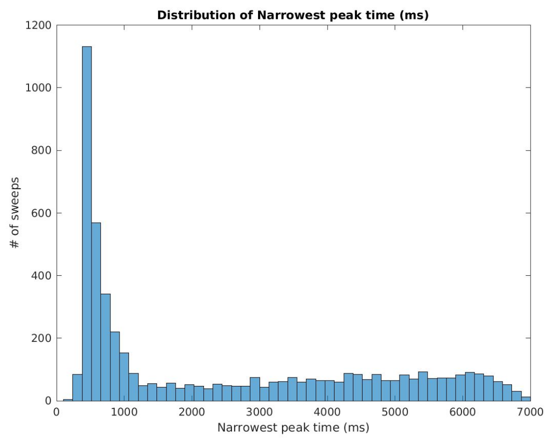

Distribution of Actual Vhold (mV)
# of sweeps
Actual Vhold (mV)

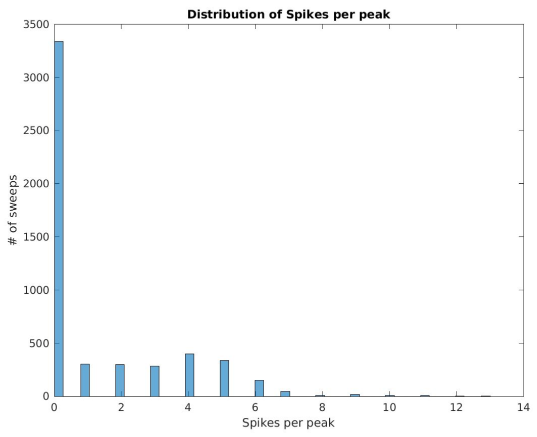

Distribution of Current pulse amplitude (pA)
# of sweeps
Current pulse amplitude (pA)

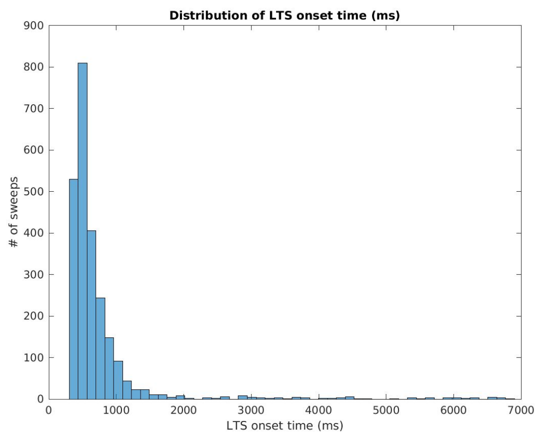

Distribution of Rinput (MOhm)
# of sweeps
Rinput (MOhm)

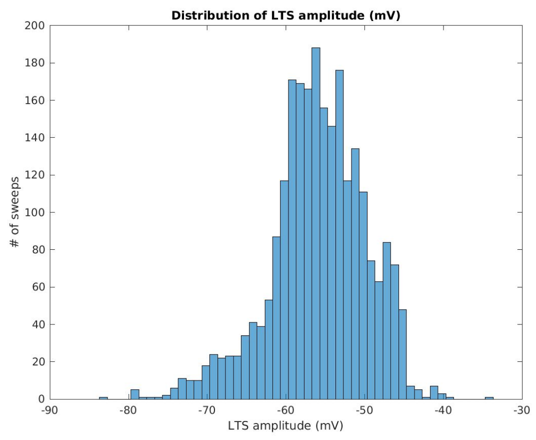

Distribution of IPSC peak time (ms)
# of sweeps
IPSC peak time (ms)

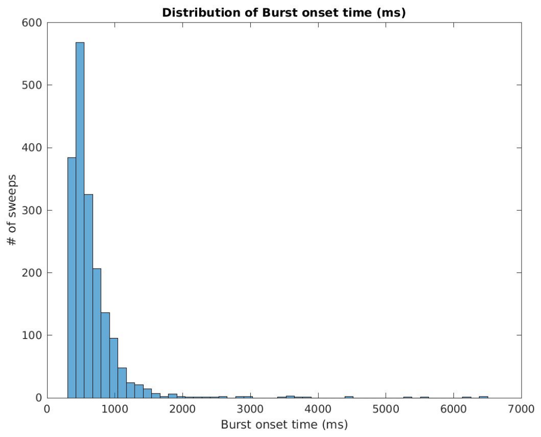

Distribution of IPSC peak amplitude (pA)
# of sweeps
IPSC peak amplitude (pA)

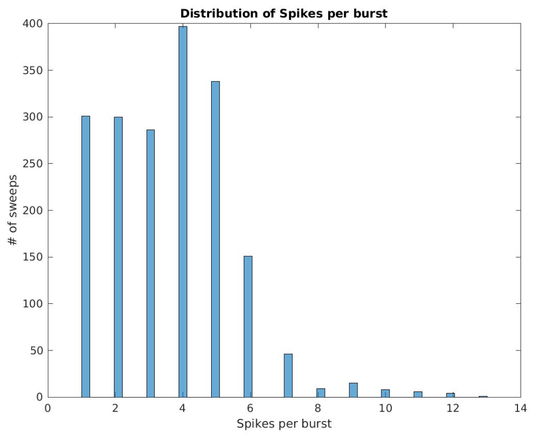

Distribution of Narrowest peak time (ms)
# of sweeps
Narrowest peak time (ms)

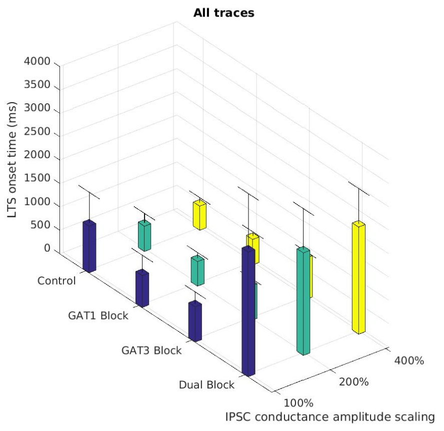

Distribution of Spikes per peak
# of sweeps
Spikes per peak

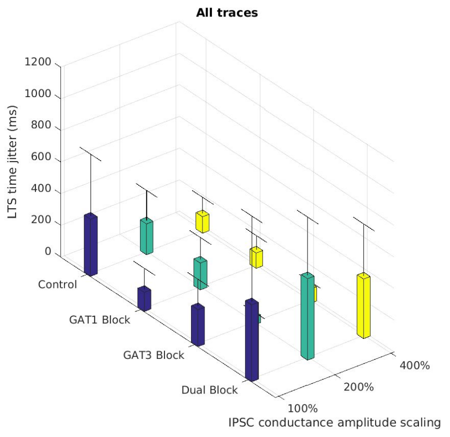

Distribution of LTS onset time (ms)
# of sweeps
LTS onset time (ms)

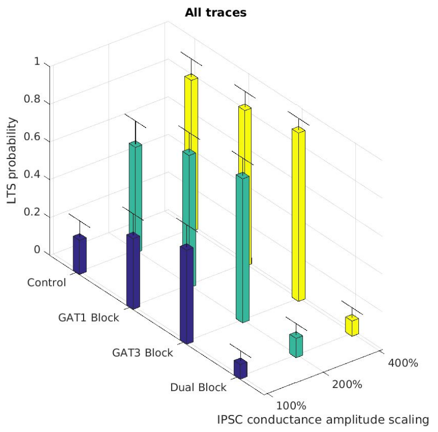

Distribution of LTS amplitude (mV)
# of sweeps
LTS amplitude (mV)

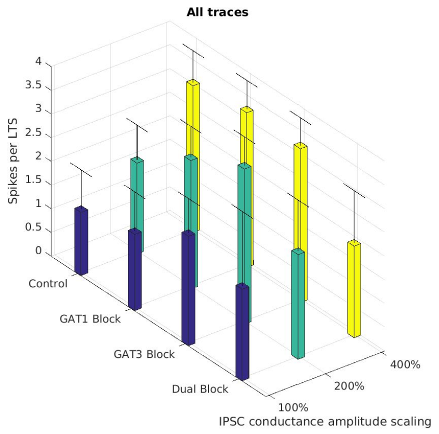

Distribution of Burst onset time (ms)
# of sweeps
Burst onset time (ms)
0
100
200
300
400
500
600
1000
2000
3000
4000
5000
6000
7000

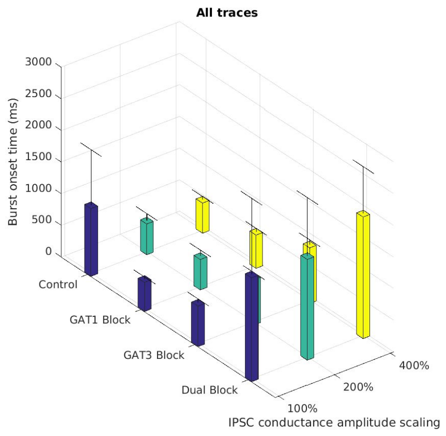

Distribution of Spikes per burst
# of sweeps
Spikes per burst
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14

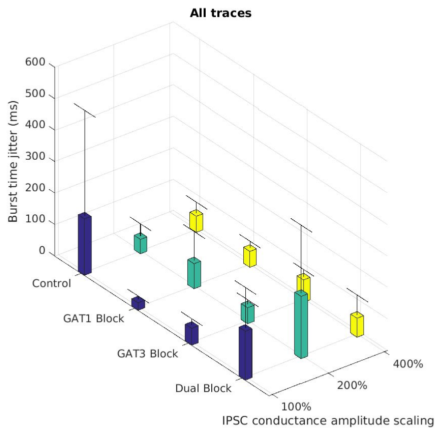

All traces
LTS onset time (ms)
0
500
1000
1500
2000
2500
3000
3500
4000
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

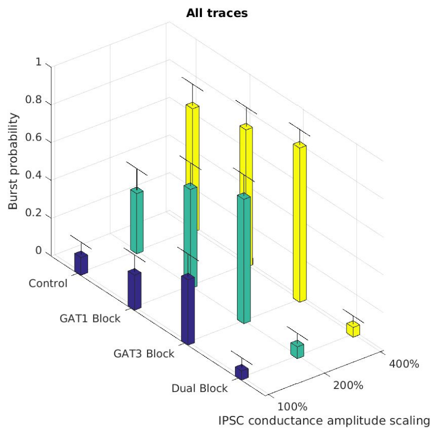

All traces
LTS time jitter (ms)
0
200
400
600
800
1000
1200
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

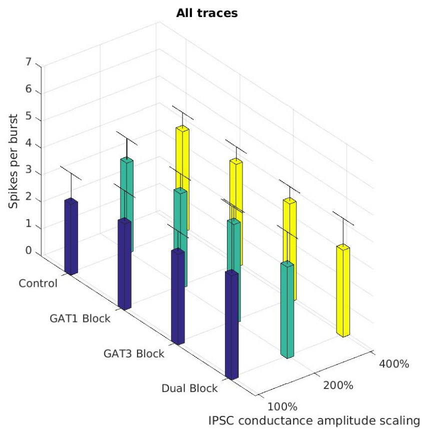

All traces
LTS probability
0
0.2
0.4
0.6
0.8
1
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

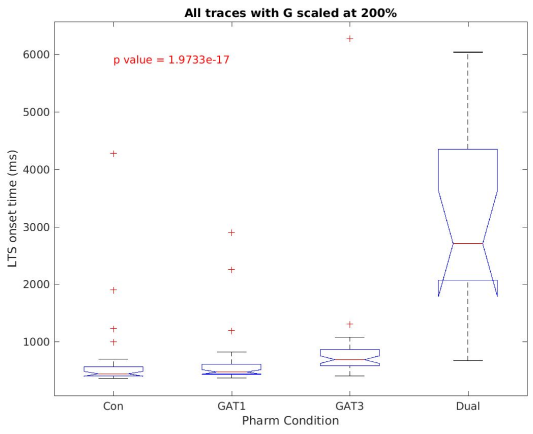

All traces
Spikes per LTS
0
0.5
1
1.5
2
2.5
3
3.5
4
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

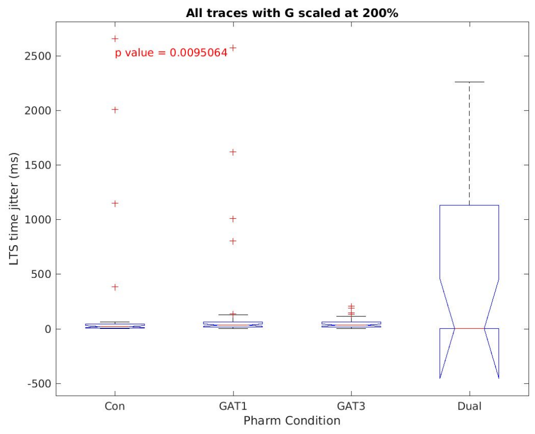

All traces
Burst onset time (ms)
3000
2500
2000
1500
1000
500
0
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling


All traces
Burst time jitter (ms)
600
500
400
300
200
100
0
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

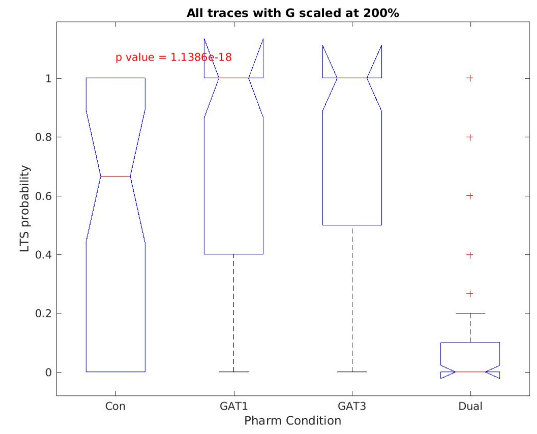

All traces
Burst probability
1
0.8
0.6
0.4
0.2
0
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

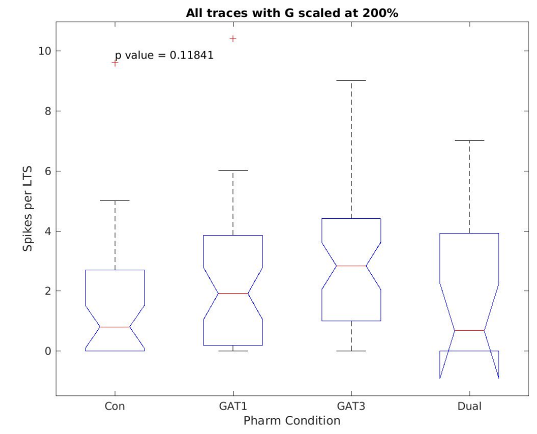

All traces
Spikes per burst
7
6
5
4
3
2
1
0
Control
GAT1 Block
GAT3 Block
Dual Block
100%
200%
400%
IPSC conductance amplitude scaling

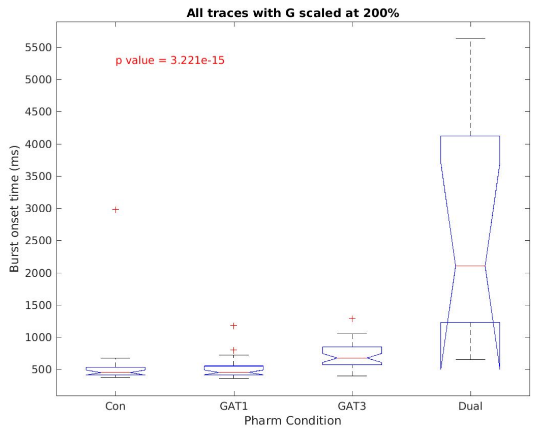

All traces with G scaled at 200%
p value = 1.9733e-17
LTS onset time (ms)
6000
5000
4000
3000
2000
1000
Con
GAT1
GAT3
Dual
Pharm Condition

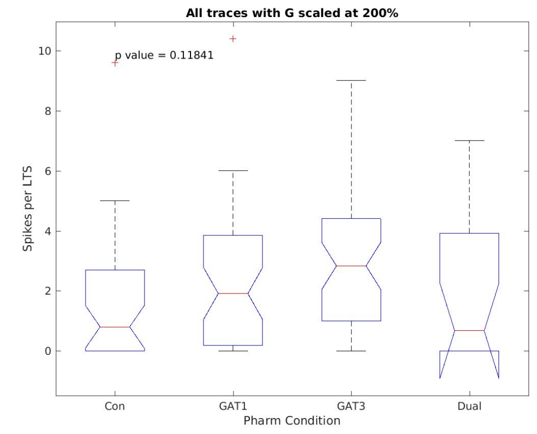

All traces with G scaled at 200%
p value = 0.0095064
LTS time jitter (ms)
2500
2000
1500
1000
500
0
-500
Con
GAT1
GAT3
Dual
Pharm Condition

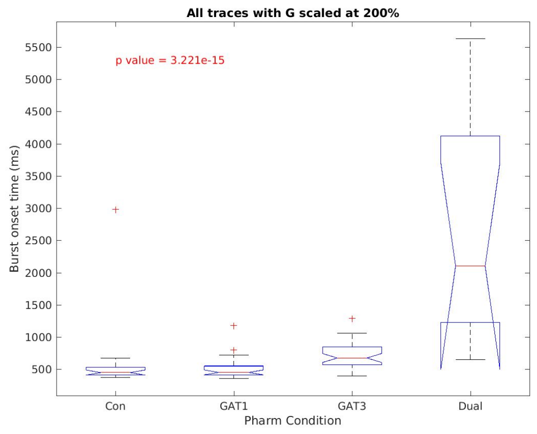
20160911

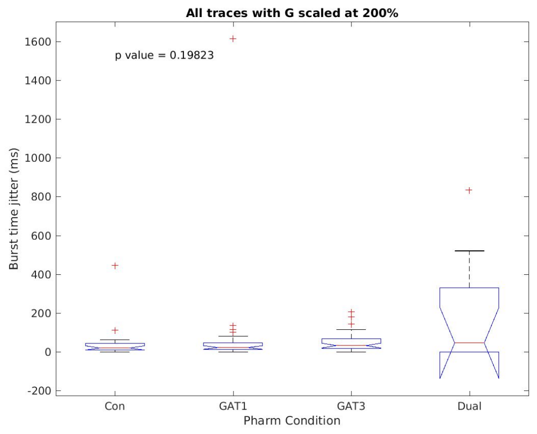

All traces with G scaled at 200%
p value = 1.1386e-18
LTS probability
0
0.2
0.4
0.6
0.8
1
Con
GAT1
GAT3
Dual
Pharm Condition

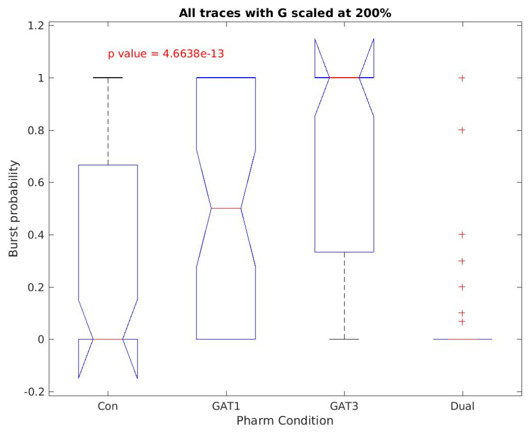

All traces with G scaled at 200%
p value = 0.11841
Spikes per LTS
0
2
4
6
8
10
Con
GAT1
GAT3
Dual
Pharm Condition

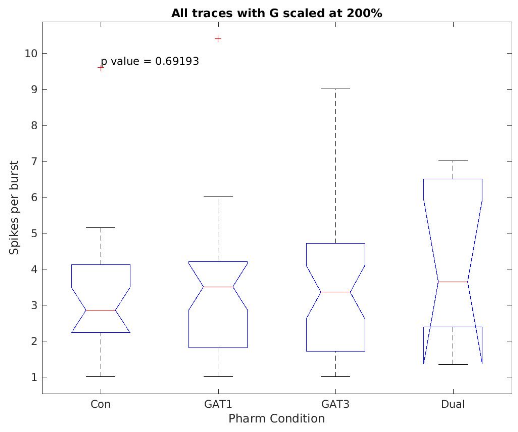

All traces with G scaled at 200%
p value = 3.221e-15
Burst onset time (ms)
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
Con
GAT1
GAT3
Dual
Pharm Condition


All traces with G scaled at 200%
p value = 0.11841
Spikes per LTS
0
2
4
6
8
10
Con
GAT1
GAT3
Dual
Pharm Condition

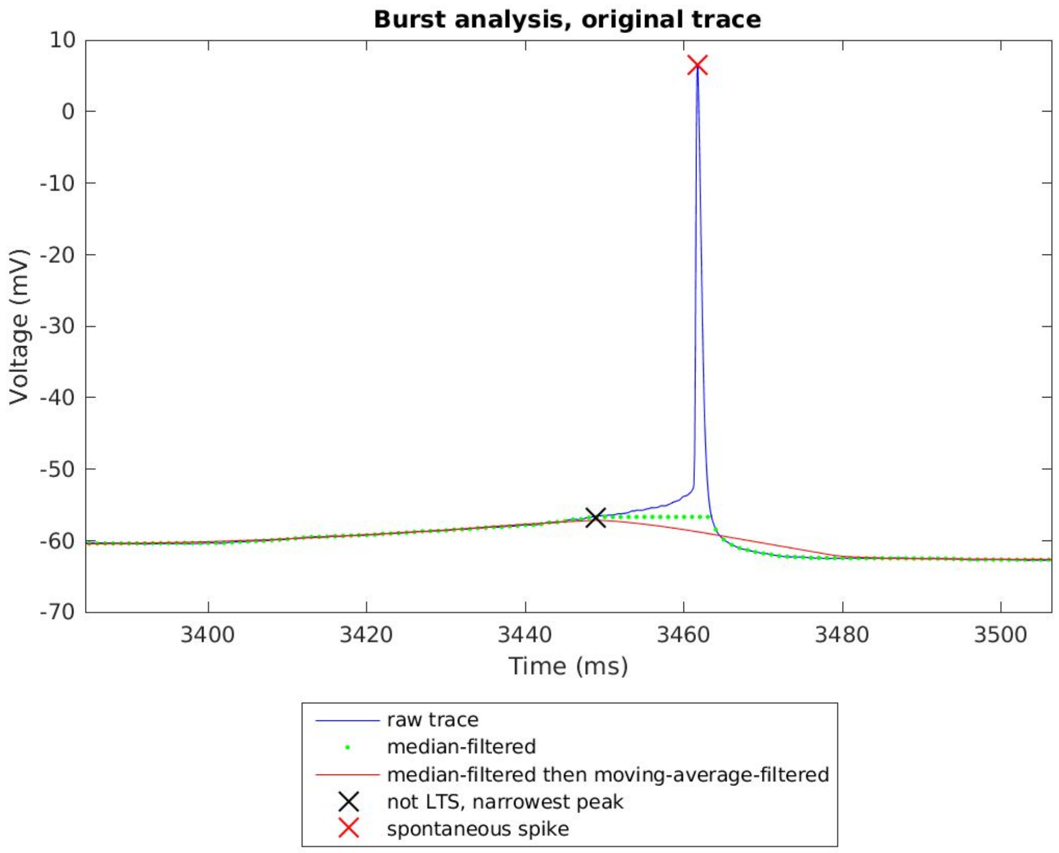

All traces with G scaled at 200%
p value = 3.221e-15
Burst onset time (ms)
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
Con
GAT1
GAT3
Dual
Pharm Condition


All traces with G scaled at 200%
p value = 0.19823
Burst time jitter (ms)
-200
0
200
400
600
800
1000
1200
1400
1600
Con
GAT1
GAT3
Dual
Pharm Condition





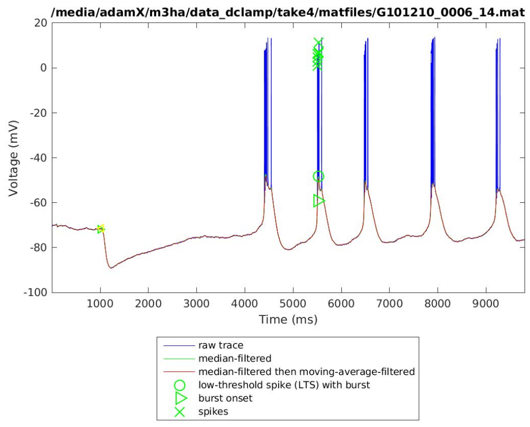
- Spontaneous spikes that were correctly classified

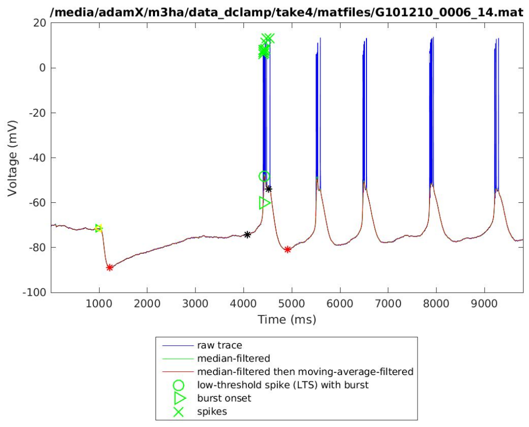


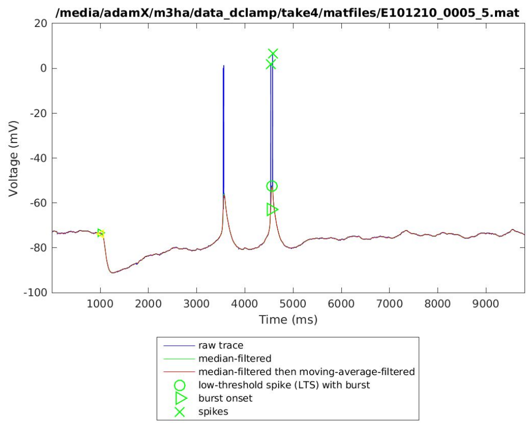
- Revisions to data extraction:
    - Fixed traces belonging to “**Not_the_first_LTS**” (**2** traces) by finding the ***first*** LTS peak that crosses the 2nd derivative threshold instead of the narrowest overall peak

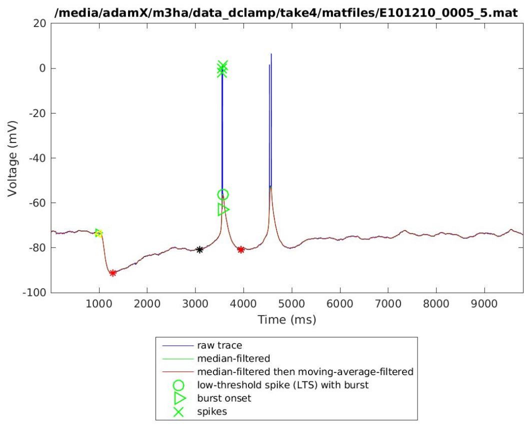
Before:

After:





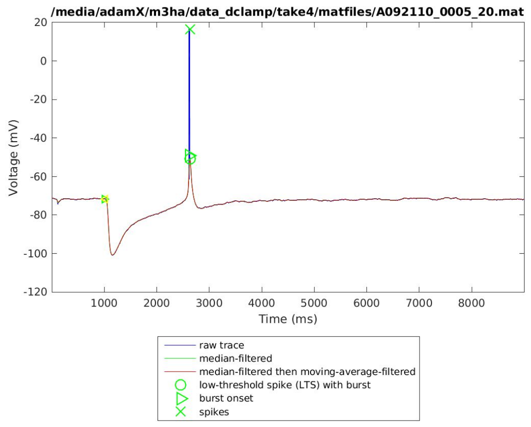




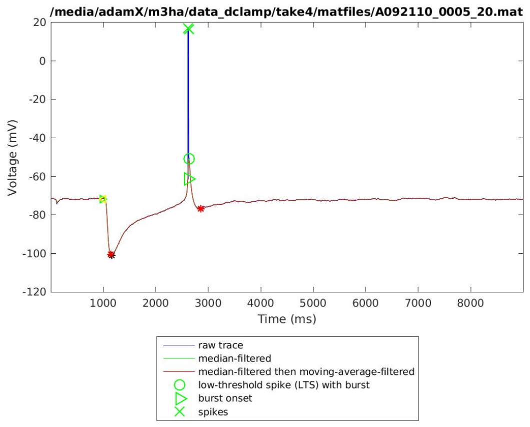
- Fixed an error in finding peak boundaries. The function **fliplr** was used instead of **flipud**.
- Changed peak boundary detection into two steps: Using no maximum noise prominence (maxnoiseprom1 = 0 mV) when deciding whether a peak is a spontaneous spike or not, but updating the peak boundary with a maximum noise prominence (maxnoiseprom2) of **2 mV** if an action potential is present, to calculate the correct number of **spikes per peak**

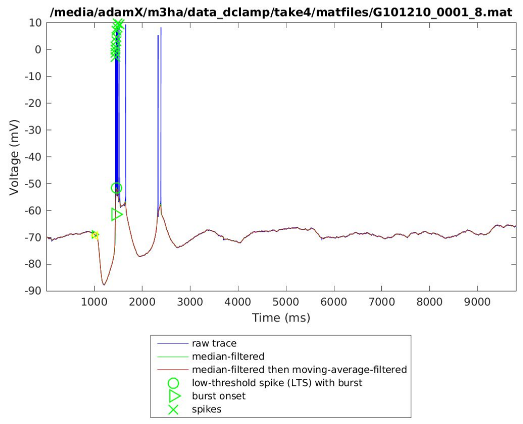
- Consequently, traces belonging to "**Spikes_per_burst_incorrect**" (**23** traces) were fixed:

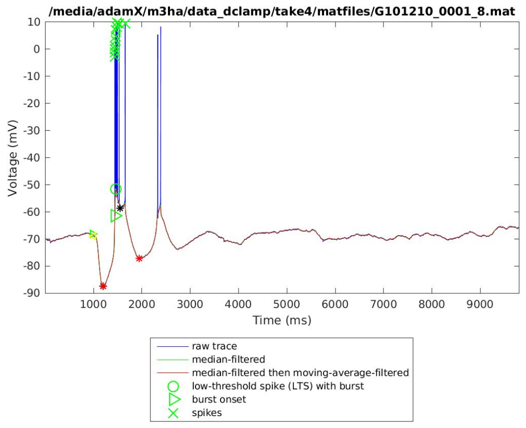
Before:



After:

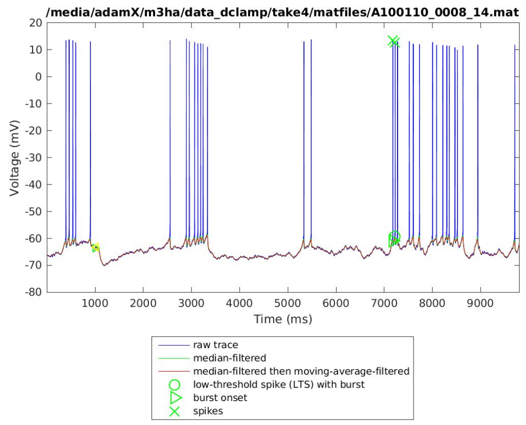




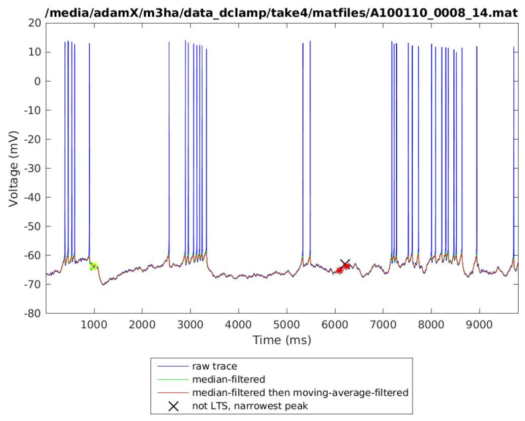


- Traces belonging to "**Spontaneous_spikes_too_close_together**" (**22** traces) were also fixed:

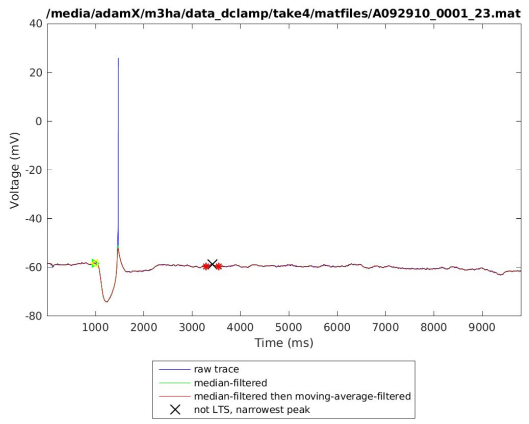
Before:

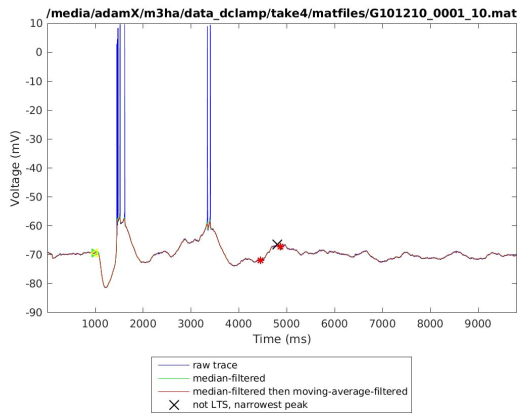


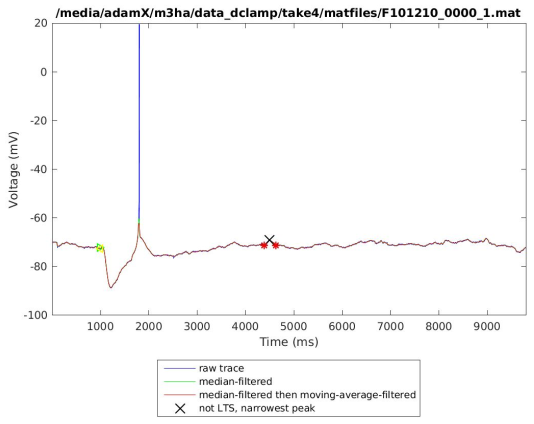
After:

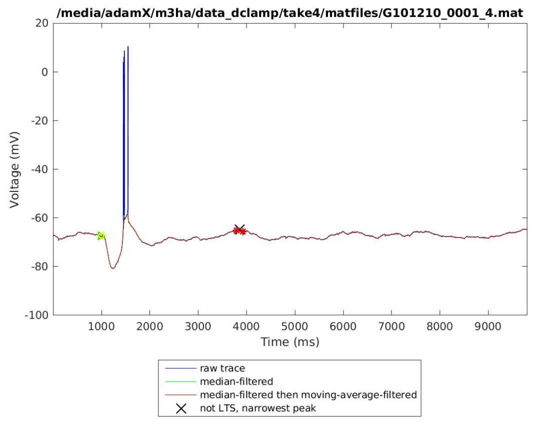


- False negatives remaining:
  - “**Missed_LTS_shape_problem**” (**4** traces)

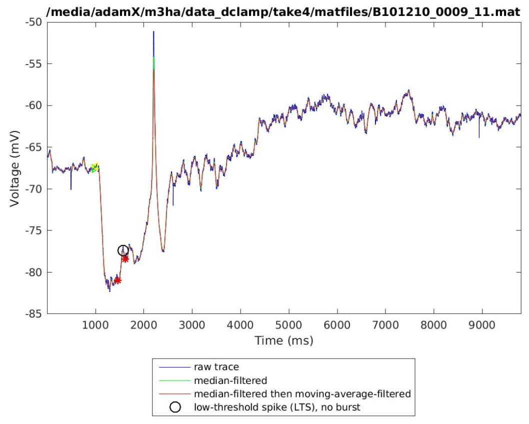


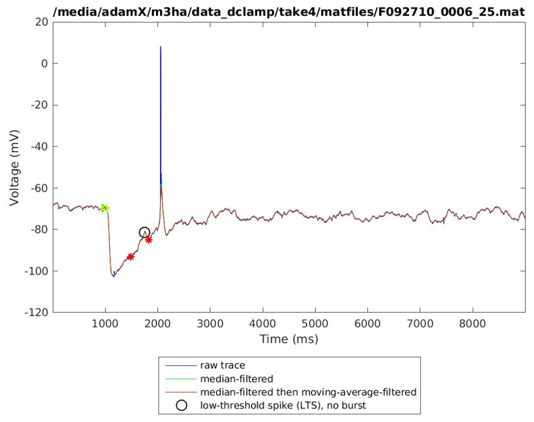


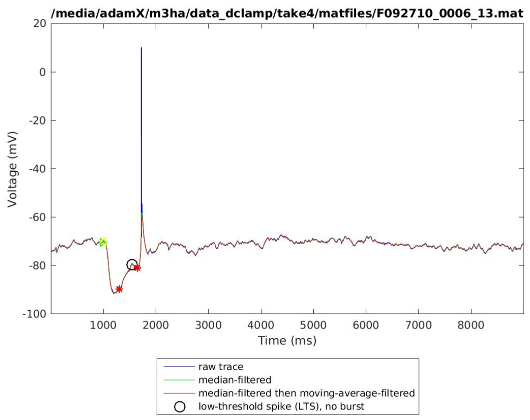




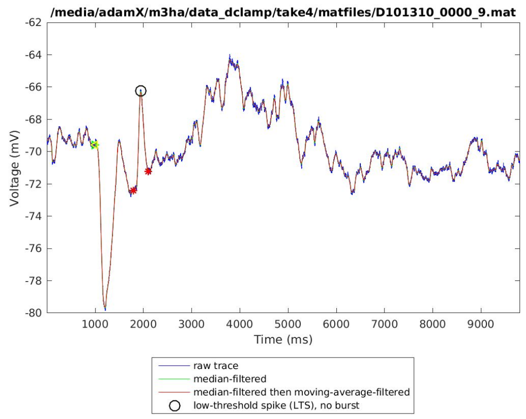
  - “**Missed_LTS_order_problem**” (**3** traces)

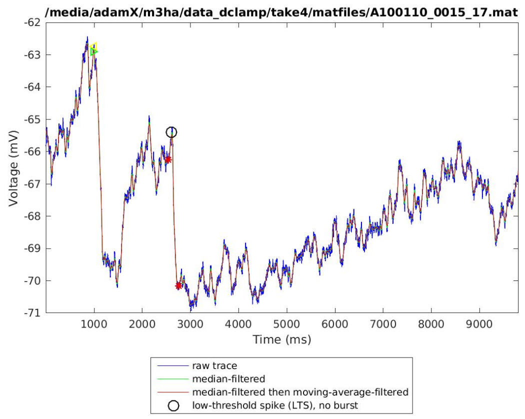




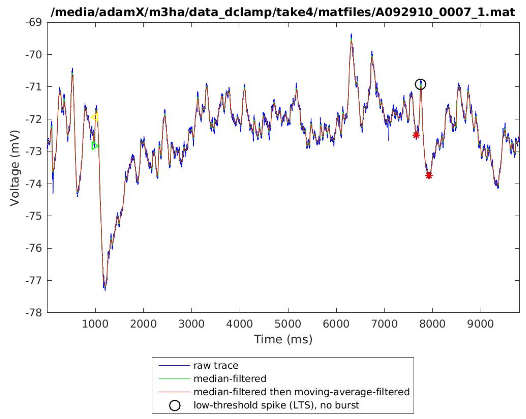


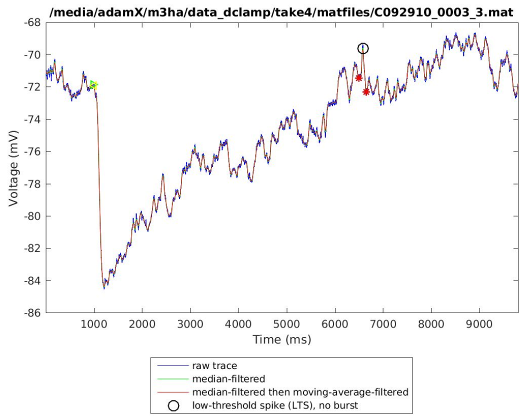
- False positives remaining:
  - “**Noisy_recording**” (**18** traces)



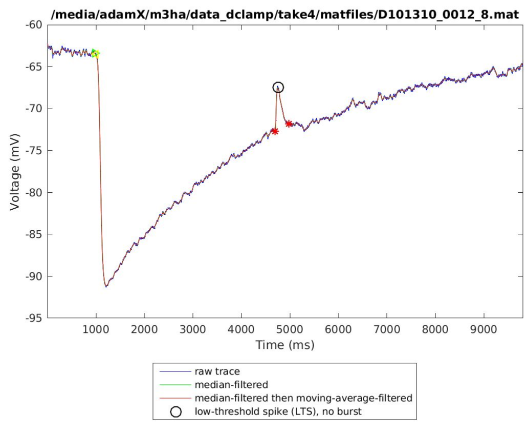


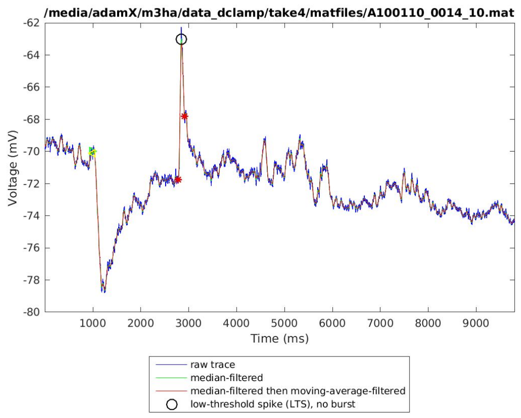
  - “**Looks_like_noise**” (**59** traces)



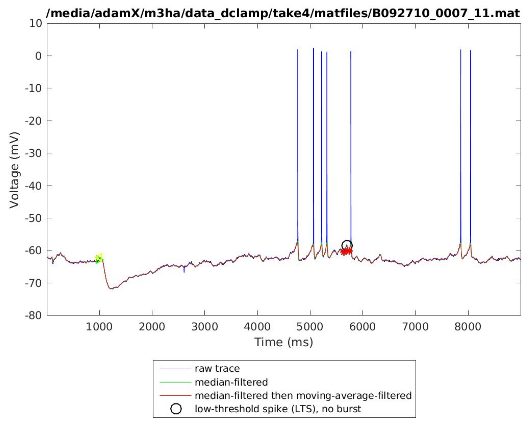


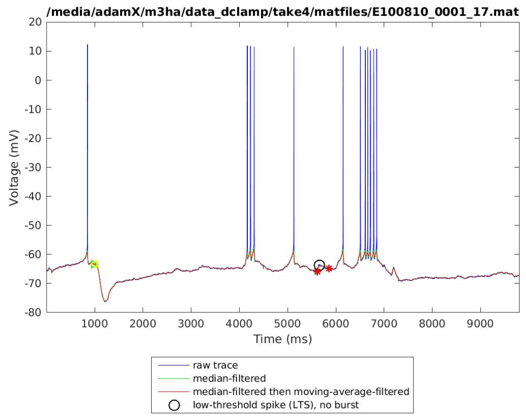
- “**Spontaneous_LTSs_or_bursts**” (**44** traces)



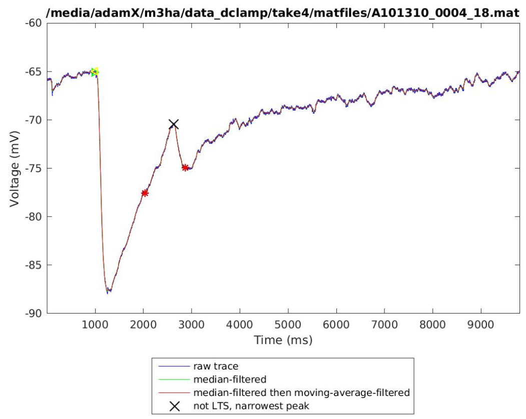


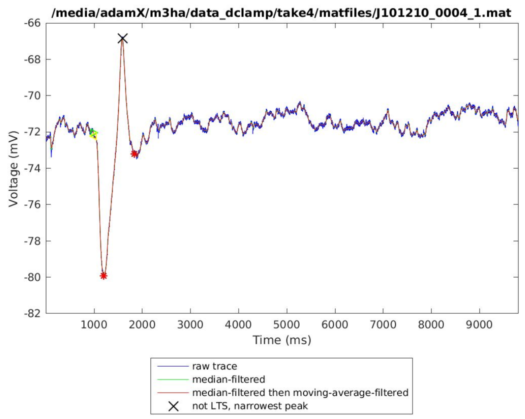
- “**Spontaneous_spikes_that_did_not_fire**” (**20** traces)



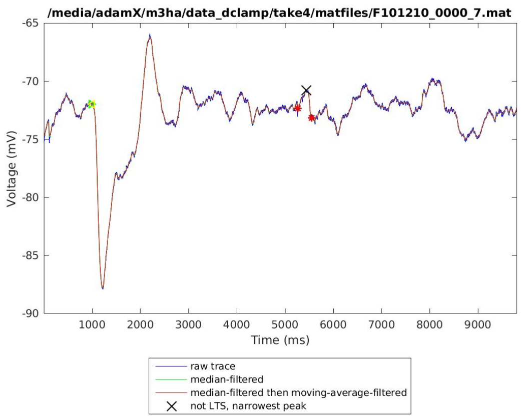


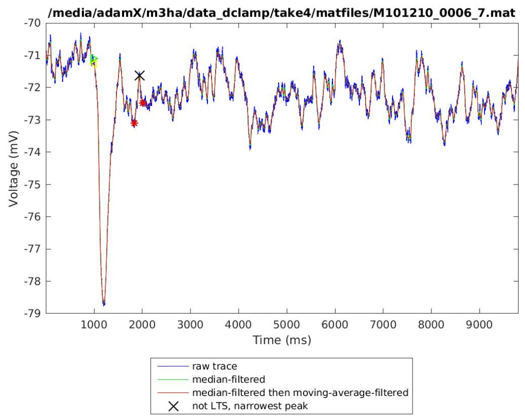
- Possible false negatives remaining:
  - “**Looks_like_LTS**” (**84** traces)



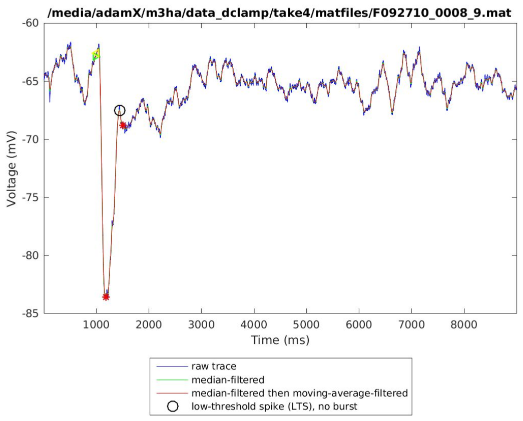


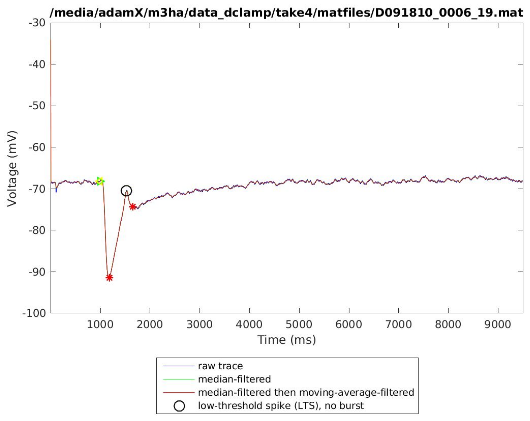
- “**Looks_like_missed_LTS**” (**7** traces)





- Possible false positives remaining:
  - “**Very_small_LTS**” (**43** traces)





Plan for next week

- Decide what to do with false negatives/positives. Try a positive 2nd derivative threshold for spontaneous LTSs (see C092810_0003_10)? Take out specific cell-pharm-g incr sets?
- Update 2nd derivative threshold based on selected data
- Decide what to do with the remaining false negatives/positives
- Recalculate LTS & burst statistics, now with ANOVA for determining whether there are significant changes
- Fit with remaining data
- Analyze the correlation between the burst probability, spikes per peak, spikes per burst, etc. and the slope or other features of an LTS. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
- OPTIONAL: Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly

- Instead of calculating the input resistance and using it, fit the model to the current injection responses to extract parameters relevant to the input resistance
- Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time
- Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data

- Johnston & Wu Ch 2 Problems

7/14/2016~8/30/2016

Dynamic clamp data analysis

- How the analysis was done:
    - 4 experiment series:
        - {'091710'; '091810'; '092110'; '092210'; '092710'; '092810'; '092910'} have G incr = **[25 50 100 200 400]**
        - {'092910_7.5'; '100110'; '100810'} have G incr = **[50 100 200 400]**
        - {'101210'} have G incr = **[100 200 400]**
        - {'101310'} have G incr = **[100 200 400 800]**
    - Problematic abf files:
        - **broken_files** = {'A100110_0015', 'B100110_0000', 'E100110_0001', 'G091810_0000', 'F092210_0007'};
          TO DO: Record the reason for some abf files to be declared broken
        - **flipped_files** = {'F092210_0000', 'F092210_0001', 'F092210_0002', 'F092210_0003', 'F092210_0004', 'F092210_0005', 'F092210_0006'};
          % The current and conductance channels were flipped
    - Template parameters used for GABAB IPSC conductance waveforms (Note: amp in Christine's thesis was actually for 200 % G incr):
        - amp = **[16.00; 24.00; 8.88; 6.32]**; % (nS)
        - Trise = [52.00; 52.00; 38.63; 39.88]; % (ms)
        - TfallFast = [90.10; 90.10; 273.40; 65.80]; % (ms)
        - TfallSlow = [1073.20; 1073.20; 1022.00; 2600.00]; % (ms)
        - w = [0.952; 0.952; 0.775; 0.629];
    - Fixed parameters used in the experiments
        - nsets = **12**;
          % A maximum of 12 sets per cell (4 pharm x 3 Vhold)
        - nswpspc = **5**;
          % # of sweeps per PVG condition
        - pharm = **[1; 2; 3; 4]**;
          % Possible pharm conditions (1 - Control; 2 - GAT1 Block; 3 - GAT3 Block; 4 - Dual Block)
        - Vhold = **[-50; -55; -60]**;
          % Possible Vhold values (as shown on PClamp, not LJP-corrected)
        - ljp = **-10**;
          % Liquid junction potential used in Christine's experiments (mV)
        - cpwin = **[95 115]**;
          % Window in which the current pulse would lie (ms) (Supposed to be 100-110 ms but there will be offset)

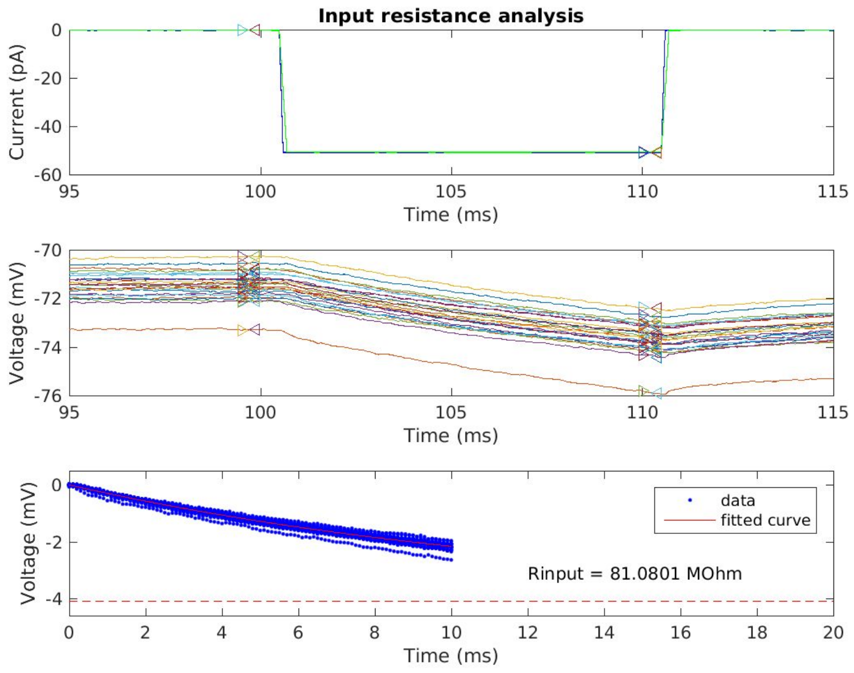
    - cpmid = **105**;
      % Approximate midpoint of the current pulse (ms)
    - ipsctwin = **[1000 1100]**;
      % Window in which IPSC is first applied (ms) (Supposed to be 1000 ms)
    - ipscpwin = **[1000 1300]**;
      % Window in which IPSC reaches peak amplitude (ms) (based on observation)
    - ltswin = **[1000 7960]**;
      % Window in which the low threshold spike would lie (ms). Previously [1000 4500] by Christine; 8000 ms is the approximate time that IPSC is terminated; subtract out 40 ms for discontinuities; 1000 ms will be replaced by ipscpt, the time when IPSC reaches peak amplitude
    - maxswps = **7410**;
      % Maximum number of sweeps recorded
    - maxcells = **49**;
      % Maximum number of neurons recorded
  - Parameters used for data reorganization:
    - mfw1 = **2.5**;
      % width in ms for the median filter for PClamp noise (conductance traces)
    - mfw2 = **10**;
      % width in ms for the median filter for corrupted data (current traces)
    - mfw3 = **30**;
      % width in ms for the median filter for spikes (voltage traces)
    - mafw1 = **5**;
      % width in ms for the moving average filter for finding IPSC offsets
    - mafw2 = **30**;
      % width in ms for the moving average filter for finding narrowest voltage peaks
    - slth = **0.1**;
      % slope threshold for finding IPSC offset
    - mvw = **0.5**;
      % width in ms for calculating mean voltage for input resistance calculations
    - blw = **20**;
      % width in ms for calculating baseline voltage (holding potential)
    - minprom = **1**;
      % minimum LTS prominence in mV
    - maxnoiseprom = **0.1**;
      % maximum noise prominence in mV
    - lts_thr = **-0.0023**;
      % 2nd derivative in V^2/s^2 below which defines an LTS peak

        - sp_thr = **-30**;
          % amplitude threshold in mV for detecting a spike (the highest LTS peak is -34.01 mV)
        - rsims = **1**;
          % resampling interval in ms (1 kHz)
    - Log file:
        - dclampdatalog_take4.csv is a comma separated value file that contains the following info:
            - 'Data filename'
            - 'Cell ID #'
            - 'Pharm condition'
            - 'Vhold (mV)'
            - 'GABAB IPSC G incr (%)'
            - 'Within condition sweep #'
            - 'GABAB IPSC G amp (nS)'
            - 'GABAB IPSC G Trise (ms)'
            - 'GABAB IPSC G TfallFast (ms)'
            - 'GABAB IPSC G TfallSlow (ms)'
            - 'GABAB IPSC G w'
            - 'Current pulse amplitude (pA)'
            - 'Rinput (MOhm)'
            - 'IPSC offset (ms)'
            - 'IPSC peak time (ms)'
            - 'IPSC peak amplitude (pA)'
            - 'Actual Vhold (mV)'
            - 'Narrowest peak time (ms)'
            - 'Narrowest peak 2nd derivative (V^2/s^2)'
            - 'Spikes per peak'
            - 'LTS onset time (ms)'
            - 'LTS amplitude (mV)'
            - 'Burst onset time (ms)’
            - 'Spikes per burst'
        - dclampdatalog_take4.mat is the corresponding mat file with the following variables:
            - 'fnrow', 'cellidrow','prow', 'vrow', 'grow', 'swpnrow', 'gabab_amp', 'gabab_Trise', 'gabab_TfallFast','gabab_TfallSlow', 'gabab_w', 'currpulse', 'Rin', 'ioffset', 'imint', 'imin', 'actVhold', 'narrowpeaktime', 'narrowpeak2ndder', 'spikesperpeak', 'ltspeaktime', 'ltspeakval', 'bursttime', 'spikesperburst'
    - Data file:
        - Each sweep is resaved as a mat file containing the following variables:
            - 'd_orig': a matrix with 4 columns -
              time, **conductance**, **current**, **voltage**

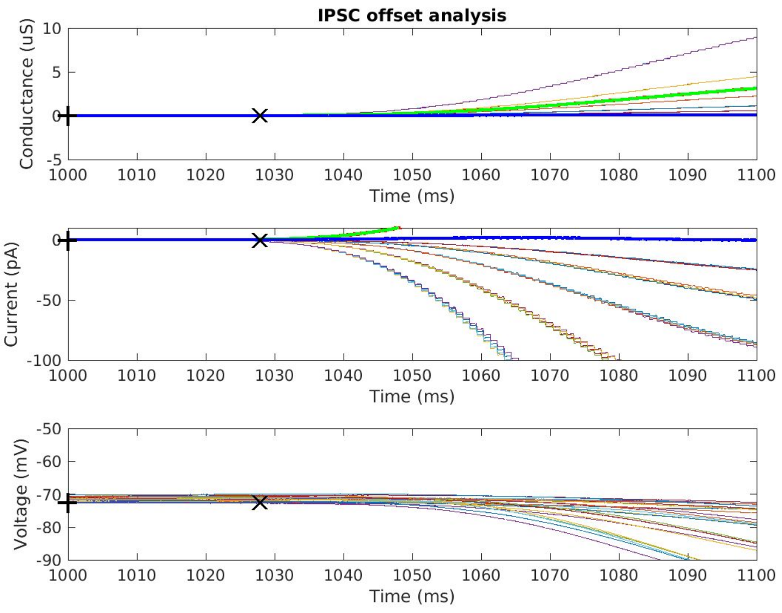
        - The conductance traces are all converted to **nS** (some are originally in **pS**)
        - Assuming Christine's belief that the current pulse applied was always 50 pA is correct, Current and conductance traces that seemed to be **arbitrarily scaled** are scaled back
        - Voltage traces are **LJP-corrected**
    - 'd_mf': same as d_orig but traces **median-filtered**
        - Median filter conductance traces to get rid of PClamp noise
        - Median filter current traces to get rid of corrupted data
        - Median filter voltage traces to get rid of spikes
    - 'd_mfrs': same as d_mf but traces **resampled**
        - Resample all traces at 1 kHz for fitting use
    - 'd_mfmaf': same as d_mf but traces **moving-average-filtered**
        - Moving-average-filter median-filtered traces for taking derivatives

- Data analysis algorithm with sample figures (E091710_0001 or E091710_0001_4)
    - Input resistance analysis:
        - This is done for each set (a pharm x Vhold pair) of 15~25 sweeps.
        - For each sweep:
            - Find the **current pulse amplitude**: mean of cpmid (**105 ms**) to cpmid + mvw (105.5 ms) minus mean of cpwin(1) (**95 ms**) to cpwin(1) + mvw (95.5 ms)
            - Find the time of **current pulse start**: last point in cpwin > cpa * ¼
            - Find the time of **current pulse end**: last point in cpwin < cpa * ¾
            - Calculate the **baseline voltage**: mean of the voltage trace between 0.5 ms before time of current pulse start and mvw (**0.5 ms**) before that
            - Calculate the **last voltage** before current pulse ends: mean of the voltage trace between time of current pulse end to mvw (**0.5 ms**) before that
            - Calculate the **voltage difference**
        - Calculate the mean current pulse amplitude (**cpa_mean**) & mean voltage difference (**dv0_mean**)
        - Put all sweeps together and fit with double exponential 'a*exp(-x/b)+c*exp(-x/d)+e' with initial conditions:
            - a = 10 * dv0_mean
            - b = 100
            - c = 10 * dv0_mean
            - d = 100
            - e = 0
        - Steady state voltage difference = -(a + c)

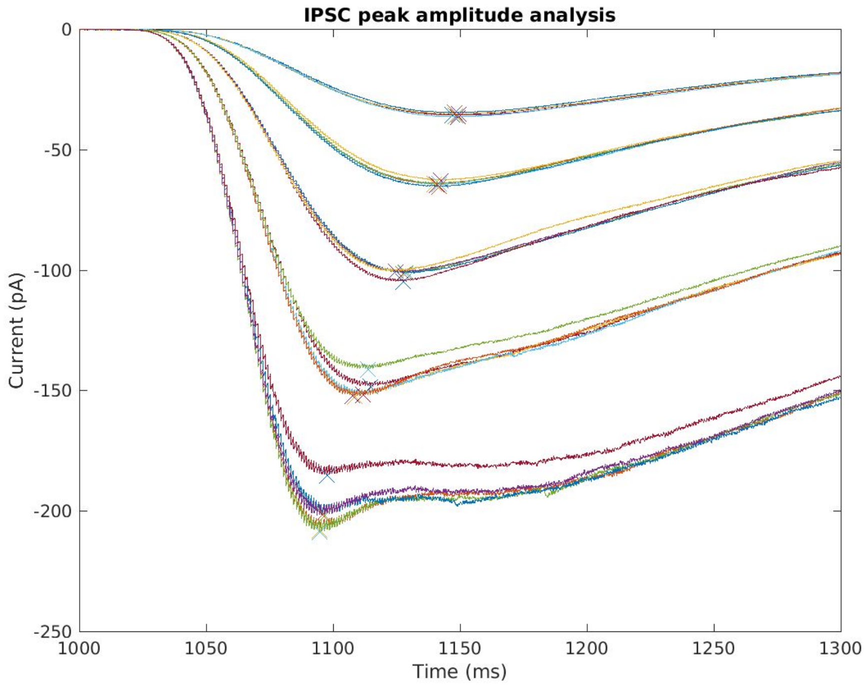
- Rinput = Steady state voltage difference / cpa_mean



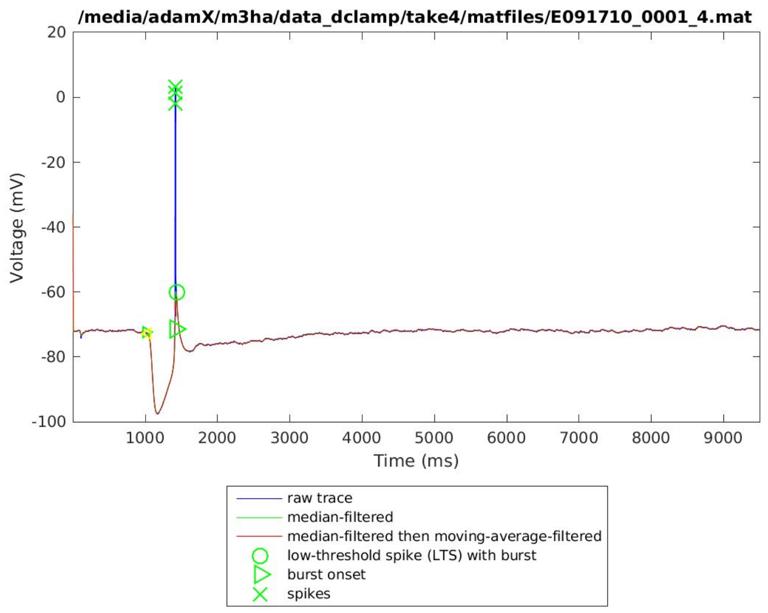
- IPSC offset analysis:
    - This is done for each set (a pharm x Vhold pair) of 15~25 sweeps.
    - Compute the **standard deviation of the current** over all sweeps
    - Smooth with a **moving average filter** spanning mafw1 (**5 ms**)
    - Find the **slope** of the smoothed standard deviation
    - Find the first point within ipsctwin (**1000~1100 ms**) with the slope greater than slth (**0.1**)



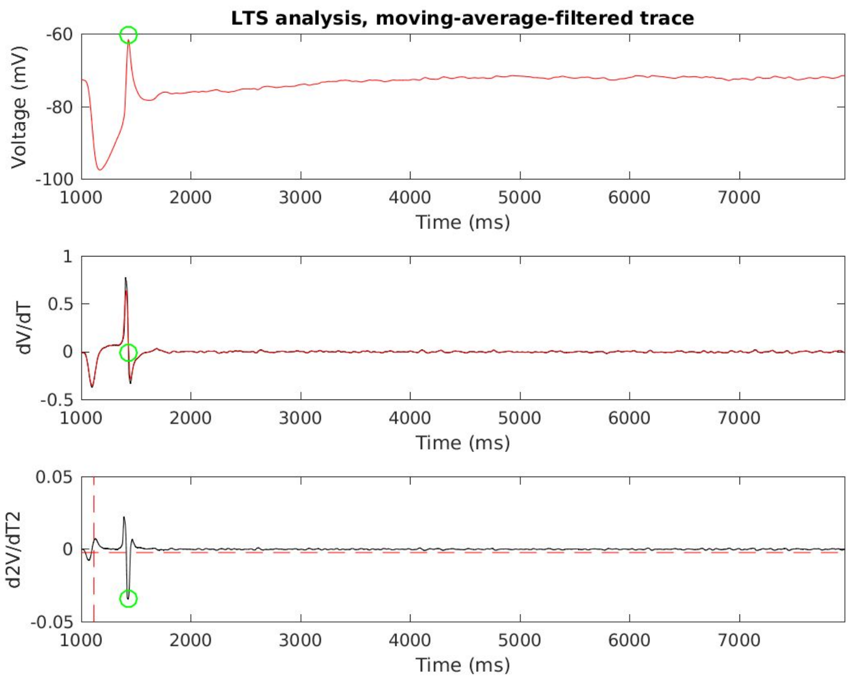
- IPSC peak amplitude analysis:
    - This is done for each sweep.
    - Find the **minimum** current value within ipscp_win (**1000~1300 ms**)



- Holding potential & LTS/burst analysis:
    - This is done for each sweep.
    - **Holding potential** is the average voltage between **IPSC offset** and blw (**20 ms**) before that

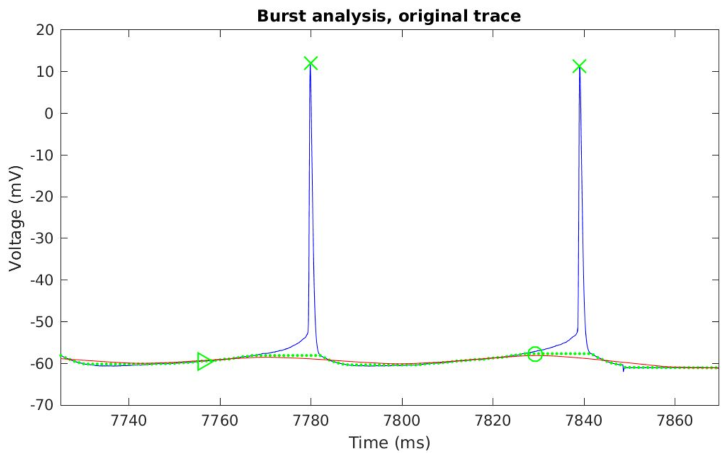


- The **second derivative** of the voltage trace is computed:
    - The voltage trace is **median-filtered** with a width of mfw3 (**30 ms**), then smoothed with a **moving-average-filter** of width mafw2 (**30 ms**)
    - The above is **differentiated**, then smoothed again with a **moving-average-filter** of width mafw2 (**30 ms**)
    - The above is **differentiated** again to yield the second derivative of the voltage trace



- The **LTS peak** is found:
    - All peaks (local maxima) of the **median-filtered than moving-average-filtered voltage trace** between the **IPSC peak** and ltswin(2) (**7960 ms**) are found
    - For each peak, do the following:
        - The **peak lower bound** and **peak upper bound** are found by locating the **first local minimum to the left** and **to the right**, respectively, with a “prominence” of the local minimum greater than maxnoiseprom (**0.1 mV**)
        - The **most negative second derivative value** is found within the bounds
        - Using the **original voltage trace**, the times of any **action potentials** (voltage >= spk_thr (**-30 mV**)) are detected within the bounds
    - Eliminate all voltage peaks with prominence < minprom (**1 mV**)
    - Eliminate all voltage peaks with the **first** action potential occurring **after** the peak of the **median-filtered than moving-average-filtered voltage trace** (this was determined to

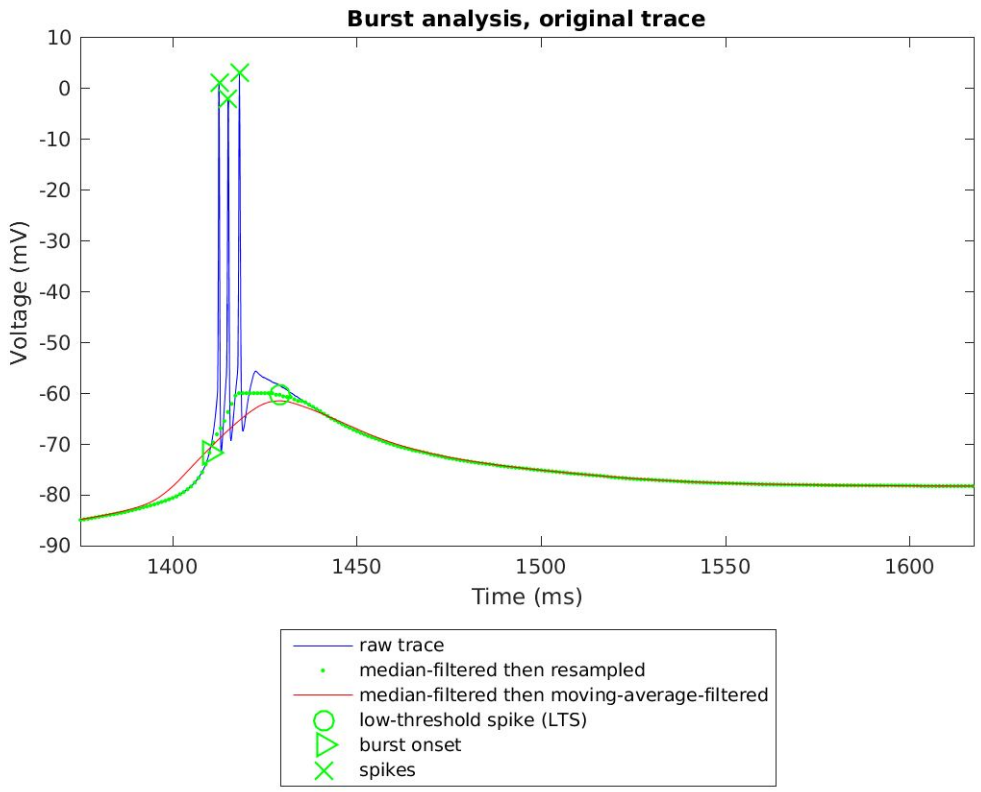
be characteristic of spontaneous spikes, see below)

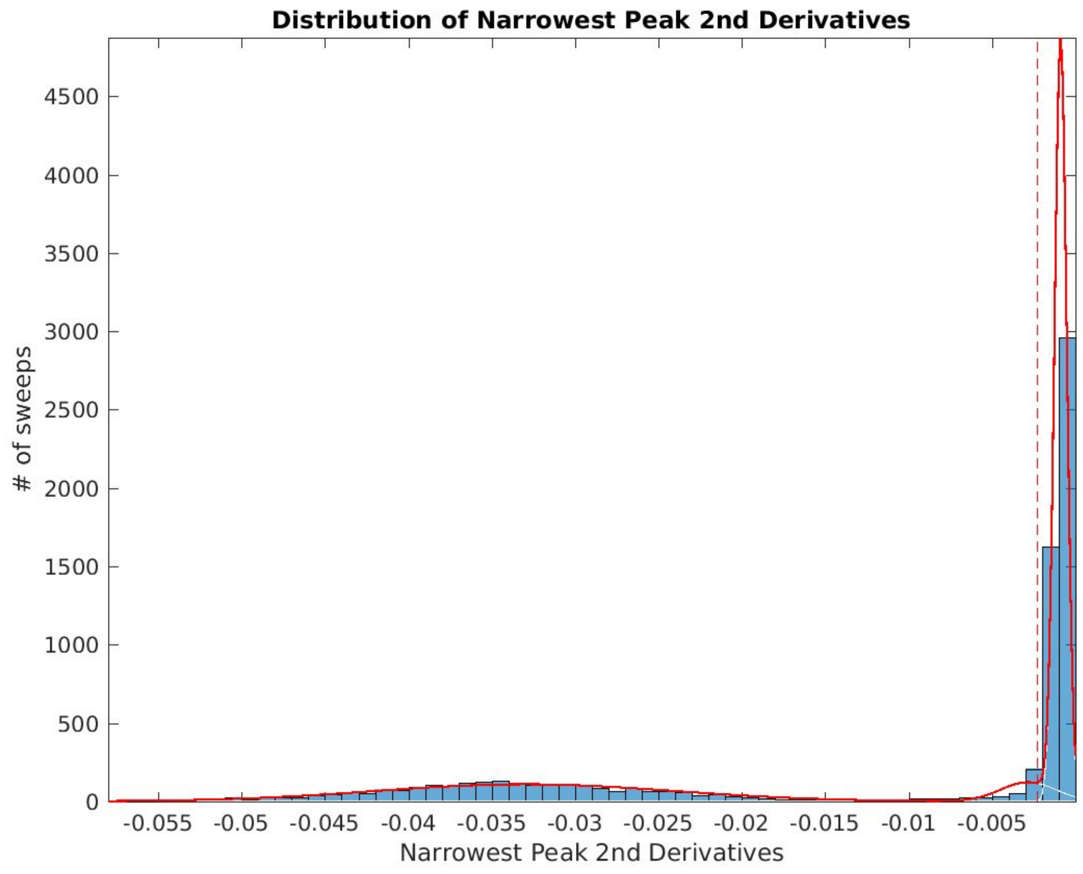


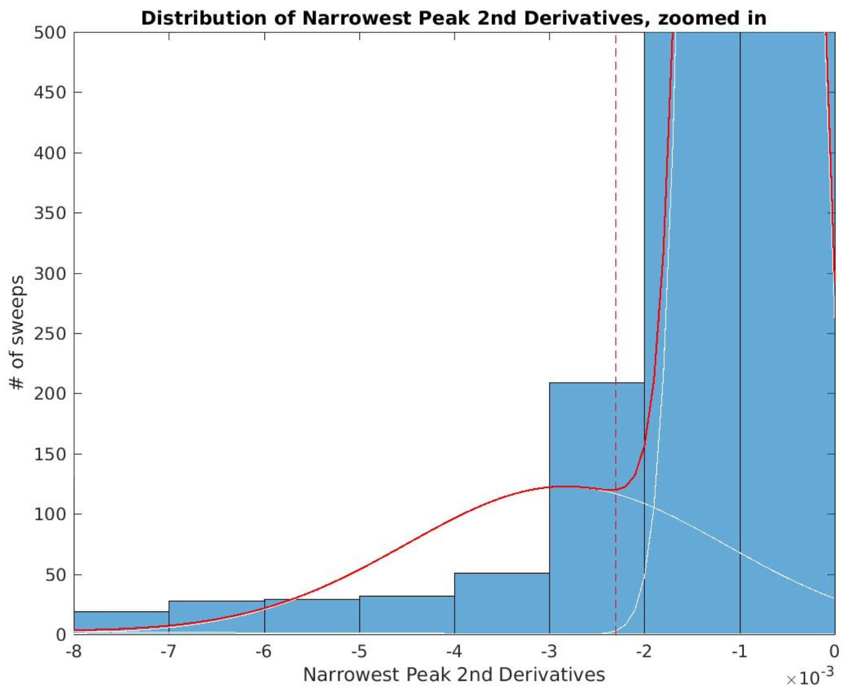
- Of the remaining peaks, the peak containing the **most negative second derivative** is the low-threshold spike.
- It's a **low threshold spike (LTS)** only if the second derivative reaches (<=) a threshold lts_thr (**-0.0023 V^2/s^2**), which is determined by fitting a 3 Gaussian curves to the distribution of most negative 2nd derivatives (see histogram on next page)
- The **LTS onset time** (LTS delay) is the time from **IPSC offset** to the **LTS peak**
- The **LTS peak amplitude** is the absolute voltage value of the **LTS peak**
- The **spikes per peak** is the number of spikes within the **LTS peak** (or within the narrowest voltage peak)
- The **spikes per burst** is the same as **spikes per peak** but must be greater than 1

■ Find the **burst onset time**:

- Find the bounds of the first spike by locating the **first local minimum to the left** and **to the right**
- The higher of the two minimums is the base of the first spike.
- The burst onset is the **last** point in time with the voltage **lower than the base of the first spike**

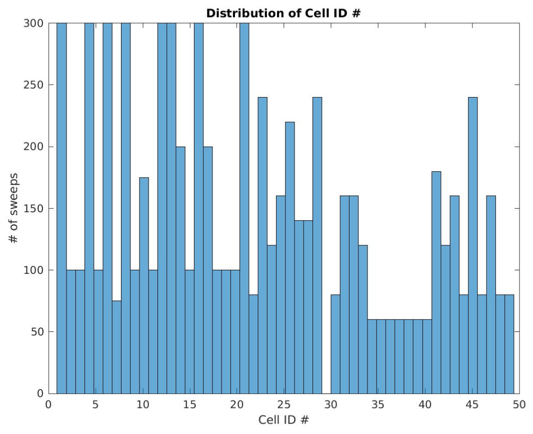

Burst analysis, original trace
Voltage (mV)
Time (ms)
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
1400
1450
1500
1550
1600
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
low-threshold spike (LTS)
burst onset
spikes

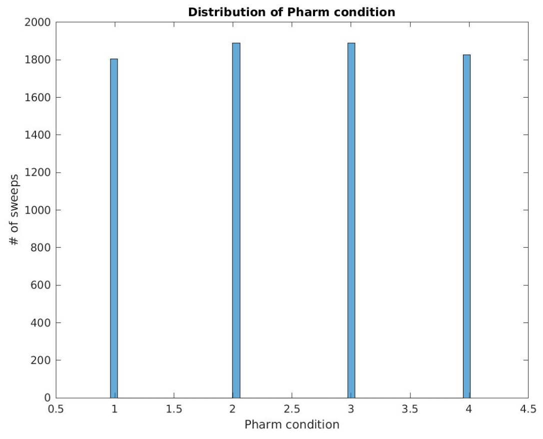

Distribution of Narrowest Peak 2nd Derivatives
# of sweeps
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-0.055
-0.05
-0.045
-0.04
-0.035
-0.03
-0.025
-0.02
-0.015
-0.01
-0.005
Narrowest Peak 2nd Derivatives

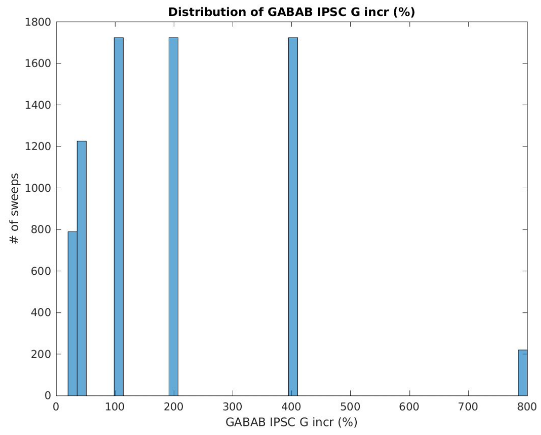

Distribution of Narrowest Peak 2nd Derivatives, zoomed in
# of sweeps
500
450
400
350
300
250
200
150
100
50
0
-8
-7
-6
-5
-4
-3
-2
-1
0
Narrowest Peak 2nd Derivatives
$\times 10^{-3}$

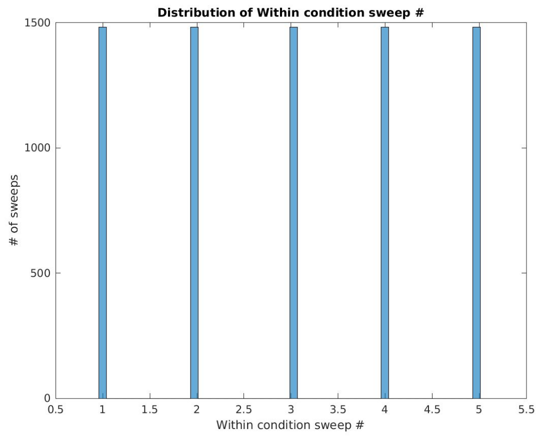
20160904

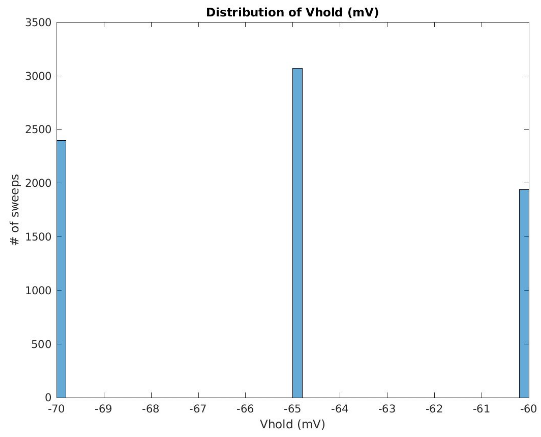
- Distribution of all data

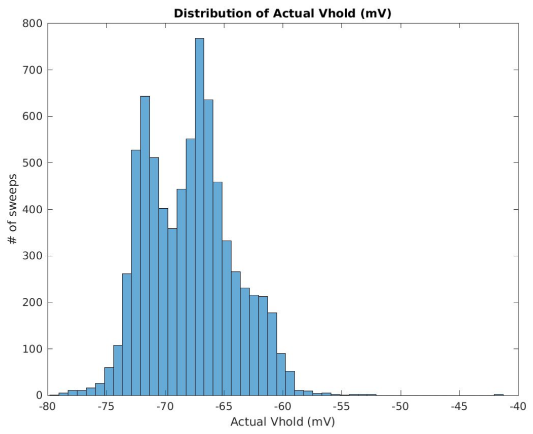


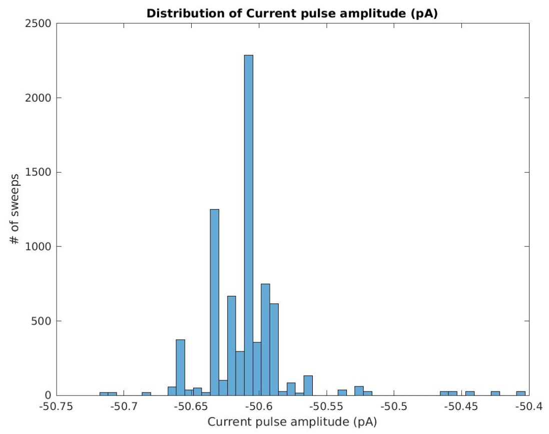


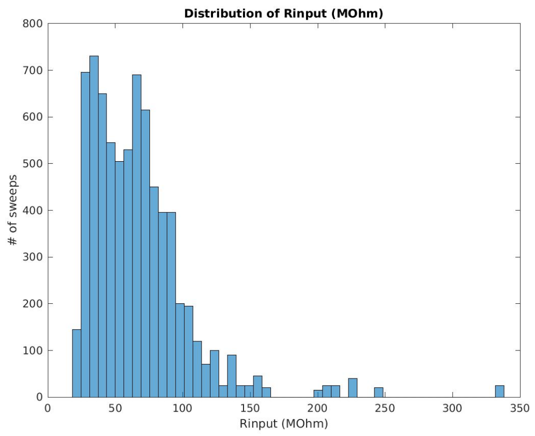


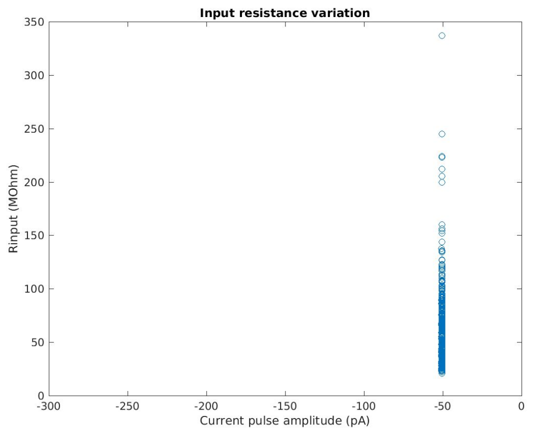


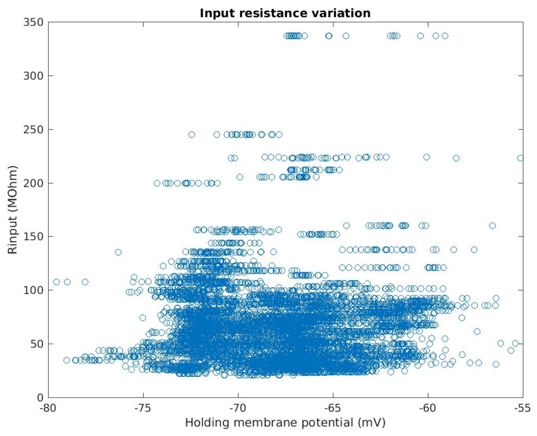

Distribution of Vhold (mV)
# of sweeps
Vhold (mV)
3500
3000
2500
2000
1500
1000
500
0
-70
-69
-68
-67
-66
-65
-64
-63
-62
-61
-60

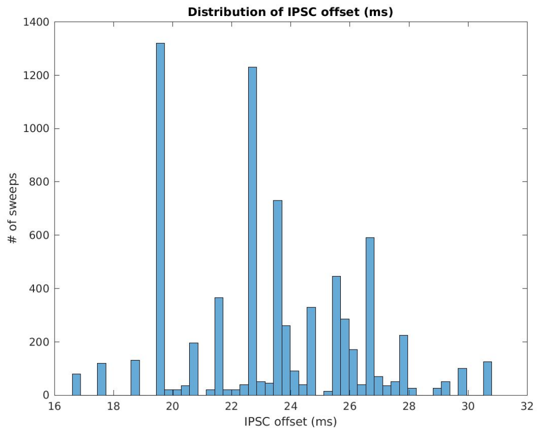

Distribution of Actual Vhold (mV)
# of sweeps
Actual Vhold (mV)
800
700
600
500
400
300
200
100
0
-80
-75
-70
-65
-60
-55
-50
-45
-40

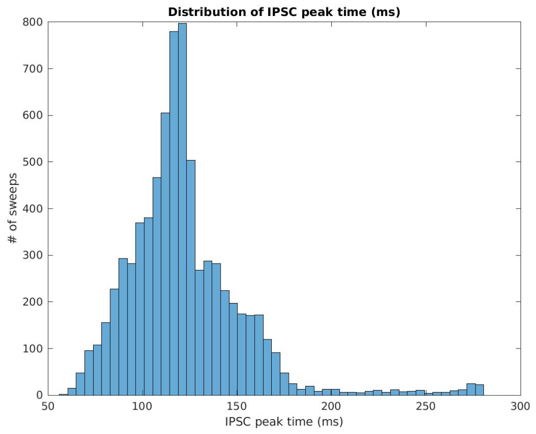

Distribution of Current pulse amplitude (pA)
# of sweeps
Current pulse amplitude (pA)
2500
2000
1500
1000
500
0
-50.75
-50.7
-50.65
-50.6
-50.55
-50.5
-50.45
-50.4

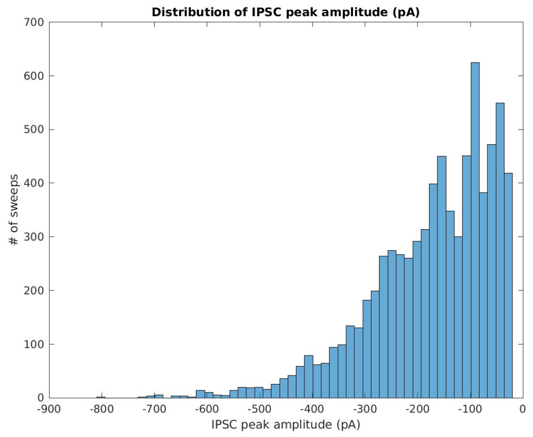

Distribution of Rinput (MOhm)
# of sweeps
Rinput (MOhm)
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350

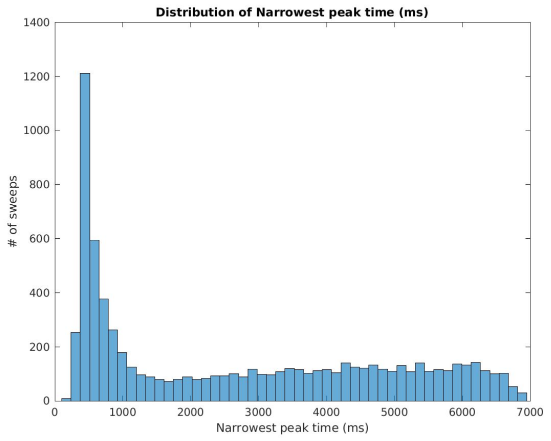

Input resistance variation
Rinput (MOhm)
Current pulse amplitude (pA)
350
300
250
200
150
100
50
0
-300
-250
-200
-150
-100
-50
0

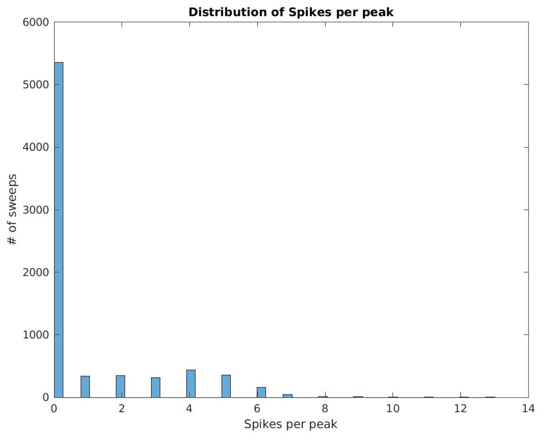

Input resistance variation
Rinput (MOhm)
Holding membrane potential (mV)
350
300
250
200
150
100
50
0
-80
-75
-70
-65
-60
-55

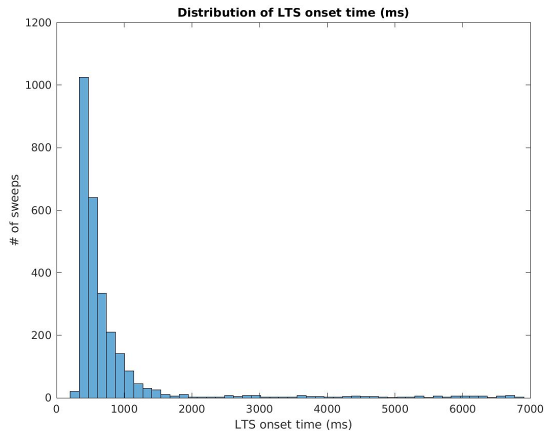

Distribution of IPSC offset (ms)
# of sweeps
IPSC offset (ms)

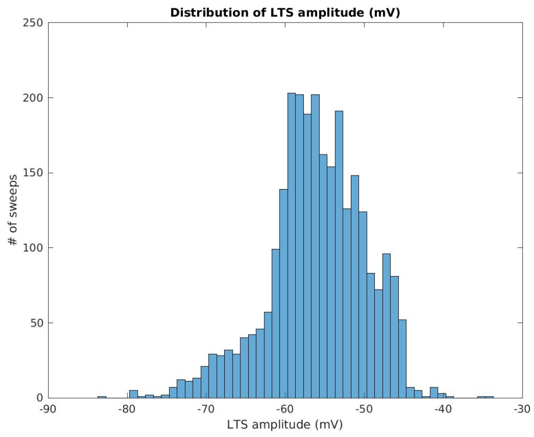

Distribution of IPSC peak time (ms)
# of sweeps
IPSC peak time (ms)

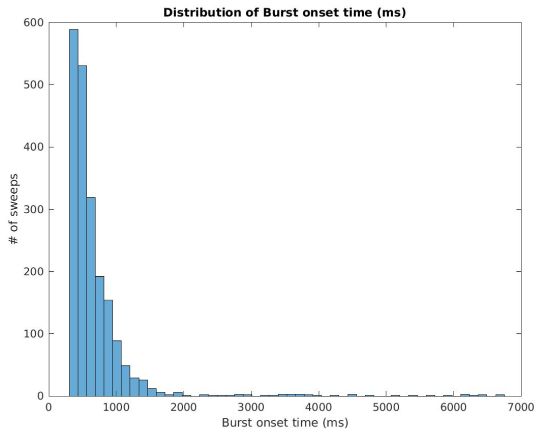

Distribution of IPSC peak amplitude (pA)
# of sweeps
IPSC peak amplitude (pA)

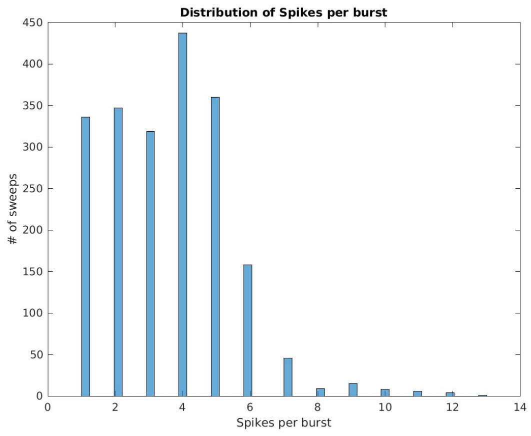

Distribution of Narrowest peak time (ms)
# of sweeps
Narrowest peak time (ms)

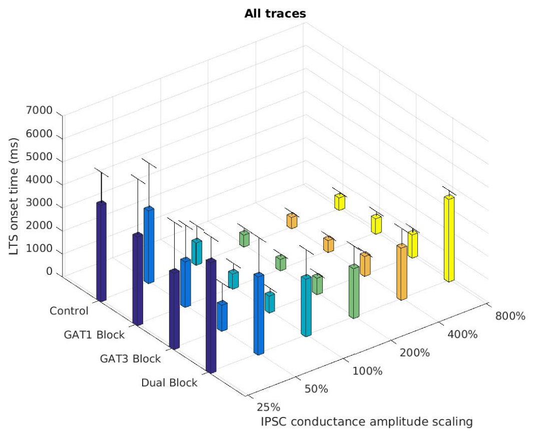

Distribution of Spikes per peak
# of sweeps
Spikes per peak

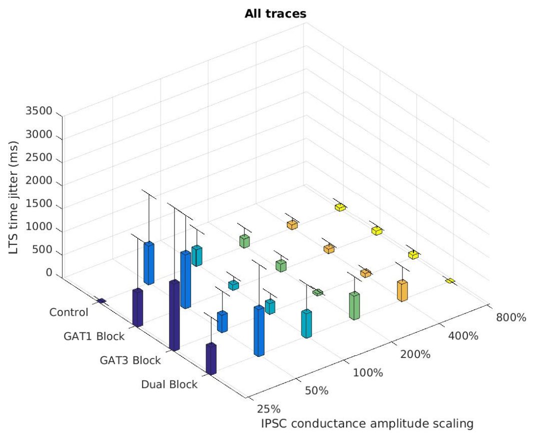

Distribution of LTS onset time (ms)
# of sweeps
LTS onset time (ms)

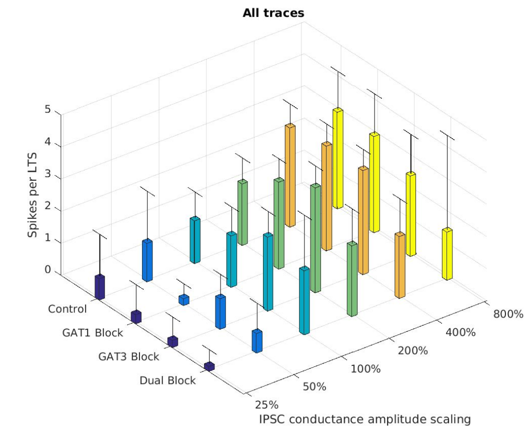

Distribution of LTS amplitude (mV)
# of sweeps
LTS amplitude (mV)

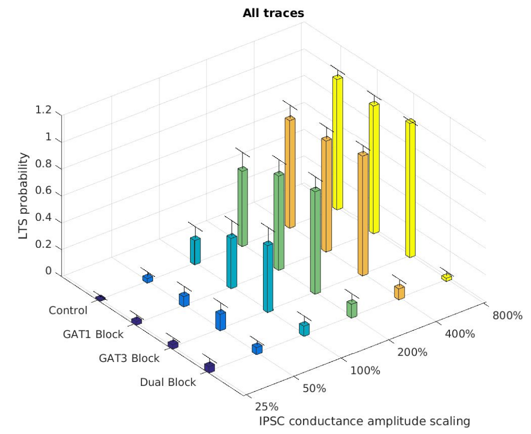

Distribution of Burst onset time (ms)
# of sweeps
Burst onset time (ms)

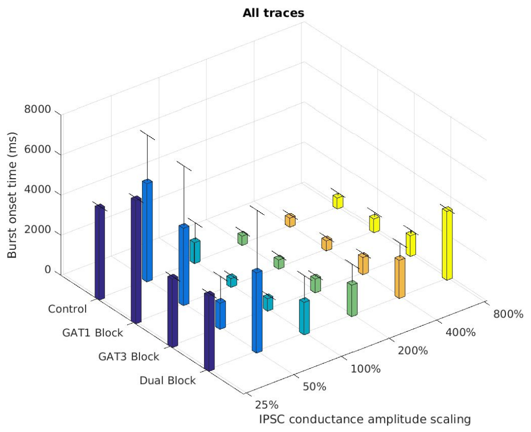

Distribution of Spikes per burst
# of sweeps
Spikes per burst

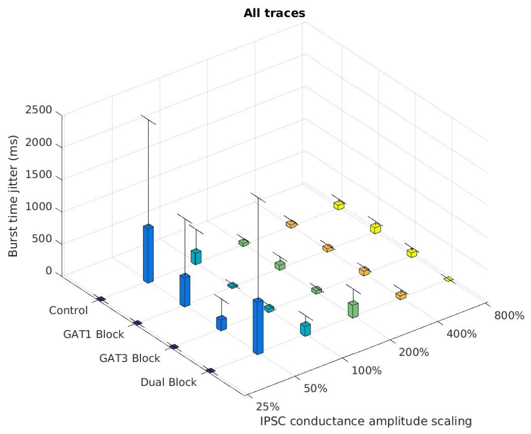

All traces
LTS onset time (ms)
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

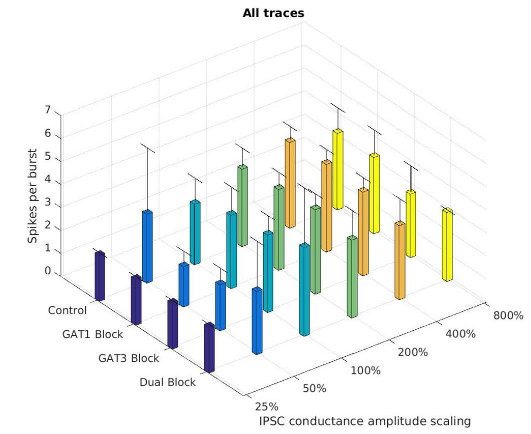

All traces
LTS time jitter (ms)
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

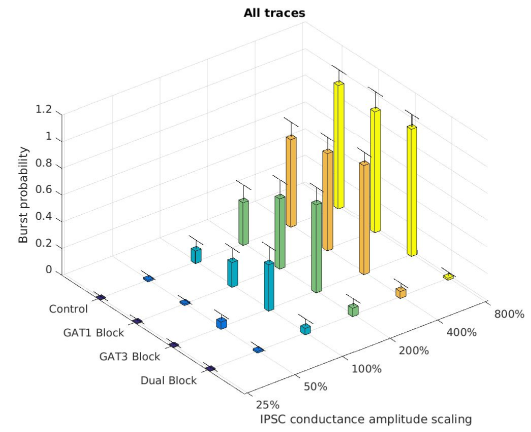

All traces
Spikes per LTS
0
1
2
3
4
5
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

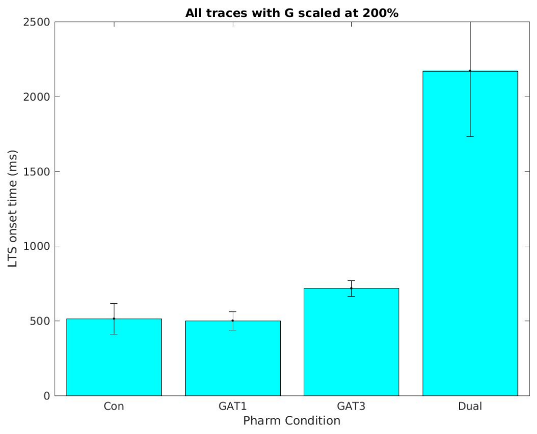

All traces
LTS probability
0
0.2
0.4
0.6
0.8
1
1.2
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

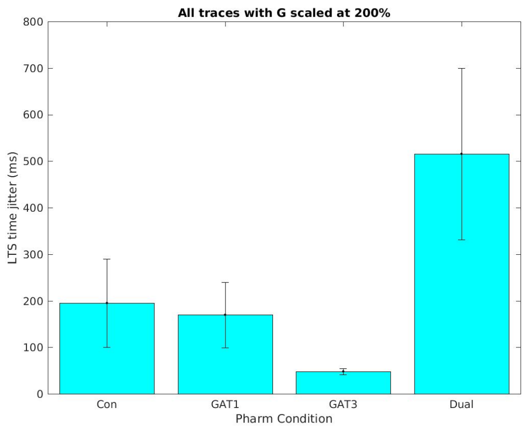

All traces
Burst onset time (ms)
0
2000
4000
6000
8000
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

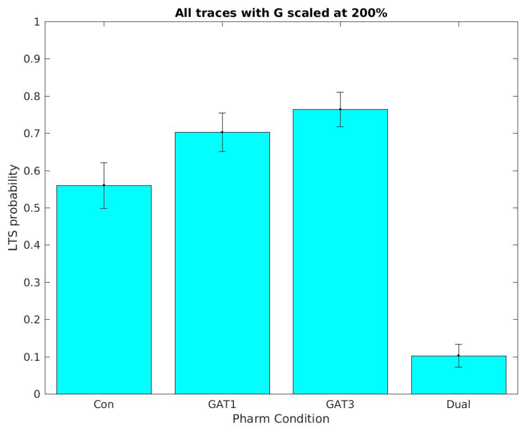

All traces
Burst time jitter (ms)
0
500
1000
1500
2000
2500
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

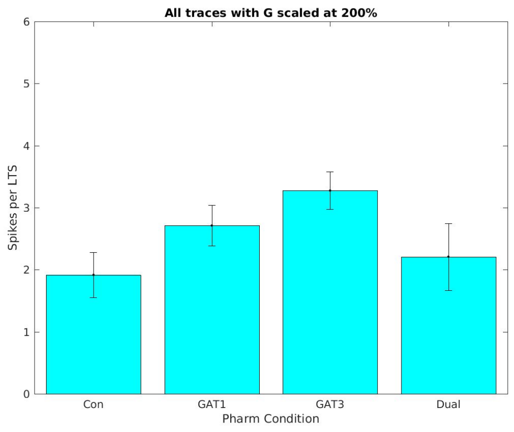

All traces
Spikes per burst
0
1
2
3
4
5
6
7
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

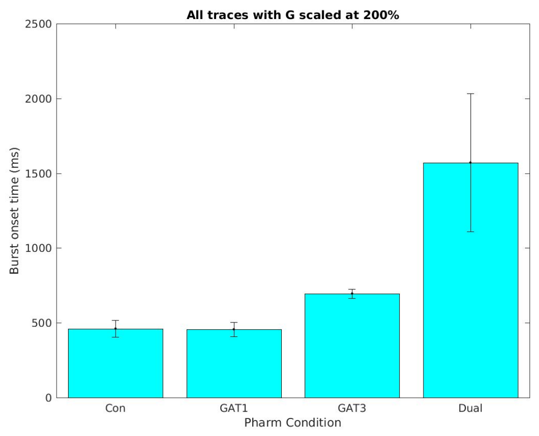

All traces
Burst probability
0
0.2
0.4
0.6
0.8
1
1.2
Control
GAT1 Block
GAT3 Block
Dual Block
25%
50%
100%
200%
400%
800%
IPSC conductance amplitude scaling

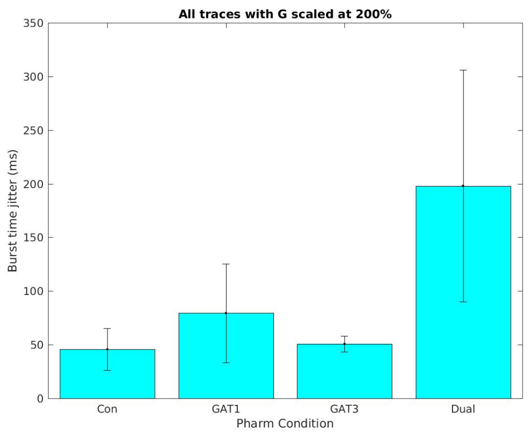

All traces with G scaled at 200%
LTS onset time (ms)
2500
2000
1500
1000
500
0
Con
GAT1
GAT3
Dual
Pharm Condition

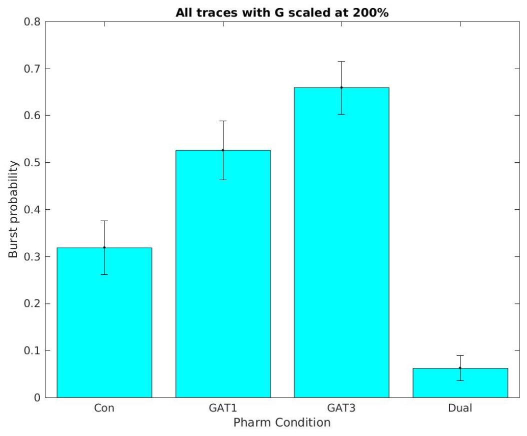

All traces with G scaled at 200%
LTS time jitter (ms)
800
700
600
500
400
300
200
100
0
Con
GAT1
GAT3
Dual
Pharm Condition

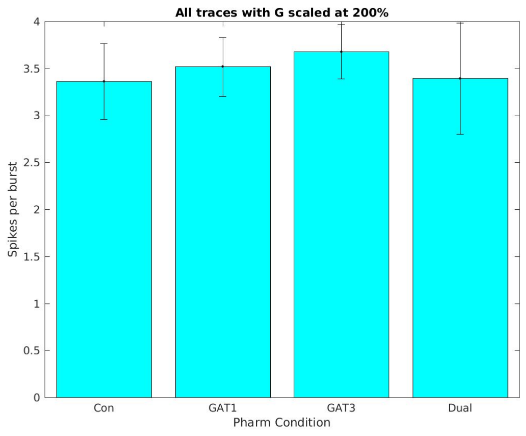

All traces with G scaled at 200%
LTS probability
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Con
GAT1
GAT3
Dual
Pharm Condition


All traces with G scaled at 200%
Spikes per LTS
6
5
4
3
2
1
0
Con
GAT1
GAT3
Dual
Pharm Condition

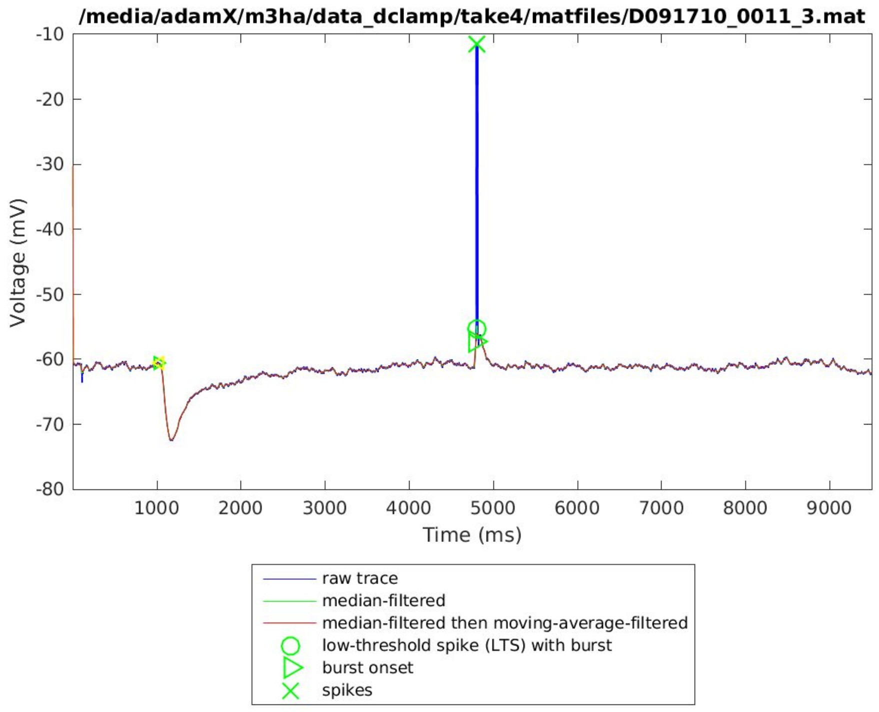

All traces with G scaled at 200%
Burst onset time (ms)
2500
2000
1500
1000
500
0
Con
GAT1
GAT3
Dual
Pharm Condition

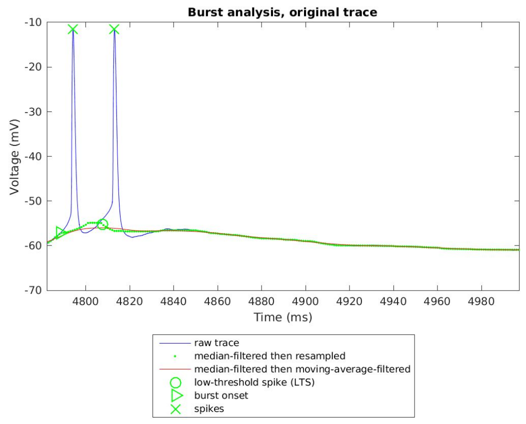

All traces with G scaled at 200%
Burst time jitter (ms)
350
300
250
200
150
100
50
0
Con
GAT1
GAT3
Dual
Pharm Condition

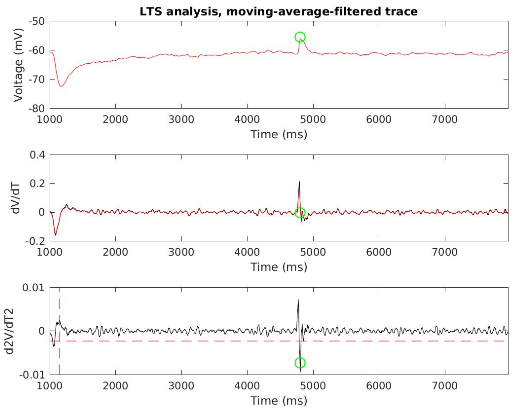

All traces with G scaled at 200%
Burst probability
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Con
GAT1
GAT3
Dual
Pharm Condition

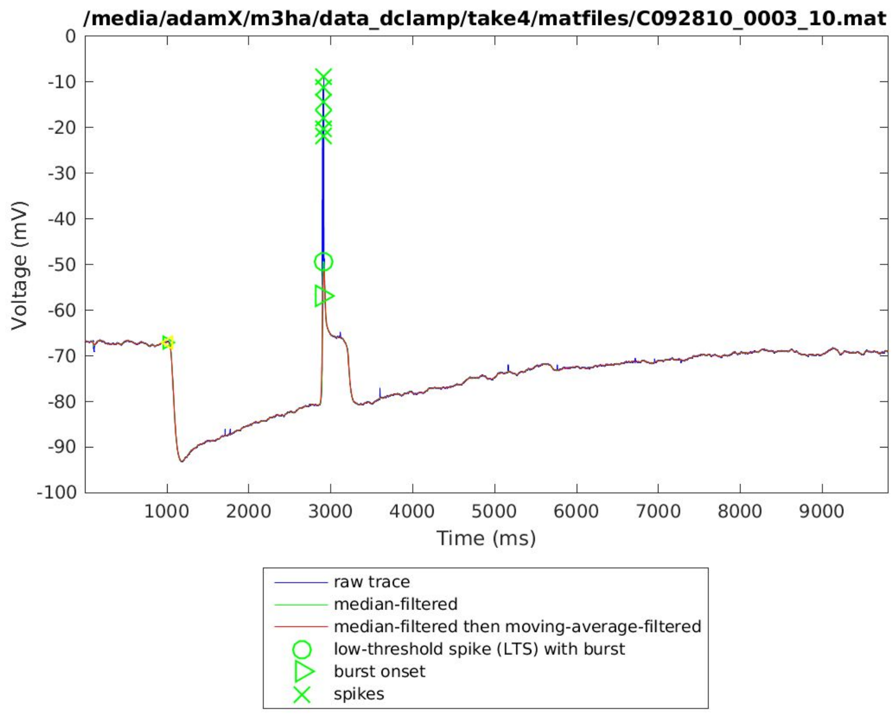

All traces with G scaled at 200%
Spikes per burst
4
3.5
3
2.5
2
1.5
1
0.5
0
Con
GAT1
GAT3
Dual
Pharm Condition

- Possible false positives:
    - “Long latency LTS/bursts”

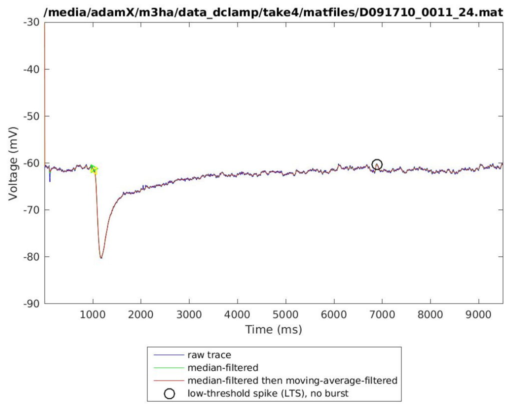


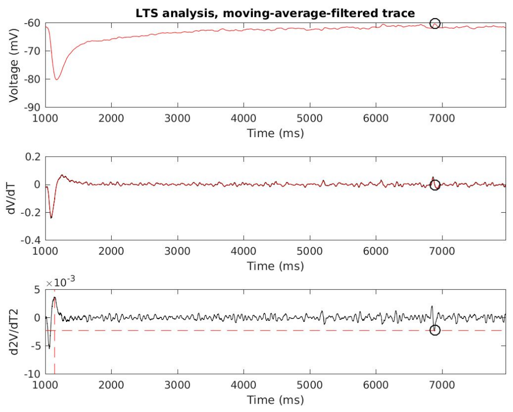


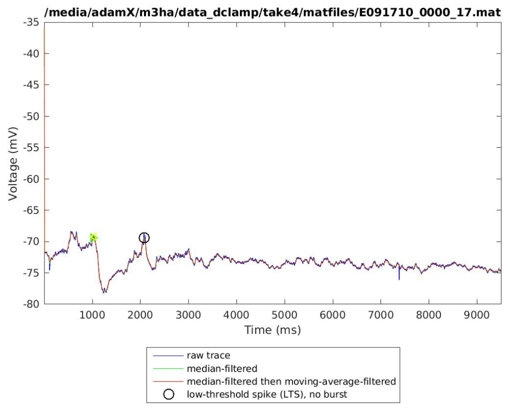


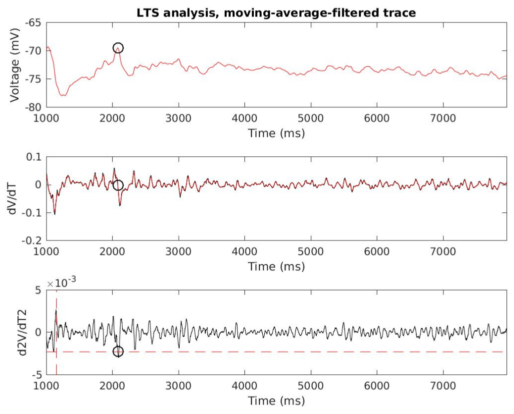

/media/adamX/m3ha/data_dclamp/take4/matfiles/C092810_0003_10.mat
Voltage (mV)
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
1000
2000
3000
4000
5000
6000
7000
8000
9000
Time (ms)
raw trace
median-filtered
median-filtered then moving-average-filtered
low-threshold spike (LTS) with burst
burst onset
spikes

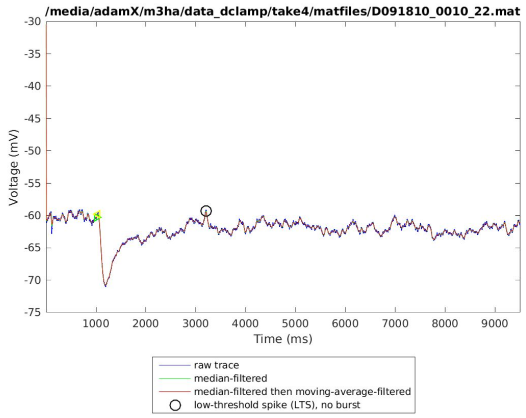
- Looks like noise?

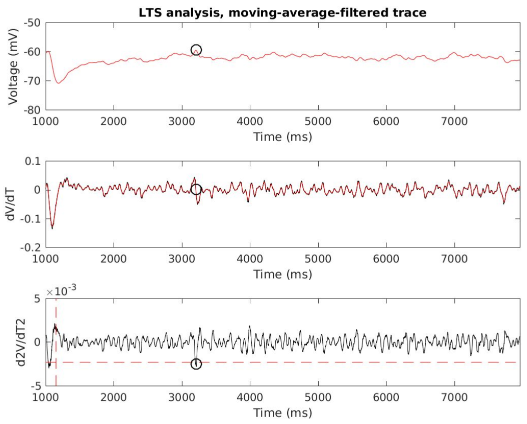




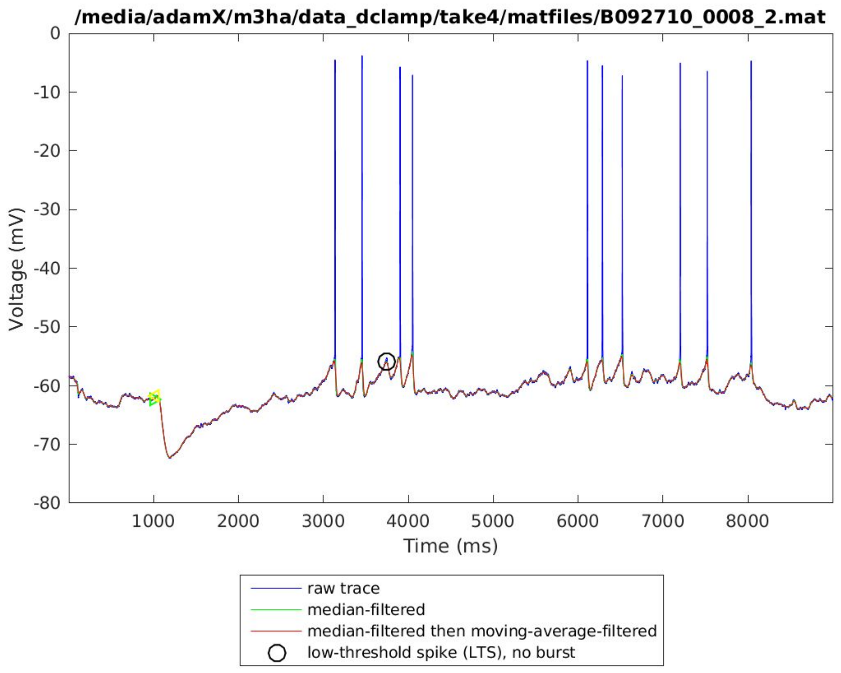


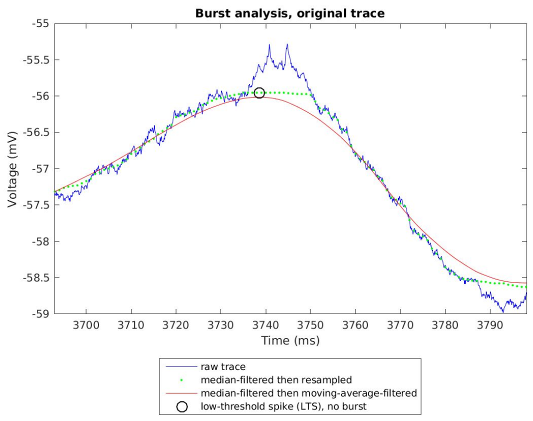


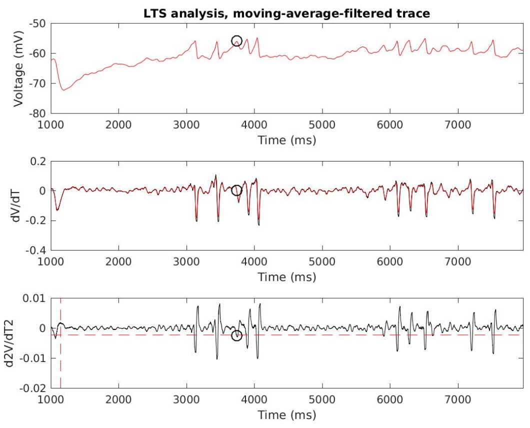




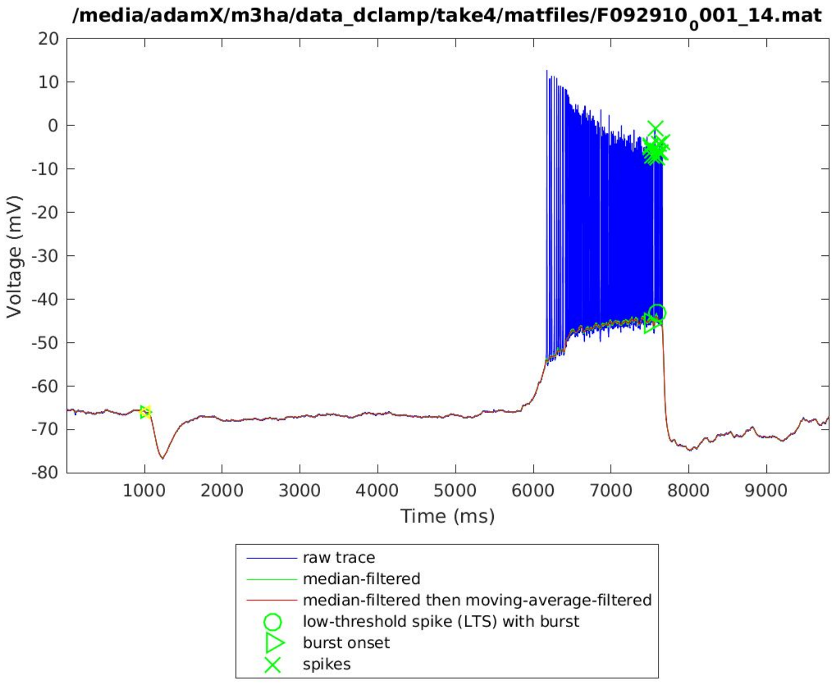
- Spontaneous spike that did not fire

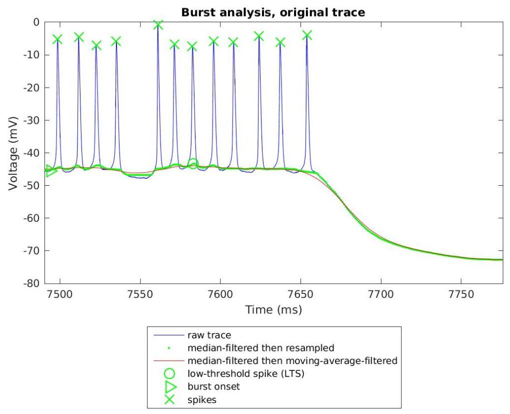


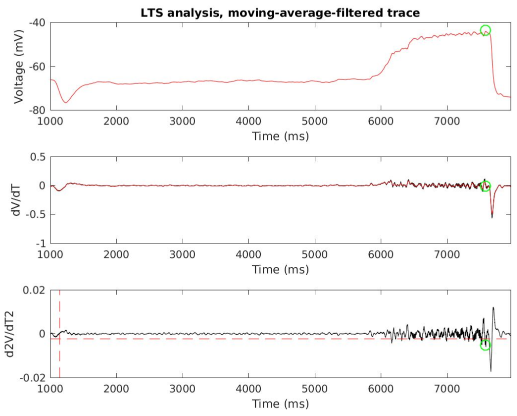


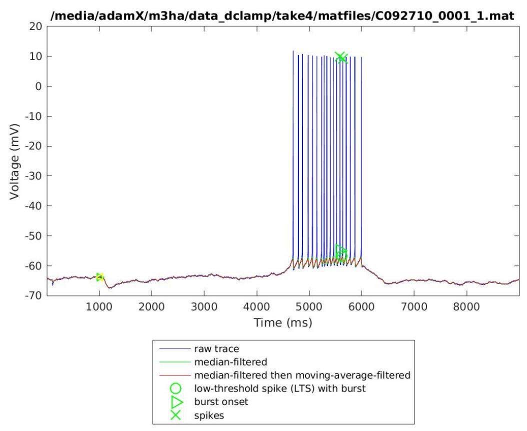


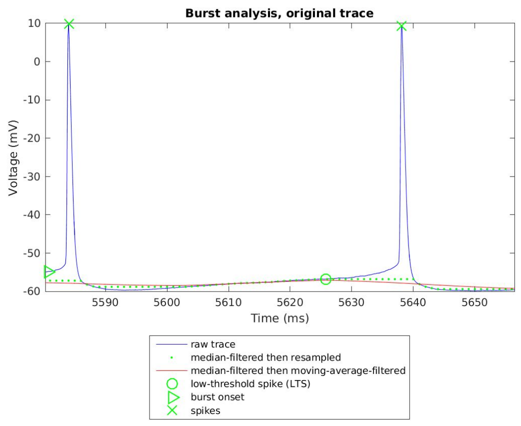
- Spontaneous bursts



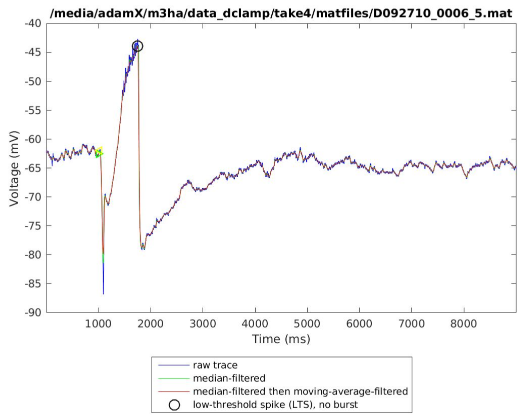


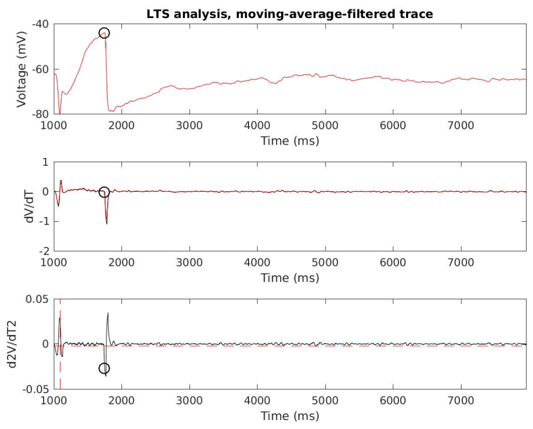


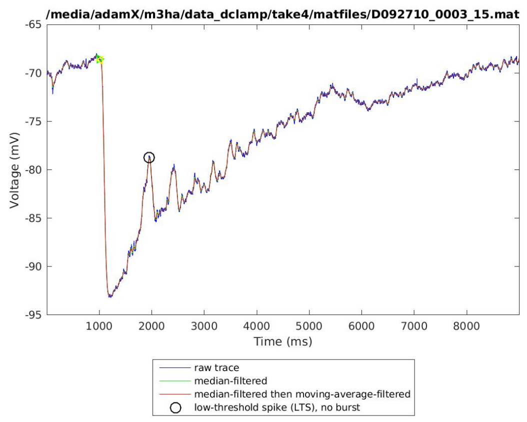

/media/adamX/m3ha/data_dclamp/take4/matfiles/C092710_0001_1.mat
Voltage (mV)
Time (ms)
raw trace
median-filtered
median-filtered then moving-average-filtered
low-threshold spike (LTS) with burst
burst onset
spikes


Burst analysis, original trace
Voltage (mV)
Time (ms)
raw trace
median-filtered then resampled
median-filtered then moving-average-filtered
low-threshold spike (LTS)
burst onset
spikes

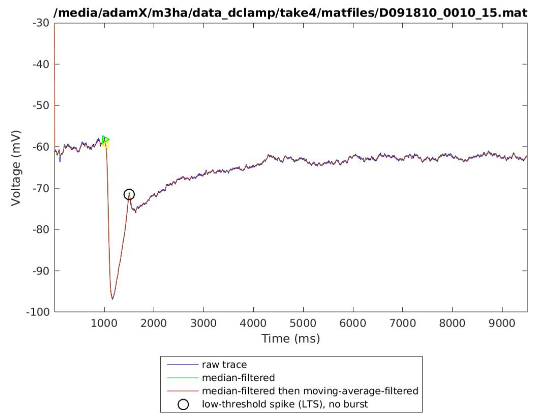
- Weird shape

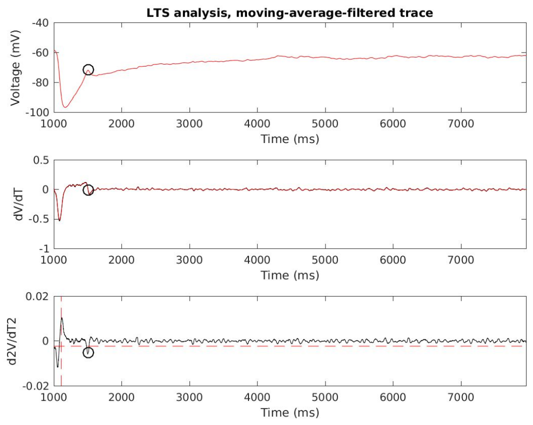

/media/adamX/m3ha/data_dclamp/take4/matfiles/D092710_0006_5.mat
Voltage (mV)
Time (ms)
raw trace
median-filtered
median-filtered then moving-average-filtered
low-threshold spike (LTS), no burst

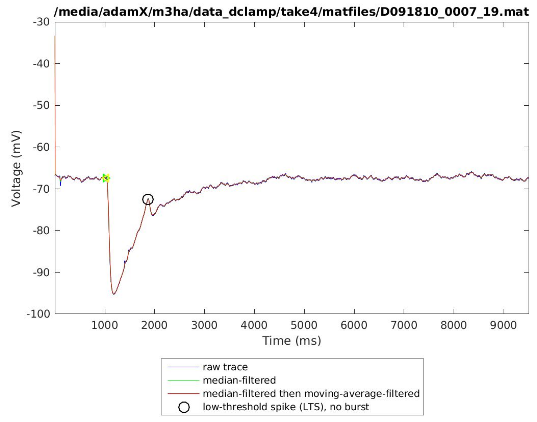

LTS analysis, moving-average-filtered trace
Voltage (mV)
Time (ms)
dV/dT
d2V/dT2

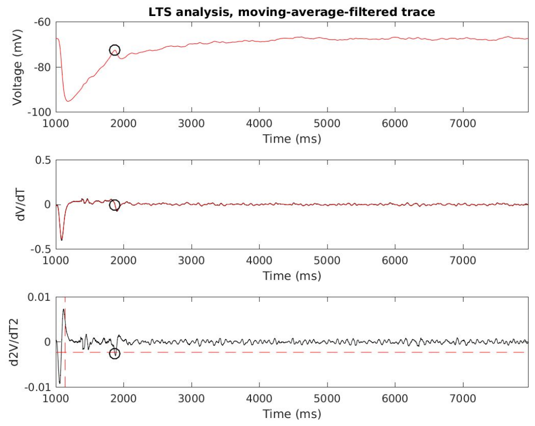

/media/adamX/m3ha/data_dclamp/take4/matfiles/D092710_0003_15.mat
Voltage (mV)
Time (ms)
raw trace
median-filtered
median-filtered then moving-average-filtered
low-threshold spike (LTS), no burst

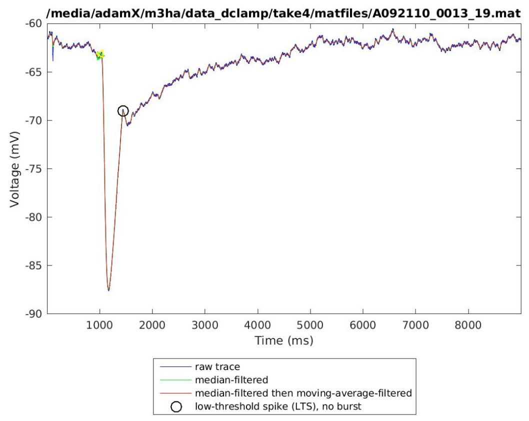
- True positives:
    - Very small LTS (now a true positive with such loose criterium):

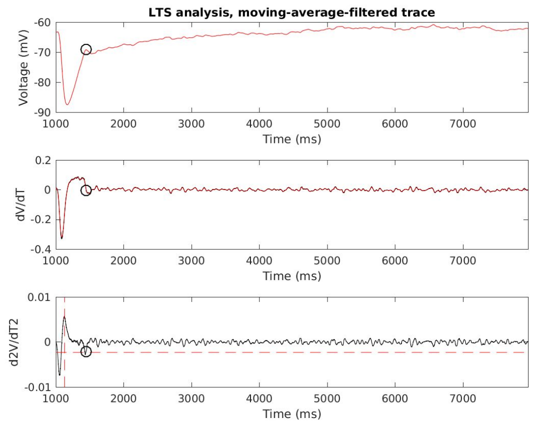




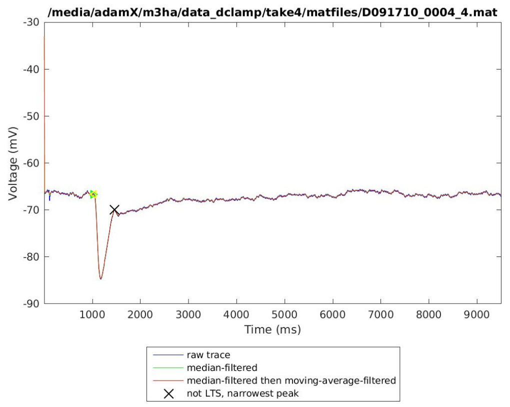


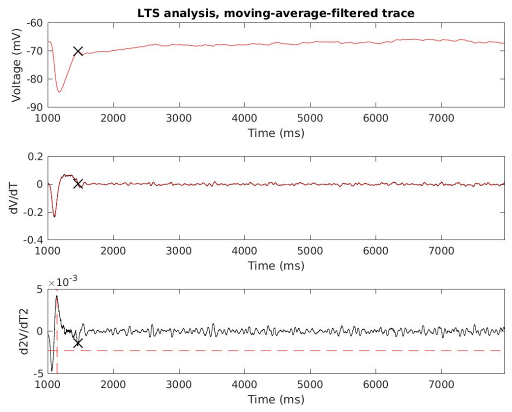


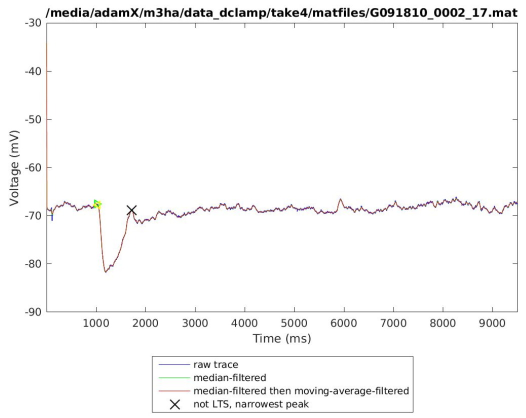


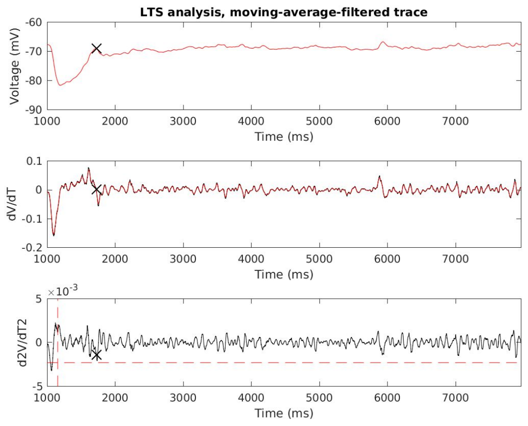


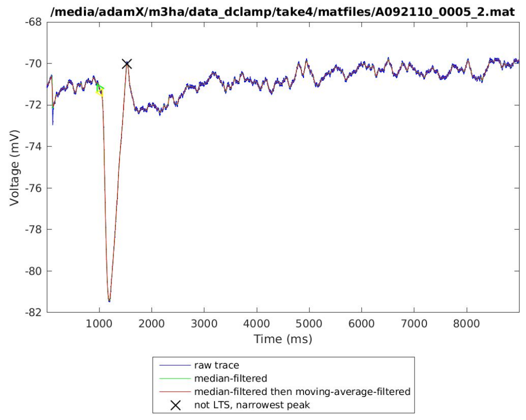
- Possible false negatives:
  - Looks like LTS, but operationally weeded out:

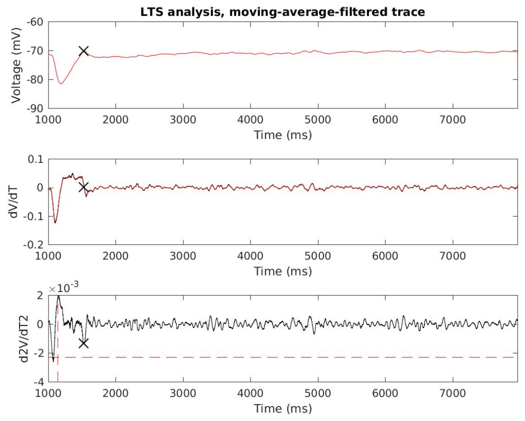




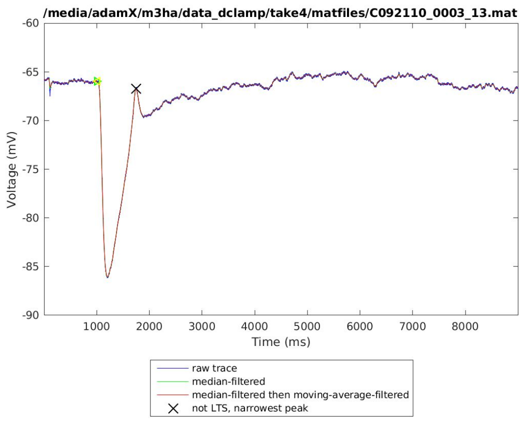


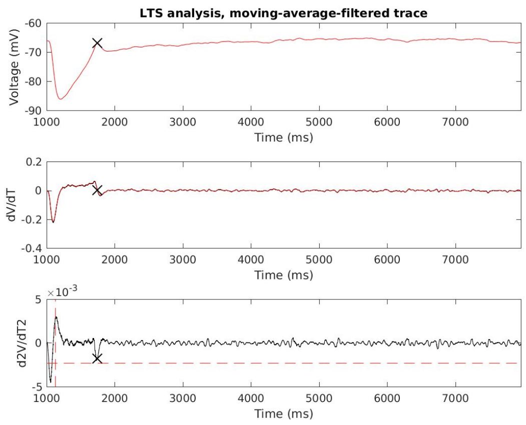




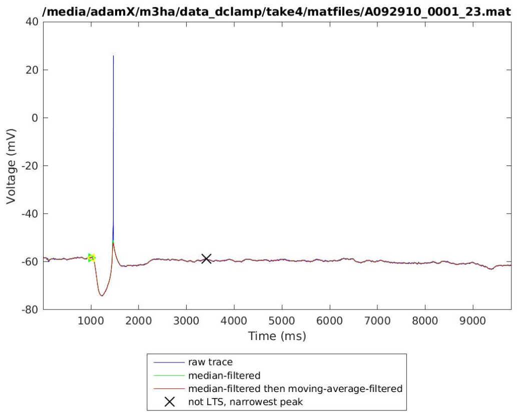


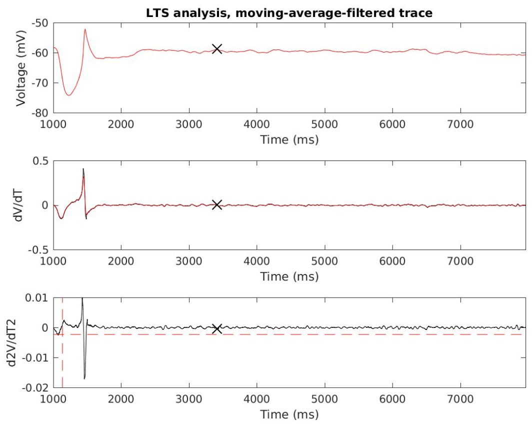
20160904





○ Missed LTS:

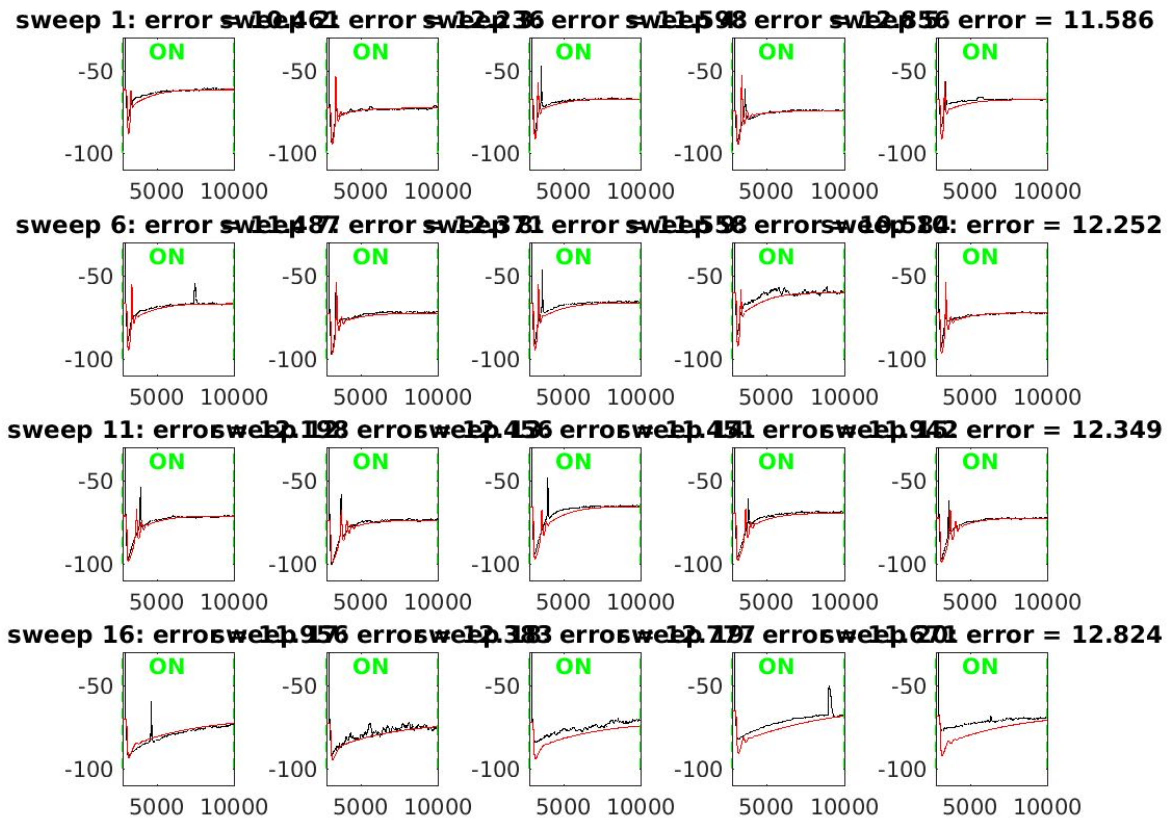




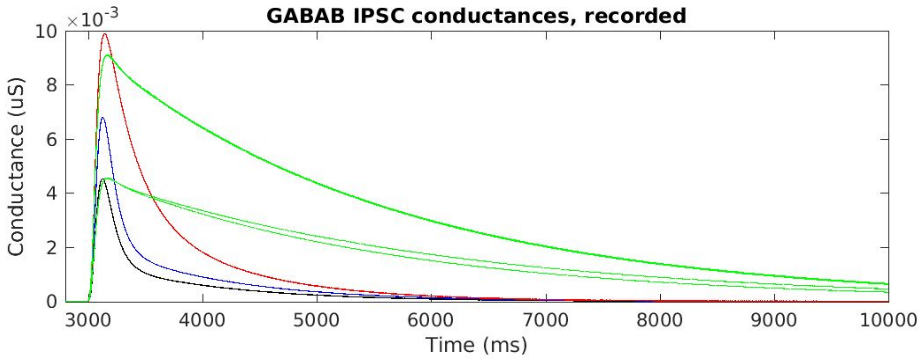
8/30/2016~

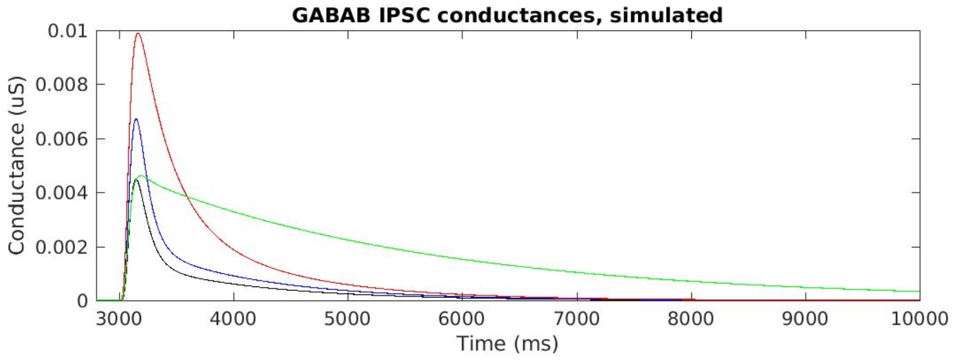
Things to discuss with Dad, Mark & John

1. Single-neuron data/experiment:
    a. IPSC offset -- is it real?
    TO DO: Compare conductance peak with theoretical peak
    b. Anomalous dual block data. How much current was actually injected?
    TO DO: Compare recorded conductance & current traces with the corresponding theoretical traces

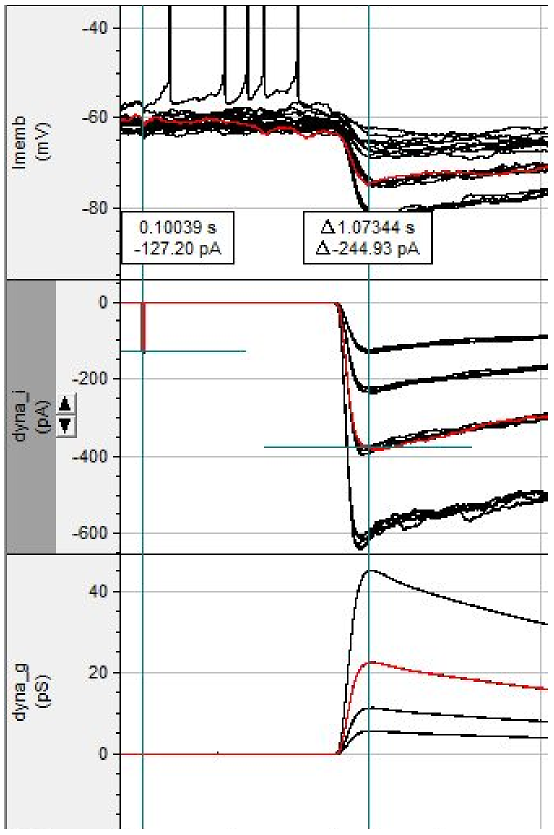


F092710_0007_4.mat↑ ↑C092910_0003_3.mat

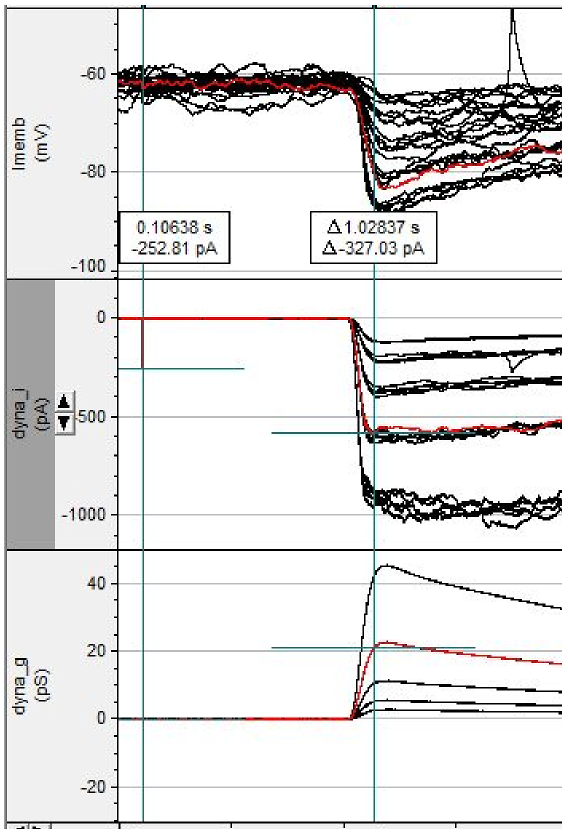




C092910_0003_3.mat
10929-13_58_49.PRN

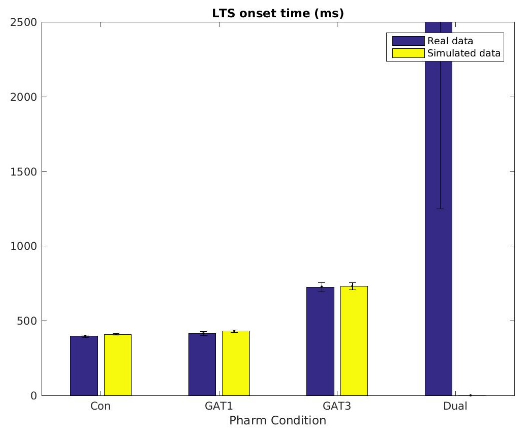


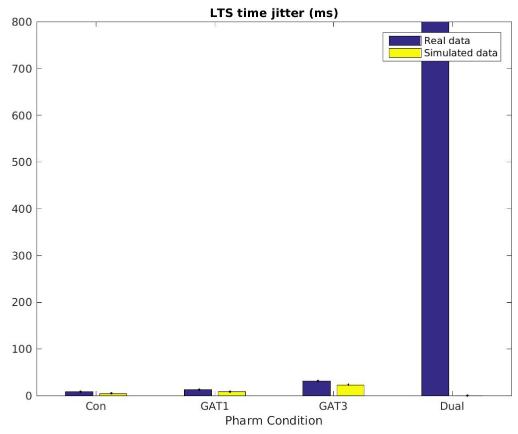
F092710_0007_4.mat
10927-18_27_26.PRN

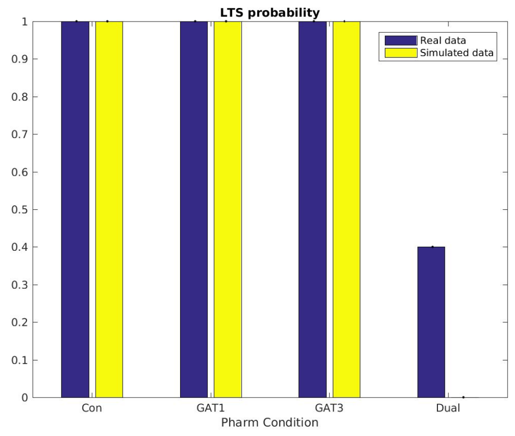


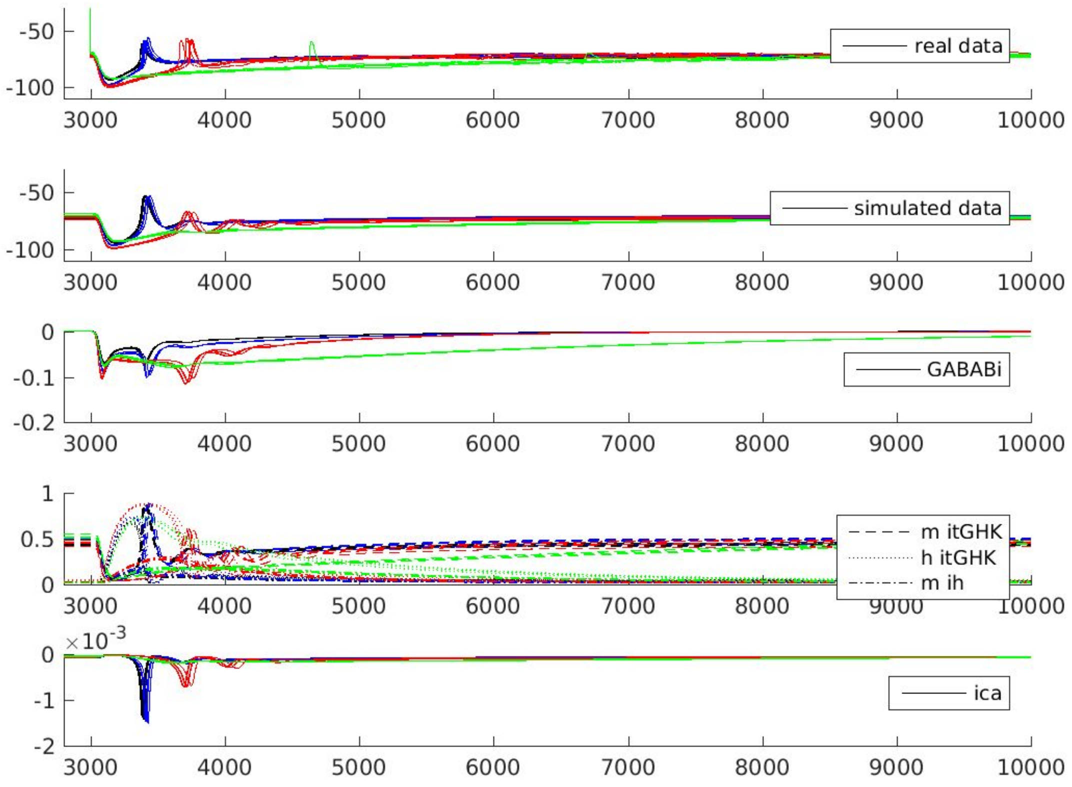
   c. Was the IPSC injected on top of the holding current? Yes.
   d. Was the TC response to each IPSC injection compared to the response to the corresponding pharm conditions? No, **we may want to do this**.
   e. Are long-latency (e.g., > 3000 ms) LTSs actually LTSs? Depends on shape and whether there is still current injected.
      TO DO: Try a positive 2nd derivative threshold for spontaneous LTSs (see C092810_0003_10). Sort all remaining questionable traces into different directories and present all of them the next time we meet. Remove all problematic traces, along with the all traces from the same cell-pharm-gincr set from data analysis
   f. Input resistance data: what can we actually use from this to apply to our model? I.e., what parameters of the model can be extracted from this data?
      TO DO: Instead of calculating the input resistance and using it, fit the model to the current injection responses to extract parameters relevant to the input resistance
   g. Burst analysis:
      TO DO: Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly
   h. Burst statistics:
      TO DO: Perform ANOVA to determine whether there are significant changes
2. Single-neuron model implementation/fitting:
   a. Currently, all channels of Amarillo et al are included except HH. Should HH be included (might slow down simulation and is not important for determining LTS onset time, but might be important for burst probability)? What about INaP & IA?
      TO DO: Keep all channels of Amarillo et al except HH. Analyze the correlation between the burst probability, spikes per peak, spikes per burst, etc. and the slope or other features of an LTS. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
   b. How were the maximal ranges determined for each parameter? Not sure, but Christine's code had the option of extending the maximal range.
   c. Can long-latency (e.g., > 3000 ms) LTSs be fitted?
      TO DO:
      i. Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time
      ii. Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data
   d. Inter-trial variability - Determine **trial-variable parameters** using data from E091710 (the particular cell whose data Christine fitted to):
      i. Trace selection:
         1. 200% g incr only

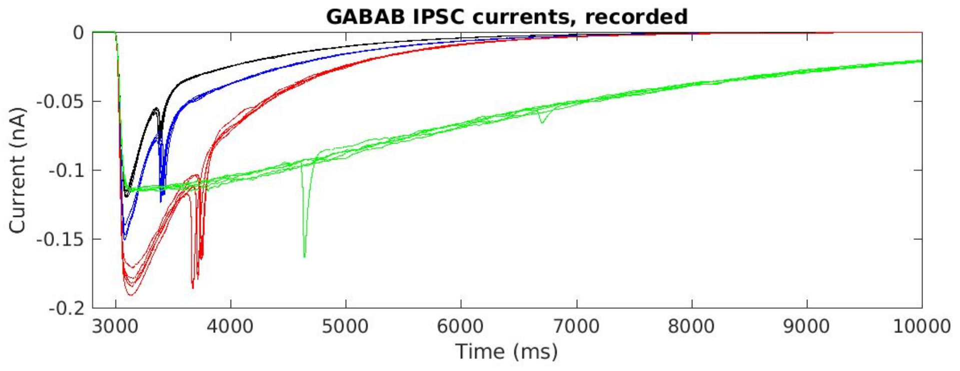
20160904

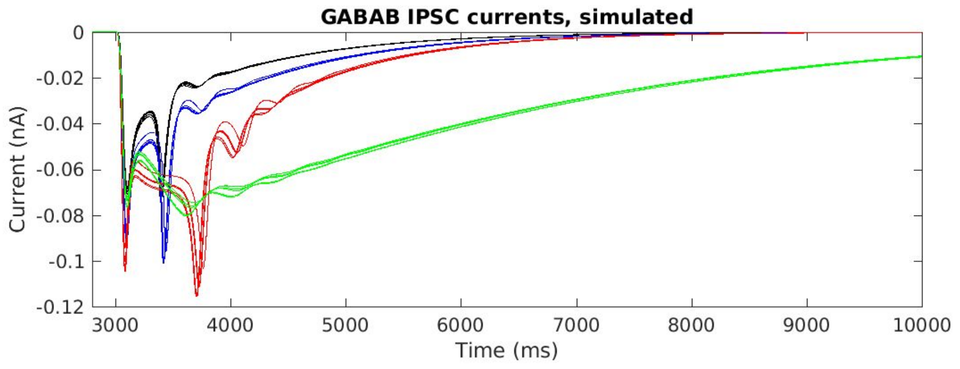

LTS onset time (ms)
Real data
Simulated data
0
500
1000
1500
2000
2500
Con
GAT1
GAT3
Dual
Pharm Condition

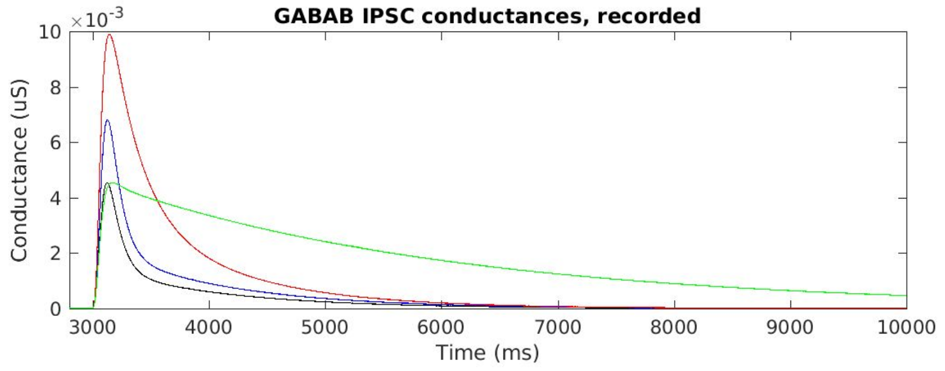

LTS time jitter (ms)
Real data
Simulated data
0
100
200
300
400
500
600
700
800
Con
GAT1
GAT3
Dual
Pharm Condition

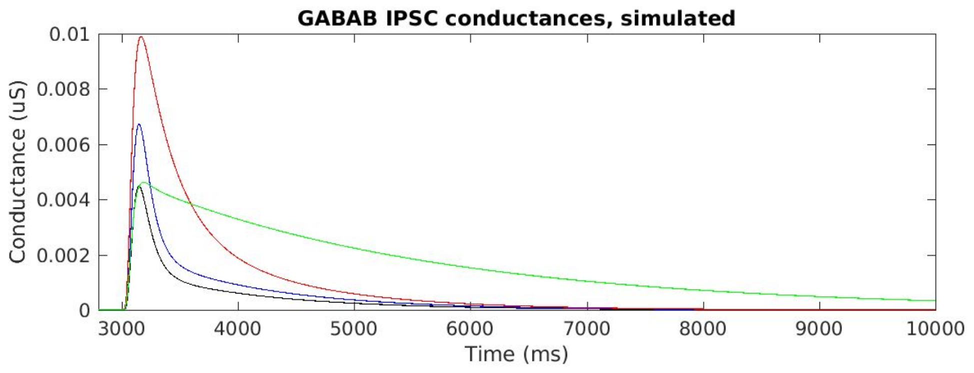

LTS probability
Real data
Simulated data
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Con
GAT1
GAT3
Dual
Pharm Condition

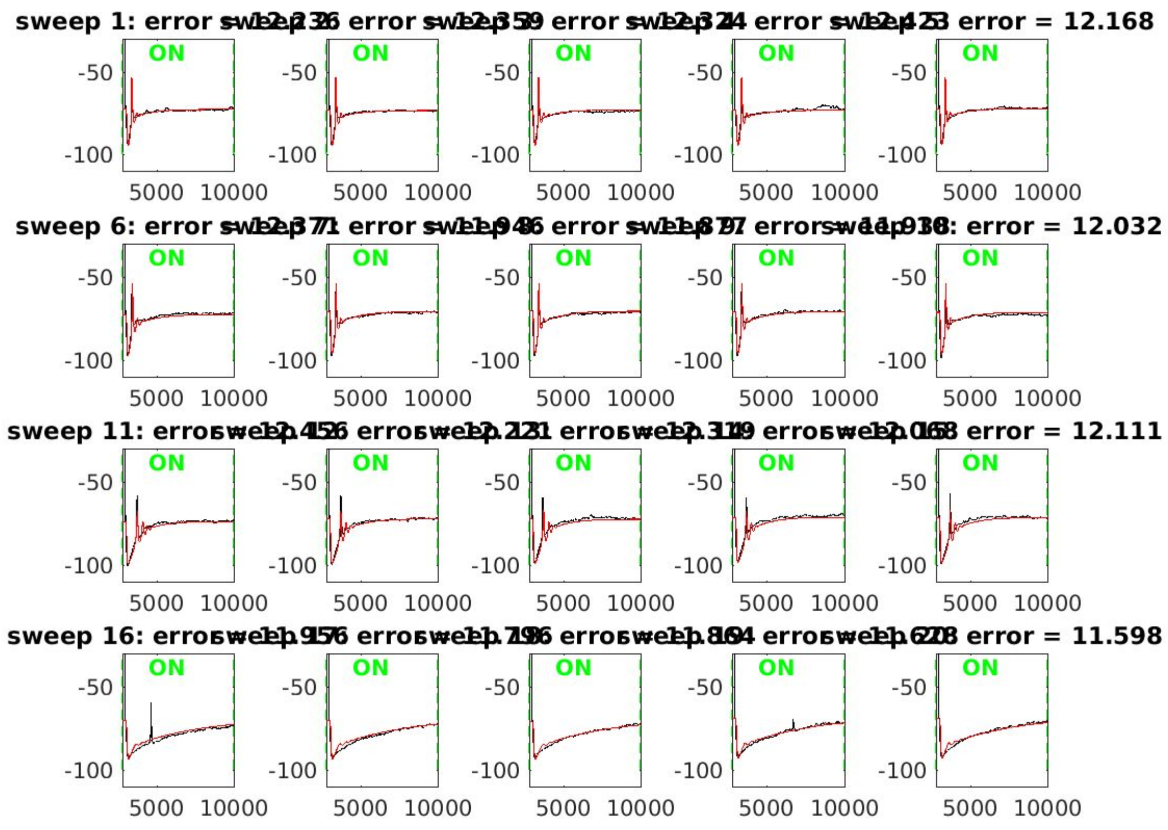

real data
simulated data
GABABi
m itGHK
h itGHK
m ih
ica
×10^-3
-50
-100
0
-0.1
-0.2
1
0.5
-1
-2
3000
4000
5000
6000
7000
8000
9000
10000


GABAB IPSC currents, recorded
Current (nA)
Time (ms)
0
-0.05
-0.1
-0.15
-0.2
3000
4000
5000
6000
7000
8000
9000
10000

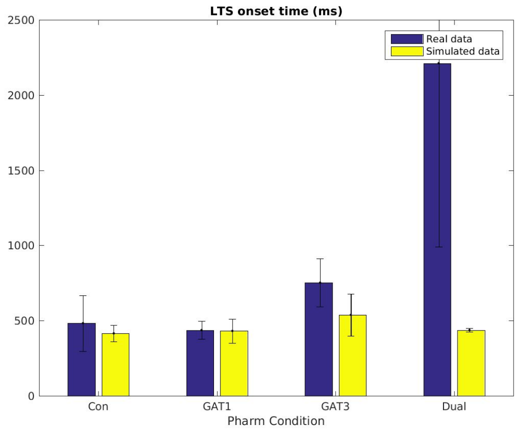

GABAB IPSC currents, simulated
Current (nA)
Time (ms)
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
3000
4000
5000
6000
7000
8000
9000
10000

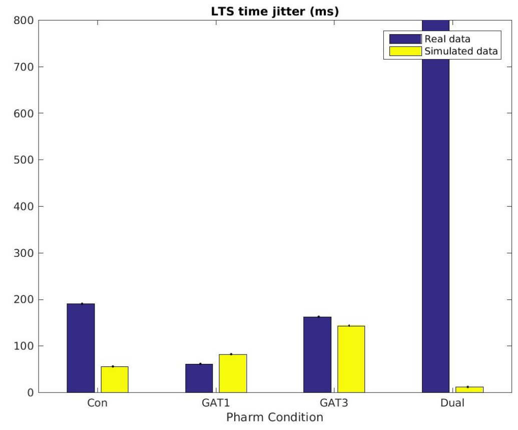

GABAB IPSC conductances, recorded
×10-3
Conductance (uS)
Time (ms)
10
8
6
4
2
0
3000
4000
5000
6000
7000
8000
9000
10000

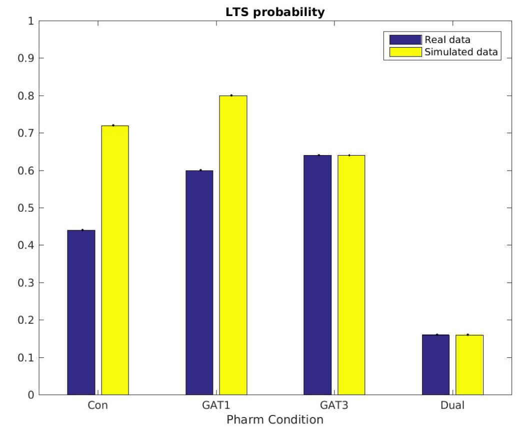

GABAB IPSC conductances, simulated
Conductance (uS)
Time (ms)
0.01
0.008
0.006
0.004
0.002
0
3000
4000
5000
6000
7000
8000
9000
10000


sweep 1: error = 12.236
sweep 2: error = 12.359
sweep 3: error = 12.324
sweep 4: error = 12.453
sweep 5: error = 12.168
sweep 6: error = 12.371
sweep 7: error = 11.946
sweep 8: error = 11.877
sweep 9: error = 11.938
sweep 10: error = 12.032
sweep 11: error = 12.456
sweep 12: error = 12.231
sweep 13: error = 12.319
sweep 14: error = 12.068
sweep 15: error = 12.111
sweep 16: error = 11.956
sweep 17: error = 11.796
sweep 18: error = 11.864
sweep 19: error = 11.628
sweep 20: error = 11.598
ON
-50
-100
5000
10000

2. All g incr -- is this physiological? Or maybe it doesn't matter as long as it is useful for model fitting.
   TO DO: Use only 100%, 200% & 400% g incr for fitting and for calculating LTS threshold.







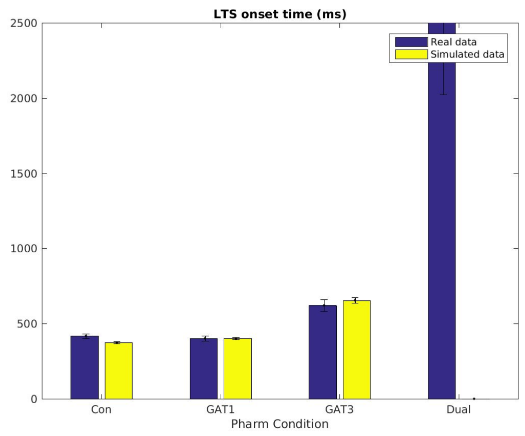
ii. Outputs (things to compare between simulation and reality):
   1. Voltage trace (throughout IPSC injection OR LTS peak only?) is probably sufficient
   2. Backup: LTS presence OR burst presence? LTS onset time OR burst onset time?

iii. Inputs (things that make each simulation different):
   1. IPSC current waveform (currently just 200% g incr; should I include all g incr?)
   2. Holding current (derived from holding potential from data). This cell only has data for holding potential around -70 mV, but for other cells, you could potentially pool data for all holding potentials together.
   3. Any other hidden trial-variable parameters?

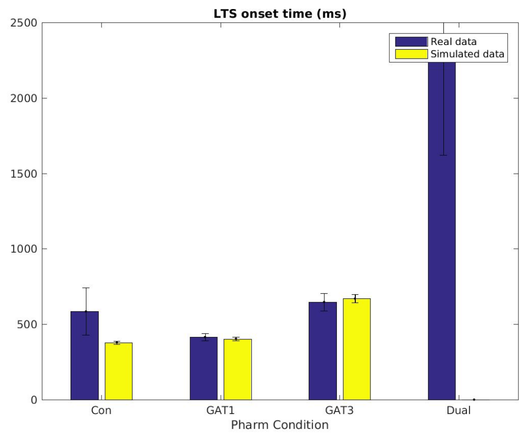
a. Perform Monte Carlo simulations for each parameter sequentially to find the range (infinite would mean impossible) necessary for reproducing the same range of outputs
    i. How to sample: Uniform distribution or Normal distribution?
b. For each parameter, fit to each trace individually using the above range. Then calculate the total error over all traces. Parameters that contribute to inter-trial variability will have a total error change percentage above some threshold.

iv. Assuming the fitted trial-variable parameters for each trace (treat as input), fit other parameters to update model for E091710.

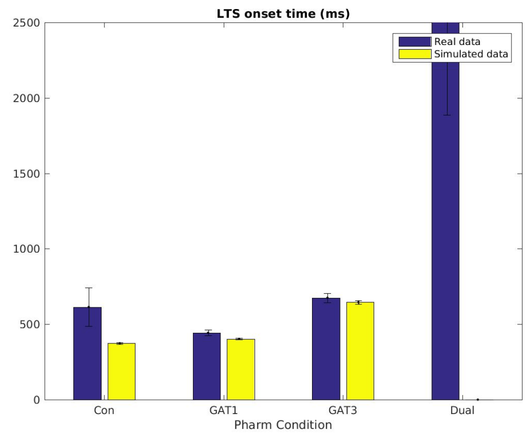
e. Inter-cell variability - determine **cell-variable parameters**
    i. Cell selection:
        1. LTSs present for all pharm conditions, 200% g incr: 13 cells; first possible trace



        2. Bursts present for all pharm conditions, 200% g incr: 8 cells; first possible trace



3. Bursts present for 3 out of 4 conditions, 200% g incr: 22 cells; burst/LTS presence prioritized



ii. Trace selection for each cell:
    1. G incr? Vhold? Currently 200% g incr & all possible Vhold
       TO DO: Use 100%, 200% & 400% g incr & all possible Vhold
    2. 1 trace per cell? First one(s) found? Random? OR
       5 traces per cell per condition (choose best Vhold)? OR
       15 traces per cell per condition (pool all Vhold)?
       TO DO: Pool all Vhold
    3. All? Only with LTSs? Only with bursts?
       TO DO: All

iii. Outputs (things to compare between simulation and reality):
    1. LTS probability OR burst probability?
       TO DO: LTS probability
    2. LTS onset time jitter OR burst onset time jitter?
       TO DO: LTS onset time jitter
    3. Mean LTS onset time OR mean burst onset time?
       TO DO: Mean LTS onset time.
    4. Mean LTS amplitude & mean LTS slope?
       TO DO: Probably if they predict burst probability or spikes per peak
    5. Voltage trace (throughout IPSC injection OR LTS peak only?) No.
    6. LTS presence OR burst presence? No.

iv. Inputs (things that make each simulation different):
    1. IPSC current waveform (see discussion above)
    2. Holding current (see discussion above)
    3. Trial-variable parameters (use mean of parameters fitted in E091710)
    4. Any other hidden cell-variable parameters?
        a. Perform Monte Carlo simulations for each parameter sequentially to find the range (infinite would mean

impossible) necessary for reproducing the same range of outputs
- i. How to sample: Uniform distribution or Normal distribution?

b. For each parameter, fit to each cell's data individually using the above range. Then calculate the total error over all cells. Parameters that contribute to inter-cell variability will have a total error change percentage above some threshold.

v. Assuming the fitted cell-variable parameters for each trace (treat as input), fit other parameters to update model for the TC neuron.

TO BE DISCUSSED NEXT TIME:

3. Network data/experiments:
    - a. How are the TC neurons activated? Are the RT neurons activated as well?
    - b. How can extracellular recordings be compared with simulated data?
4. Network model implementation/fitting:
    - a. Model implementation:
        - i. GABA_B synapses can be implemented with IPSC waveforms instead
        - ii. How to model RT neurons?
        - iii. Generate field recording? 3 Hz-5 kHz band-pass filter
    - b. Outputs:
        - i. Oscillation duration
        - ii. Oscillation period
        - iii. Anything else?
    - c. Inputs:
        - i. What parameters of the network can we extract from the real data?
            1. Percent of neurons initially activated
        - ii. Cell-variable parameters are varied as each neuron is created.
        - iii. Trial-variable parameters are treated as states that are stochastically varied throughout the simulation.
    - d. Parameters to fit:
        - i. Synapse weights?
        - ii. Synapse numbers?

9/5/2016

Notes from Scimemi 2014: Structure, function, and plasticity of GABA transporters

- The GABA transporter family:
    - Includes A1/GAT1, A13/GAT2, A11/GAT3, A12/BGT1 (transports **betaine**), A8/CT1 (transports **creatine**), and A6/TauT (transports **taurine**)
    - Part of the **solute carrier 6** (**SLC6**) family, which are **sodium symporters** and also includes amino acid & monoamine transporters
    - Nomenclature:

Human	rats	mice
A1	GAT1	GAT1
A11	GAT3	**GAT4**

-
    - Substrate specificity:
        - GAT3 accept substrates with a carboxyl group in the β-position and an amino group in the γ-position of their carbon backbone structure, like GABA and **β-alanine**
- Biophysical properties
    - Active process: requires **Na+ gradient**, presence of **extracellular Cl-** & voltage-dependent
    - Stoichiometric current:
        - Previously **1**GABA:**2**Na+:**1**Cl-
        - Now thought to be **1**GABA:**2**Na+, with rapid Cl- exchange
    - Michaelis-Menten constant: **2.5 μM** (3.1-10.6 μM for GAT1)
    - Turnover rate: estimates range from **2.5 s^{-1}** to **93 s^{-1}**
    - Other currents associated with **GAT1**:
        - **Agonist-induced Na+ inward** current
            - The GAT1 inhibitor **SKF89976A** selectively blocks this
            - May function as **negative feedback** (depolarization inhibits GABA uptake)
        - **Agonist-independent leak cationic** current (alkali ions)
    - GAT1 structure:
        - Structurally similar to LeuTAa, a prokaryotic leucine transporter that has been crystallized
        - **Multimeric**; each monomer transports GABA independently, but multimerization allows **trafficking** of GAT1 from the ER to the plasma membrane
        - Structural rearrangements of LeuTAa are consistent with an **alternating access** transport mechanism
- Expression of GABA transporters:
    - Distribution in the nervous system:

	GAT1	GAT2	GAT3
Intense labeling	cerebellum (molecular layer), basal ganglia (ventral pallidum, globus pallidus), olfactory bulb (glomerular layer), retina (inner nuclear layer), and interpeduncular nucleus		olfactory bulb (glomerular layer) and retina (inner nuclear layer)
Moderate labeling	throughout the neocortex (hippocampus proper and dentate gyrus), amygdala, septum, thalamus (**ventral lateral geniculate**, **reticular nuclei**), zona incerta, subthalamic nucleus, hypothalamus (suprachiasmatic and periventricular nuclei, anterior hypothalamic and pre-optic areas), superior colliculus, dorsal tegmental nuclei, basal ganglia (substantia nigra), nucleus of Darkschewitsch, pons and medulla (trapezoid, medial and lateral vestibular, dorsal cochlear and parabrachial nuclei, nucleus of the solitary tract and of the trigeminal nerve) and also in the spinal cord (dorsal horn laminae 1, 2, 4, 10)		septum (medial nucleus and vertical nucleus of the diagonal band), basal ganglia (ventral pallidum and globus pallidus), subfornical organ, amydala, thalamus (**paraventricular nucleus**, **lateral habenula**), superior colliculus, ventral tegmental nucleus, basal ganglia (substantia nigra pars compacta), and medial vestibular nucleus
Weak labeling	cerebellar Purkinje cells, deep cerebellar nuclei and also in the spinal cord (ventral horn)	leptomeninges	lateral reticular and parabrachial nuclei, deep cerebellar nuclei, spinal cord and in the entire neocortex

- Cell types:
  - In the **neocortex**:
    - GAT1:
      - Axon terminals of symmetrical synapses
      - Distal astrocytic processes
      - Dendrites and soma of post-synaptic neurons
    - GAT3:
      - In rodents, exclusively in **astrocytes**
      - In cats, monkeys and humans, in **astrocytes & oligodendrocytes**
  - In the **hippocampus**:

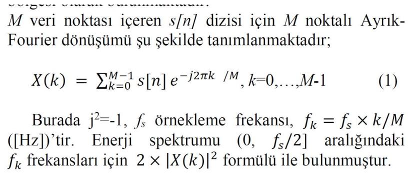
            - More in **stratum radiatum** than in stratum oriens interneurons
        - In the **cerebellum**:
            - No GAT1
            - **GAT3** expressed in **Bergmann glial cell** processes
        - In the **thalamus**:
            - Only expressed in **astrocytes**
            - **GAT1** located **closer to synaptic contacts** than **GAT3**, former mediates phasic synaptic transmission, latter controls tonic inhibition
            - Distribution in the cell membrane:
        - GAT1 density: 500-800 μm^{-2}
        - Highly dynamic & **activity-dependent**
        - Cell-to-cell variability: *topic of research*
- Functional role of GABA transporters:
    - GAT1 & GAT3 regulate different signaling pathways (Song et al., 2013; Kersante et al., 2013)
    - Inhibition of GABA uptake prolongs the neuronal response to iontophoretic GABA applications & to repetitive synaptic stimulations
    - GABA **diffusion** shape the profile of evoked IPSCs
    - GABA **uptake** limits GABA escape from active synapses
    - May maintain a **spatial separation** between somatic and dendritic GABAergic signals (*topic of research*)
    - **Tonic, extrasynaptic GABA_A currents** (which are regulated by GAT3) controls the onset of **neural network oscillations** in the thalamus (Rovo et al., 2014)
- Regulation of GABA transporters:
    - There are **intracellular signaling cascades** that regulate the **cytoplasm-to-surface partitioning** (Corey et al., 1994; Whitworth and Quick, 2001)
        - In Xenopus oocytes, activating **PKC** or inhibiting **PP2B enhances** GABA uptake via **GAT1**, by altering trafficking and changing the surface-to-cytoplasm expression (V_max) and not by changing the binding affinity (K_m)
        - However, in rat cultures, activating PKC does not alter GABA uptake in neurons and **decreases** it in glial cells.
        - In rats, **Tyrosine kinase** phosphorylation (by **BDNF** for instance) reduces internalization of the GABA transporter => increase in GABA uptake
    - Interaction with components of the **cytoskeleton** control the mobility within the cell membrane (Imoukhuede et al., 2009)
        - Disrupting the interaction between GAT1 and the **actin cytoskeleton** (by **Pals1** & **ezrin**) increases GABA uptake
    - Increasing **lateral mobility** of the transporters **increases GABA diffusion**

<u>Plan for next week</u>

1. Record the reason for some abf files to be declared broken
2. Compare conductance peak with theoretical peak to find IPSC offset
3. Compare recorded conductance & current traces with the corresponding theoretical traces
4. Try a positive 2nd derivative threshold for spontaneous LTSs (see C092810_0003_10). Sort all remaining questionable traces into different directories and present all of them the next time we meet. Remove all problematic traces, along with the all traces from the same cell-pharm-gincr set from data analysis
5. Instead of calculating the input resistance and using it, fit the model to the current injection responses to extract parameters relevant to the input resistance
6. Change median filter widths & moving-average filter widths and see whether the burst onset times change significantly
7. Perform ANOVA to determine whether there are significant changes
8. Keep all channels of Amarillo et al except HH. Analyze the correlation between the burst probability, spikes per peak, spikes per burst, etc. and the slope or other features of an LTS. Use the features to predict spikes (**burst threshold** & **spikes per burst**) in the network model.
9. Use only 100%, 200% & 400% g incr for fitting and for calculating LTS threshold.
10. Perform Monte Carlo simulation for all parameters simultaneously to find maximal range of LTS onset time
11. Adjust ranges of parameters OR update model to reproduce maximal range of LTS onset time found in data

6/13/2016

Past Research on Spike-Wave Detection Algorithms in Rodents

Author (Lab) Title	**Year**	**Model (data type)**	**Approach**	**Performance**
Van Luijtelaar et al (Van Luijtelaar) "Methods of automated absence seizure detection, interference by stimulation, and possibilities for prediction in genetic absence models"	2016	NIL	Review article	NIL
Bauquier et al (Cook) "Evaluation of an automated spike-and-wave complex detection algorithm in the EEG from a rat model of absence epilepsy"	2015	**GAERS** rats (**ECoG**)	Implemented in Matlab w/ Signal Processing Toolbox: 1. Bandpass filtered (2nd order Butterworth filter) between **3~30 Hz** in both reverse and forward directions 2. SWD detected via: a manually-set threshold, # of spike rises between 5~13 within a 1 s window, ISI: 40~300 ms Note: during EEG recordings, if the rats were perceived as being asleep and after confirmation of no seizure activity, noise stimuli of 94 and 98 dB were delivered	Sensitivity: **95±8%** Specificity: **96±3%** PPV: **94±5%** NPV: **97±2%**
Şayli (?) "Detection of spike-wave discharges in the EEG signals of WAG/Rij rats"	2015	**WAG/Rij rats** (**ECoG**)	Fourier transform, implemented in Matlab:  M veri noktası içeren $s[n]$ dizisi için M noktalı Ayrık-Fourier dönüşümü şu şekilde tanımlanmaktadır; $X(k) = \sum_{k=0}^{M-1} s[n]\, e^{-j2\pi k\ /M}, k=0,\ldots,M-1$ (1) Burada $j^2=-1$, f_s örnekleme frekansı, $f_k = f_s \times k/M$ ([Hz])'tir. Enerji spektrumu $(0,\ f_s/2]$ aralığındaki f_k frekansları için $2 \times \lvert X(k)\rvert^2$ formülü ile bulunmuştur.	No data?

Richard et al (Frankel) "SWDreader: a wavelet-based algorithm using spectral phase to characterize spike-wavemorphological variation in genetic models of absence epilepsy"	2015	4 different **mouse** strains (**ECoG**)	Not only detects SWD but also characterizes its **morphology**. Uses a **Morlet wavelet function** combined with **spectral phase analysis**	WIth specificity **> 90%**, sensitivity depends on strain: **94.2±9.4%**, **87.0±17.1%**, **98.8±1.5%**, **93.7±5.6%**, respectively
Petersen et al (Sorensen) "Automatic characterization of dynamics in Absence Epilepsy"	2013	Human (EEG)	Continuous wavelet transform with a **Morlet wavelet function**: : $W_{a,b} = \frac{1}{\sqrt{\lvert a \rvert}} \cdot \sum_n s(n)\psi(\frac{n-b}{a})$ $\psi(x) = e^{\frac{x^2}{2}} \cdot cos(5x)$ $\psi(x) = x \cdot e^{\frac{x^2}{2}} \cdot cos(5x)$	No data
Ovchinnikov et al (van Luijtelaar) "An algorithm for real-time detection of spike-wave discharges in rodents"	2010	**WAG/Rij** rats (**ECoG**)	Real-time continuous wavelet transform with a **Morlet wavelet function**: $\phi(\eta) = \frac{1}{\sqrt[4]{\pi}} e^{j\omega_0 \eta} e^{-(\eta^2/2)}$	Sensitivity: **100%** Mean PPV: **96.6%**
Xanthopoulos et al (Pardalos) "A Novel Wavelet Based Algorithm for Spike and Wave Detection in Absence Epilepsy"	2010	Human (EEG)	Wavelet decomposition with a **Morlet wavelet function**	Sensitivity: **97.25%**
Xanthopoulos et al (Pardalos) "A robust spike and wave algorithm for detecting seizures in a genetic absence seizure model"	2009	**Fischer 334** rats	Adapted from a human EEG detection algorithm. Wavelet decomposition with a **Morlet wavelet function**	Sensitivity: **>90%**

Pan et al (Mamun) "Detection of Epileptic Spike-Wave Discharges Using SVM"	2007	**WAG/Rij** rats (**two-chanel EEG**)	**Support vector machine** (SVM)	PPV: **100%**
Jacquin et al (John) "Automatic identification of spike-wave events and non-convulsive seizures with a reduced set of electrodes"	2007	Human EEG	A combination of **wavelet** and **non-linear (fractal)** methods	Sensitivity: **83%** Specificity: **96%**
Van Hese et al (Van de Walle) "Detection of spike and wave discharges in the cortical EEG of genetic absence epilepsy rats from Strasbourg"	2003	**GAERS** rats (**ECoG**)	Short-time Fourier transform + Spectral-comb analysis	EER: **83~96%** With specificity > 80%, sensitivity: **~90%**
Fanselow et al	2000		Maximum **absolute value** of the EEG amplitude	
Westerhuis et al	1996		**Steepness** of the EEG signal	

PPV = positive predictive value
NPV = negative predictive value
EER = equal error rate (sensitivity = specificity)

6/12/2016~6/19/2016

Details of the Destexhe et al 1996 model

- Overview of all files

File name	Called by	Procedures/templates/mechanisms included	Notes
mosinit.hoc	None		

rundemo.hoc	mosinit.hoc	make_demopanel()
		destroy_elec()
		restart()
			Menu showing the 6 possible simulations

Fspin.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
		text()
			A **spindle** oscillation in the 4-cell circuit (**short** time scale)

FspinL.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
		text()
			A **spindle** oscillation in the 4-cell circuit (**long** time scale)

Fdelta.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
		text()
			Delta oscillations in the 4-cell circuit (**short** time scale)

FdeltaL.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
		text()
			Delta oscillations in the 4-cell circuit (**long** time scale)

Fbic.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
			Bicuculline-induced oscillations in the 4-cell circuit (**short** time scale)

		text()	

FbicL.oc	rundemo.hoc	addgraph()
		addtext()
		addline()
		ncon()
		assign_synapses()
		text()
			Bicuculline-induced oscillations in the 4-cell circuit (**long** time scale)

TC.tem	Fspin.oc FspinL.oc Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc	*Class:* **sTC**	Template file for defining **thalamocortical neurons**
RE.tem	Fspin.oc FspinL.oc Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc	*Class:* **sRE**	Template file for defining **reticular neurons**
ampa.mod	Fspin.oc FspinL.oc Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc	*Point Process:* **AMPA_S**	Simple model for glutamate AMPA receptors
gabaa.mod	Fspin.oc FspinL.oc Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc	*Point Process:* **GABAa_S**	Simple model for GABAa receptors
gabab.mod	Fspin.oc FspinL.oc Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc	*Point Process:* **GABAb_S**	Kinetic model of GABA-B receptors
kleak.mod	TC.tem	*Point Process:* **kleak**	Leak potassium current
cadecay.mod	TC.tem	*Mechanism:* **cad**	Fast mechanism for submembranal

	RE.tem		Ca++ concentration
HH2.mod	TC.tem RE.tem	*Mechanism:* **hh2**	Fast Na+ and K+ currents responsible for action potentials
Ih.mod	TC.tem	*Mechanism:* **iar**	**Anomalous Rectifier Ih** - cation (Na/K) channel in thalamocortical neurons
IT.mod	TC.tem	*Mechanism:* **it**	Ca++ current responsible for low threshold spikes (**LTS**) in **thalamocortical neurons**
IT2.mod	RE.tem	*Mechanism:* **it2**	Ca++ current responsible for low threshold spikes (**LTS**) in **reticular neurons**

- **mosinit.hoc**
    - Loads "nrngui.hoc" and ”rundemo.hoc"

- **rundemo.hoc**
    - Procedures

Name & Arguments	Description	Called by
make_demopanel()	Creates a panel containing buttons to start each of the 6 possible demos	
destroy_elec()	Destroys the objects referenced by “**El.stim**” and “**El.vc**”	
restart($o1**)**	1. Destroys electrodes if present 2. Destroys all sections (calls **delete_section()**) 3. Destroys all graphs in **graphList** 4. Removes all graphs in **flush_list** & **fast_flush_list** 5. Close all windows in the Print & File Window Manager 6. Updates all panels 7. Set stoprun = 0, cvode_active = 0 8. Loads the file named “$o1.oc”	make_demopanel()

- Variables

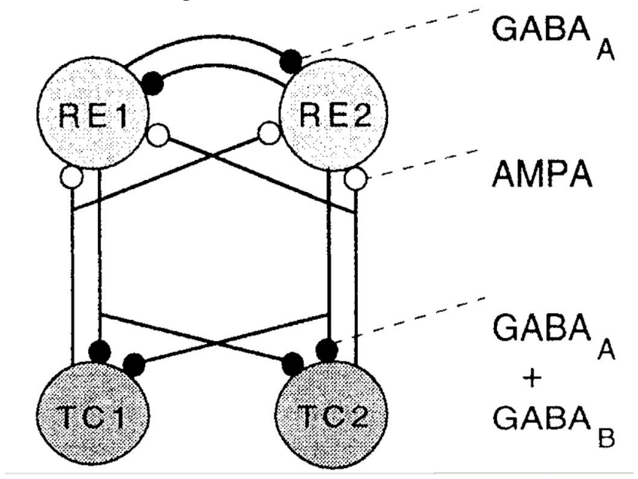
Name	Description	Initialization	Used by
ismenu	Whether **nrncontrolmenu()** is called	0	Fspin.oc FspinL.oc

			Fdelta.oc FdeltaL.oc Fbic.oc FbicL.oc
electrodes_present	Whether electrodes are created (However, I can't find where it's ever set to 1)	0	restart()

- Objects

Name	Description	Class	Used by
tstr	Stores names of files	String	restart()
El	An object that contains "**stim**" & "**vc**" as member objects		destroy_elec()

- **Fspin.oc**
  - A **spindle** oscillation in the 4-cell circuit (**short** time scale)
  - Network configuration:



  - Reproduces **Figure 7** of the paper
  - Creates a maximum of **20** graphs
  - Geometry of each type of cell is set up by loading **TC.tem** & **RE.tem**
  - Network parameters/objects:

Name	Description	Default value/Class	Notes
ncells	Number of cells in the same layer	**2**	
TC[ncells]	TC neurons	**sTC**	

RE[ncells]	RE neurons	**sRE**	
diamTCRE	Diameter of connectivity for TC->RE	**1**	
diamRERE	Diameter of connectivity for RE->RE	**1**	
diamRETC	Diameter of connectivity from RE->TC	**1**	
reG[ncells] [nTCRE]	AMPA-mediated synapses from TC->RE Total synaptic conductance = **0.2 µS**	**AMPA_S**	
reA[ncells] [nRERE]	GABA_A-mediated synapses from RE->RE Total synaptic conductance = **0.2 µS**	**GABAa_S**	
tcA[ncells] [nRETC]	GABA_A-mediated synapses from RE->TC Total synaptic conductance = **0.02 µS**	**GABAa_S**	
tcB[ncells] [nRETC]	GABA_B-mediated synapses from RE->TC Total synaptic conductance = **0.04 µS**	**GABAb_S**	

- Whenever a TC cell and an RE cell are within **diamTCRE** of each other, an AMPA-mediated synapse is created at **soma(0.5)** of the RE cell, with the presynaptic variable is set to **soma.v** of the TC cell. These synapses have the following parameter values:

Name	Description	Default value	Notes
Alpha_AMPA_S	Forward (binding) rate [$ms^{-1}mM^{-1}$]	**0.94**	
Beta_AMPA_S	Reverse (unbinding) rate [ms^{-1}]	**0.18**	
Cmax_AMPA_S	Maximum transmitter concentration [mM]	**0.5**	
Cdur_AMPA_S	Transmitter release duration (rising phase) [ms]	**0.3**	
Erev_AMPA_S	Reversal potential [mV]	**0**	

- Whenever two RE cells are within **diamRERE** of each other, a GABA_A-mediated synapse is created at **soma(0.5)** of one RE cell, with the presynaptic variable is set to **soma.v** of the other RE cell. These synapses have the following parameter values:

Name	Description	Default value	Notes
Alpha_GABAa_S	Forward (binding) rate [$ms^{-1}mM^{-1}$]	**20**	

Beta_GABAa_S	Reverse (unbinding) rate [ms^{-1}]	**0.162**	
Cmax_GABAa_S	Maximum transmitter concentration [mM]	**0.5**	
Cdur_GABAa_S	Transmitter release duration (rising phase) [ms]	**0.3**	
Erev_GABAa_S	Reversal potential [mV]	**-80**	

- Whenever an RE cell and a TC cell are within **diamRETC** of each other, a GABA_A-mediated synapse and a GABA_B-mediated synapse is created at **soma(0.5)** of the TC cell, with the presynaptic variable is set to **soma.v** of the RE cell. These synapses have the following parameter values:

Name	Description	Default value	Notes
Alpha_GABAa_S	Forward (binding) rate [$ms^{-1}mM^{-1}$]	**20**	
Beta_GABAa_S	Reverse (unbinding) rate [ms^{-1}]	**0.162**	
Cmax_GABAa_S	Maximum transmitter concentration [mM]	**0.5**	
Cdur_GABAa_S	Transmitter release duration (rising phase) [ms]	**0.3**	
Erev_GABAa_S	Reversal potential [mV]	**-85**	Rinzel's Erev
K1_GABAb_S	Forward (binding) rate [$ms^{-1}mM^{-1}$]	**0.09**	
K2_GABAb_S	Reverse (unbinding) rate [ms^{-1}]	**0.0012**	
K3_GABAb_S	Rate of G-protein production [ms^{-1}]	**0.18**	
K4_GABAb_S	Rate of G-protein decay [ms^{-1}]	**0.034**	
KD_GABAb_S	Dissociation constant of K+ channel [1]	**100**	
n_GABAb_S	Number of binding sites of G-proteins on the K+ channel	**4**	
Erev_GABAb_S	Reversal potential [mV]	**-95**	
Cmax_GABAb_S	Maximum transmitter concentration [mM]	**0.5**	
Cdur_GABAb_S	Transmitter release duration (rising phase) [ms]	**0.3**	

- Model alterations specific to **spindle generation**:

Name	Description	New value	Default value

TC[1].soma.ghbar_iar	Maximum H current conductance [S/cm²]	**1.5e-5**	2e-5
TC[1].kl.gmax	Conductance of potassium leak current [μS]	**0.003**	0.004
TC[0].soma.ghbar_iar	Maximum H current conductance [S/cm²]	**2e-5**	2e-5
TC[0].kl.gmax	Conductance of potassium leak current [μS]	**0.005**	0.004

- Simulation parameters:

Name	Description	Default value	Notes
Dt	Time interval between points plotted [ms]	**1**	
npoints	Number of points plotted	**20000**	
dt	Time step of integration [ms]	**0.1**	
trans	Sets the start time [ms]	0	
tstart	Start time to plot [ms]	trans	
tstop	End time to plot [ms]	trans + npoints*Dt	
runStopAt		tstop	
steps_per_ms	Points plotted per ms	1/Dt	
celsius	Temperature of experiment [℃]	**36**	
v_init	Resting membrane potential [mV]	**-70**	

- Other Global variables/objects:

Name	Description	Type/ Class	Notes
g[i]	Graphs (including text graphs)	Graph	
ngraph	Number of graphs created so far	float	
text_id	Index of the last text graph created	float	
ismenu	Whether a control panel is created	float	
nTCRE	Maximum # of RE cells receiving synapses from one TC cell	float	

nRERE	Maximum # of RE cells receiving synapses from one RE cell	float	
nRETC	Maximum # of TC cells receiving synapses from one RE cell	float	
nv	# of synapses made so far	float	
Sim	Vector of simulation points	Vector	
gtxt	Stores strings	String	

- Functions/procedures:

Name & Arguments	Description	Called by
addgraph("*variable*", minvalue, maxvalue)	1. Creates a graph with axes == [**tstart**,**tstop**,**$2**,**$3**], variable == **$s1**, color index == **1** (**black**), brush index == 0 2. Saves the graph to **graphList[0]**	
addtext("*text*")	1. Creates a text graph and adds the string *text* to the position **(0.1 0.8)**. 2. Saves the graph to **graphList[0]**	
addline("*text*")	Adds the string *text* under the previous added line.	
ncon(*interaction diameter*)	Returns the number of connections to be made from a cell based on the interaction diameter (currently **2*ID+1**)	To compute **nTCRE, nRERE, nRETC**
assign_synapses ($1, $2, $3, $4)	Assigns synaptic conductances by distributing the total conductances $1, $2, $3, $4 over all **RE->RE**, **RE->TC (GABA_A)**, **RE->TC (GABA_B)**, **TC->RE** synapses respectively.	
text()	Creates a new window containing model parameters	

- Creates the following graphs:
  - Voltage vs. time for **soma.v(0.5)** of each of the four cells
  - A text graph printing out model parameters

- **FspinL.oc**
  - A **spindle** oscillation in the 4-cell circuit (**long** time scale)
  - Code exactly the same as Fspin.oc except that **npoints** is changed to **60000**.

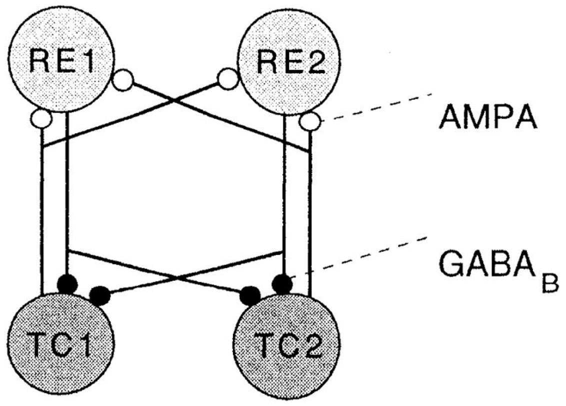
- **Fdelta.oc**

- **Delta** oscillations in the 4-cell circuit (**short** time scale)
- Network configuration same as Fspin.oc
- Code exactly the same as Fspin.oc except:

Name	Description	New value	Default value
TC[0].soma.ghbar_iar	Maximum H current conductance [S/cm²]	**1e-5**	2e-5
TC[0].kl.gmax	Conductance of potassium leak current [μS]	**0.005**	0.004
ginc_iar	Augmentation of conductance associated with the Ca++ bound state [1]	**1**	2

- **FdeltaL.oc**
  - **Delta** oscillations in the 4-cell circuit (**long** time scale)
  - Code exactly the same as Fdelta.oc except that **npoints** is changed to **60000**.

- **Fbic.oc**
  - **Bicuculline**-induced oscillations in the 4-cell circuit (**short** time scale)
  - Network configuration:



  - Reproduces **Figure 8** of the paper
  - Code exactly the same as Fspin.oc except that total synaptic conductance of GABA_A-mediated synapses is set to **0 μS**.

- **FbicL.oc**
  - **Bicuculline**-induced oscillations in the 4-cell circuit (**long** time scale)
  - Code exactly the same as Fbic.oc except that **npoints** is changed to **60000**.

- **TC.tem**
  - Template for a single-compartment **TC** neuron
  - Geometry:

	nseg	Length [μm]	Diameter [μm]	Specific capacitance [μF/cm²]	Cytoplasmic resistivity [Ω·cm]

Soma	**1**	**96**	**96**	**1**	**100**

- Mechanisms inserted:
  - **pas**
  - **hh2**
  - **it**
  - **iar**
  - **cad**
- Point processes inserted:
  - kl = new **kleak**()
  - **soma** kl.loc(**0.5**)
- Parameters:

Name	Description	Default value	Notes
e_pas	Reversal potential of nonspecific leak current [mV]	**-70**	From Rinzel
g_pas	Conductance of nonspecific leak current [S/cm²]	**1e-5**	
Erev_kleak	Reversal potential of potassium channel [mV]	**-100**	
kl.gmax	Conductance of potassium leak current [μS]	**0.004**	
ek	Reversal potential of potassium channel [mV]	**-100**	
ena	Reversal potential of sodium channel [mV]	**50**	
vtraub_hh2	Threshold v_T [mV]	**-25**	high
gnabar_hh2	Maximum sodium conductivity [S/cm²]	**0.09**	
gkbar_hh2	Maximum potassium conductivity [S/cm²]	**0.01**	
cai	Calcium concentration inside the cell [mM]	**2.4e-4**	
cao	Calcium concentration outside the cell [mM]	**2**	
eca	Reversal potential of calcium channel [mV]	**120**	
gcabar_it	Maximum calcium conductance [S/cm²]	**0.002**	
eh	Reversal potential for H current [mV]	**-40**	
nca_iar	Number of binding sites of Ca++ [1]	**4**	
k2_iar	Inverse of the real time constant of the binding of Ca++ to the CB protein [ms^{-1}]	**0.0004**	

cac_iar	Half-activation (p0 = p1) of calcium dependence of CB protein (=(k2/k1)^(1/nca)) [mM]	**0.002**	
nexp_iar	Number of binding sites on Ih channels [1]	**1**	
k4_iar	Inverse of the real time constant of the binding of the CB protein to Ih channels (it basically governs the interspindle period) [ms^{-1}]	**0.001**	
Pc_iar	Half-activation (o1 = o2) of CB protein dependence of I_h channels (=(k4/k3)^(1/nexp)) [1]	**0.01**	
ginc_iar	Augmentation of conductance associated with the Ca++ bound state [1]	**2**	
ghbar_iar	Maximum H current conductance [S/cm²]	**2e-5**	
depth_cad	Depth of the shell just beneath the membrane [µm]	**1**	
taur_cad	Time constant of calcium extrusion, must be fast) [ms]	**5**	
cainf_cad	Equilibrium concentration of calcium [mM]	**2.4e-4**	
kt_cad	Dummy parameter specific to the calcium pump	**0**	no pump

- **RE.tem**
  - Template for a single-compartment **RE** neuron
  - Geometry:

	nseg	Length [µm]	Diameter [µm]	Specific capacitance [µF/cm²]	Cytoplasmic resistivity [Ω·cm]
Soma	**1**	**64.86**	**70**	**1**	**100**

  - Mechanisms inserted:

    pas
    hh2
    it2
    cad
  - Parameters:

Name	Description	Default value	Notes
e_pas	Reversal potential of leak current [mV]	**-90**	
g_pas	Conductance of leak current [S/cm²]	**5e-5**	
ek	Reversal potential of potassium channel [mV]	**-100**	

ena	Reversal potential of sodium channel [mV]	**50**	
vtraub_hh2	Threshold v_T [mV]	**-55**	
gnabar_hh2	Maximum sodium conductivity [S/cm²]	**0.2**	
gkbar_hh2	Maximum potassium conductivity [S/cm²]	**0.02**	
cai	Calcium concentration inside the cell [mM]	**2.4e-4**	
cao	Calcium concentration outside the cell [mM]	**2**	
eca	Reversal potential of calcium channel [mV]	**120**	
shift_it2	Shift towards hyperpolarization of *both activation & inactivation curves* [mV]	**2**	
qm_it2	Q_10 for activation curve [1]	**2.5**	low
qh_it2	Q_10 for inactivation curve [1]	**2.5**	
gcabar_it2	Maximum calcium conductance [S/cm²]	**0.003**	strong
depth_cad	Depth of the shell just beneath the membrane [µm]	**1**	
taur_cad	Time constant of calcium extrusion, must be fast) [ms]	**5**	
cainf_cad	Equilibrium concentration of calcium [mM]	**2.4e-4**	
kt_cad	Dummy parameter specific to the calcium pump	**0**	no pump

- **ampa.mod**
  - Simplified model for glutamate AMPA receptors
  - Whole-cell recorded postsynaptic currents mediated by AMPA/Kainate receptors (Xiang et al., J. Neurophysiol. 71: 2552-2556, 1994) were used to estimate the parameters of the present model; the fit was performed using a simplex algorithm (see Destexhe et al., J. Computational Neurosci. 1: 195-230, 1994).
  - Implemented as a Point Process: **AMPA_S()**
  - Parameters:

Name	Description	Default value	Type
Cmax	Maximum transmitter concentration [mM]	**0.5**	global
Cdur	Transmitter release duration (rising phase) [ms]	**0.3**	global
Alpha	Forward (binding) rate [$ms^{-1}mM^{-1}$]	**0.94**	global

Beta	Reverse (unbinding) rate [ms^{-1}]	**0.18**	global
Erev	Reversal potential [mV]	**0**	global
Prethresh	Voltage level necessary for release [1]	**0**	global
Deadtime	Minimum time between release events [ms]	**1**	global
gmax	Maximum conductance [μS]		range

- Other variables visible to hoc:

Name	Description	Default value	Type
C	Transmitter concentration [mM]	0	range
R	Fraction of channels open [1]	0	range
R0	Fraction of channels open at start of release [1]		range
R1	Fraction of channels open at end of release [1]	0	range
g	Conductance [μS]		range
lastrelease	Time of last release [ms]	-9e9	range
TimeCount	Time until the current release will end [ms]	-1	range
Rinf	Steady state of fraction of channels open [1]	$\frac{\alpha C_{max}}{\alpha C_{max}+\beta}$ =**0.72**	global
Rtau	Time constant of channel binding [ms]	$\frac{1}{\alpha C_{max}+\beta}$ =**1.54**	global
pre	Pointer to presynaptic variable		pointer

- Equations:
  - If transmitter is being released (C == Cmax):

$$R = R_{inf} + (R_0 - R_{inf})e^{-(t-t_{lr})/R_{tau}}$$

    If transmitter is not present (C == 0):

$$R = R_1 e^{-\beta(t-(t_{lr}+C_{dur}))}$$

  - Update conductance and current

$$g = g_{max}R$$

$$I = g(V - E_{rev})$$

- Procedures and functions:

Name & Arguments	Description	Called by
release()	1. If it is at least **Deadtime** after the last release event ended, and if the presynaptic variable is greater than **Prethresh**, a new release event is started. 2. The transmitter concentration **C** is **Cmax** during a release and **0** otherwise. 3. Update fraction of channels open **R** based on above equations	BREAKPOINT
exptable(x)	Returns **exp(x)** from a table if **-10 < x < 10** (precision **0.01**); Returns **0** otherwise.	release()

- **gabaa.mod**
  - Simplified model for GABAa receptors
  - Whole-cell recorded GABA-A postsynaptic currents (Otis et al, J. Physiol. 463: 391-407, 1993) were used to estimate the parameters of the present model; the fit was performed using a simplex algorithm (see Destexhe et al., J. Neurophysiol. 72: 803-818, 1994).
  - Implemented as a Point Process: **GABAa_S()**
  - Parameters:

Name	Description	Default value	Type
Cmax	Maximum transmitter concentration [mM]	**0.5**	global
Cdur	Transmitter release duration (rising phase) [ms]	**0.3**	global
Alpha	Forward (binding) rate [$\text{ms}^{-1}\text{mM}^{-1}$]	10.5	global
Beta	Reverse (unbinding) rate [ms^{-1}]	0.166	global
Erev	Reversal potential [mV]	**-80**	global
Prethresh	Voltage level necessary for release [1]	**0**	global
Deadtime	Minimum time between release events [ms]	**1**	global
gmax	Maximum conductance [μS]		range

  - Other variables visible to hoc:

Name	Description	Default value	Type
C	Transmitter concentration [mM]	0	range

R	Fraction of channels open [1]	0	range
R0	Fraction of channels open at start of release [1]		range
R1	Fraction of channels open at end of release [1]	0	range
g	Conductance [μS]		range
lastrelease	Time of last release [ms]	-9e9	range
TimeCount	Time until the current release will end [ms]	-1	range
Rinf	Steady state of fraction of channels open [1]	$\frac{\alpha C_{max}}{\alpha C_{max}+\beta}$ =**0.97**	global
Rtau	Time constant of channel binding [ms]	$\frac{1}{\alpha C_{max}+\beta}$ =**0.18**	global
pre	Pointer to presynaptic variable		pointer

- Equations (same as ampa.mod):
  - If transmitter is being released (C == Cmax):

    $$R = R_{inf} + (R_0 - R_{inf})e^{-(t-t_{lr})/R_{tau}}$$

    If transmitter is not present (C == 0):

    $$R = R_1 e^{-\beta(t-(t_{lr}+C_{dur}))}$$

  - Update conductance and current

    $$g = g_{max}R$$

    $$I = g(V - E_{rev})$$

- Procedures and functions (same as ampa.mod):

Name & Arguments	Description	Called by
release()	1. If it is at least Deadtime after the last release event ended, and if the presynaptic variable is greater than Prethresh, a new release event is started. 2. The transmitter concentration C is Cmax during a release and 0 otherwise. 3. Update fraction of channels open R based on above equations	BREAKPOINT

exptable(x)	Returns exp(x) from a table if -10 < x < 10 (precision 0.01); Returns 0 otherwise.	release()

- **gabab.mod**
  - Kinetic model of GABA-B receptors
  - 2nd-order G-protein transduction => Cooperative binding of G-proteins to K+ channel => fast K+ channel opening
  - Features:
    - Peak at 100 ms; time course fit to Tom Otis' PSC
    - SUMMATION (psc is much stronger with bursts)
  - Approximations:
    - Single binding site on receptor
    - Alpha G-protein directly activates K+ channel
    - G-protein dynamics is **second-order**; simplified as follows:
      - Saturating receptor
      - No desensitization
      - Michaelis-Menten of receptor for G-protein production
      - "Resting" G-protein is in excess
      - Quasi-stat of intermediate enzymatic forms
    - Binding of G-proteins on K+ channel is fast, reaching steady-state based on Michaelis-Menten kinetics instantaneously
  - Parameters estimated from patch clamp recordings of GABAB PSP's in rat hippocampal slices (Otis et al, J. Physiol. 463: 391-407, 1993).
  - Implemented as a Point Process: **GABAb_S()**
  - Parameters:

Name	Description	Default value	Type
Cmax	Maximum transmitter concentration [mM]	**0.5**	global
Cdur	Transmitter release duration (rising phase) [ms]	**0.3**	global
K1	Forward (binding) rate [$ms^{-1}mM^{-1}$]	0.52	global
K2	Reverse (unbinding) rate [ms^{-1}]	0.0013	global
K3	Rate of G-protein production [ms^{-1}]	0.098	global
K4	Rate of G-protein decay [ms^{-1}]	0.033	global
KD	Dissociation constant of K+ channel [1]	**100**	global
n	Number of binding sites of G-proteins on the K+ channel	**4**	global
Erev	Reversal potential [mV]	**-95**	global
Prethresh	Voltage level necessary for release [1]	**0**	global

Deadtime	Minimum time between release events [ms]	**1**	global
gmax	Maximum conductance [μS]		range

- Other variables visible to hoc:

Name	Description	Default value	Type
C	Transmitter concentration [mM]	0	range
g	Conductance [μS]		range
lastrelease	Time of last release [ms]	-9e9	range
TimeCount	Time until the current release will end [ms]	-1	range
pre	Pointer to presynaptic variable		pointer

- States & initialization:

Name	Description	Initialization
R	Fraction of activated receptors [1]	0
G	Fraction of activated G-proteins [1]	0

- Equations:
  - Solve for states **R** & **G** based on the transmitter concentration [T] (C in the code) and the following differential equations:

$$\frac{dR}{dt} = K_1[T](1-R) - K_2R$$

$$\frac{dG}{dt} = K_3R - K_4G$$

  - Update conductance and current

$$g = g_{max}\frac{G^n}{G^n+K_D}$$ (Steady state assuming fast binding)

$$I = g(V - E_{rev})$$

- Procedures and functions:

Name & Arguments	Description	Called by
release()	1. If it is at least **Deadtime** after the last release event ended, and if the presynaptic variable is greater than **Prethresh**, a new release event is started. 2. The transmitter concentration **C** is **Cmax** during a release and **0** otherwise.	DERIVATIVE

- **kleak.mod**
  - Leak potassium current
  - Implemented as a Point Process: **kleak()**
  - Usage:

```
objref kl
kl = new kleak()
access <compartment_name>
kl.loc(0.5)
kl.gmax = ...
```

  - Parameters:

Name	Description	Default value	Type
gmax	Maximum conductance [µS]	**0.004**	range
Erev	Reversal potential [mV]	**-100**	global

  - Equations:

$$I = g_{max}(V - E_{rev})$$

- **cadecay.mod**
  - Fast mechanism for submembranal Ca++ concentration (cai) (Same as Destexhe et al 1998a model but with different default values of **depth** and **cainf**)
  - Suffix: "**cad**" (same as calcium pump)
  - Input/Output: reads **ica** ([mA/cm²]) & **cai**, writes **cai**
  - Parameters:

Name	Description	Default value	Type
depth	Depth of the shell just beneath the membrane [µm]	**1**	range
cainf	Equilibrium concentration of calcium [mM]	**2.4e-4**	range
taur	Time constant of calcium extrusion, must be fast) [ms]	**5**	range
kt, kd	Dummy parameters (parameters specific to the calcium pump)	**0**	range

  - States & initialization:

Name	Description	Initialization
cai	submembranal Ca++ concentration [mM]	cainf

  - Equations:
    - **Inward rectification**:

```
drive_channel = - (10000) * ica / (2 * FARADAY * depth)
                                                    : 10000 µm/cm
```

if (drive_channel <= 0.) { drive_channel = 0. } : cannot pump inward
: (ica should be negative)

- Differential equations:

$$\frac{d[Ca]_i}{dt} = -\frac{I_{Ca}}{2Fd} + \frac{[Ca]_\infty - [Ca]_i}{\tau_r}$$

where F is Faraday's constant, d is the depth of the shell just beneath the membrane
- cai' = drive_channel + (cainf - cai)/taur

- **HH2.mod**
  - Hippocampal Hodgkin-Huxley channels
  - **Q10** was assumed to be **3** for both currents
  - Suffix: "**hh2**"
  - Input/Output: reads **ena** ([mV]) & **ek** ([mV]), writes **ina** [mA/cm²] & **ik** [mA/cm²]
  - Parameters:

Name	Description	Default value	Type
gnabar	Maximum sodium conductivity [S/cm²]	.003	range
gkbar	Maximum potassium conductivity [S/cm²]	.005	range
ena	Reversal potential of sodium channel [mV]	50	global
ek	Reversal potential of potassium channel [mV]	-90	global
celsius	Temperature [℃]	36	global
vtraub	Threshold v_T [mV]	-63	range

  - Other variables visible to hoc:

Name	Description	Default value	Type
m_inf	Asymptotic sodium activation gating variable		range
h_inf	Asymptotic sodium inactivation gating variable		range
n_inf	Asymptotic potassium activation gating variable		range
tau_m	Time constant for sodium activation		range
tau_h	Time constant for sodium activation		range
tau_n	Time constant for sodium activation		range
m_exp	$1 - e^{-(t_{i+1} - t_i)/\tau_m}$		range

h_exp	$1-e^{-(t_{i+1}-t_i)/\tau_h}$		range
n_exp	$1-e^{-(t_{i+1}-t_i)/\tau_n}$		range

- States:

Name	Description	Initialization
m	gating variable for activation of sodium current	0
h	gating variable for inactivation of sodium current	0
n	gating variable for activation of potassium current	0

- Equations:
  - First, update gating variables:

$$T_{adj} = Q_{10}^{(T-36)/10}, Q_{10} = 3$$

$$m_{i+1} = m_i + (1 - e^{-(t_{i+1}-t_i)/\tau_m})(m_\infty - m_i)$$

$$a_m = 0.32\left(\frac{13-(V-V_T)}{e^{(13-(V-V_T))/4}-1}\right)$$

$$b_m = 0.28\left(\frac{(V-V_T)-40}{e^{((V-V_T)-40)/5}-1}\right)$$

$$\tau_m = \frac{1}{T_{adj}(a_m+b_m)}$$

$$m_\infty = \frac{a_m}{a_m+b_m}$$

$$h_{i+1} = h_i + (1 - e^{-(t_{i+1}-t_i)/\tau_h})(h_\infty - h_i)$$

$$a_h = 0.128\left(\frac{17-(V-V_T)}{18}\right)$$

$$b_h = \frac{4}{1+e^{(40-(V-V_T))/5}}$$

$$\tau_h = \frac{1}{T_{adj}(a_h+b_h)}$$

$$h_\infty = \frac{a_h}{a_h+b_h}$$

$$n_{i+1} = n_i + (1 - e^{-(t_{i+1}-t_i)/\tau_n})(n_\infty - n_i)$$

$$a_n = 0.032\left(\frac{15-(V-V_T)}{e^{(15-(V-V_T))/5}-1}\right)$$

$$b_n = 0.5e^{(10-(V-V_T))/40}$$

$$\tau_n = \frac{1}{T_{adj}(a_n+b_n)}$$

$$n_\infty = \frac{a_n}{a_n+b_n}$$

- Next, update currents:

$$I_{Na} = \overline{g}_{Na} m^3 h(V - E_{Na})$$
$$I_K = \overline{g}_K n^4 (V - E_K)$$

- Equations modified by Traub, for Hippocampal Pyramidal cells, in: Traub & Miles, Neuronal Networks of the Hippocampus, Cambridge, 1991

- Procedures and functions:

Name & Arguments	Description	Called by
states()	Updates state variables **m, h, n** based on current voltage	
evaluate_fct(v(mV))	Updates **m_inf, h_inf, n_inf, tau_m, tau_h, tau_n, m_exp, h_exp, n_exp** based on current voltage	states()
vtrap(x,y)	Apply the approximation $\frac{x}{e^{x/y}-1} = y\frac{x/y}{e^{x/y}-1} \approx y(1-\frac{1}{2}(\frac{x}{y}))$ for |x/y| <10^-6	evaluate_fct()
Exp(x)	The exponential function when x ≥ -100, equals zero otherwise	vtrap(x,y) evaluate_fct()

- **Ih.mod**
  - **Anomalous Rectifier I_h** cation (Na/K) channel in **thalamocortical neurons**
  - Model of Destexhe et al., Biophys J. 65: 1538-1552, 1993, based on the voltage-clamp data on the calcium dependence of If in heart cells (Harigawa & Irisawa, J. Physiol. 409: 121, 1989); The voltage-dependence is derived from Huguenard & McCormick, J Neurophysiol. 68: 1373-1383, 1992, based on voltage-clamp data of McCormick & Pape, J. Physiol. 431: 291, 1990.
  - Modified model of the binding of **calcium** through a **calcium-binding (CB) protein**, which in turn acts on Ih channels:
    - Normal voltage-dependent opening of Ih channels
      c1 (closed) <-> **o1** (open) rate constants **α(V)**, **β(V)**
    - Ca++ binding on CB protein (nca=4 binding sites)
      p0 (inactive) + **nca Ca** <-> **p1** (active) rate constants **k1**, **k2**
    - Binding of active CB protein on the open form (nexp binding sites)
      o1 (open) + **nexp p1** <-> **o2** (open) rate constants **k3**, **k4**
  - Remarks:
    - “This simple model for the binding of Ca++ on the open channel suffices to account for the shift in the voltage-dependence of Ih activation with calcium (see details in Destexhe et al, 1993).”
    - “It may be that calcium just binds to the Ih channel, preventing the

conformational change between open and closed; in this case one should take into account binding on the closed state, which is neglected here."
    - "This file also contains a procedure ("**activation**") to estimate the steady-state activation of the current; callable from outside"
  - Suffix: "**iar**"
  - Input/Output: reads **eh** [mV] & **cai** [mM], writes **ih** [mA/cm²], valence = 1
  - Parameters:

Name	Description	Default value	Type
eh	Reversal potential for H current [mV]	-40	global
celsius	Temperature [°C]	36	global
ghbar	Maximum H current conductance [S/cm²]	2e-5	range
shift	Shift towards **de**polarization of the activation curve of I_h [mV]	0	range
cac	Half-activation (p0 = p1) of calcium dependence of CB protein (=(k2/k1)^(1/nca)) [mM]	0.002	global
k2	Inverse of the real time constant of the binding of Ca++ to the CB protein [ms^{-1}]	0.0004	global
Pc	Half-activation (o1 = o2) of CB protein dependence of I_h channels (=(k4/k3)^(1/nexp)) [1]	0.01	global
k4	Inverse of the real time constant of the binding of the CB protein to Ih channels (it basically governs the interspindle period) [ms^{-1}]	0.001	global
nca	Number of binding sites of Ca++ [1]	4	global
nexp	Number of binding sites on Ih channels [1]	1	global
ginc	Augmentation of conductance associated with the Ca++ bound state [1]	2	global
taum	Minimum value of tau_s [ms]	20	global

  - Other variables visible to hoc:

Name	Description	Default value	Type
m	Percent of H channels opened [1]		range
h_inf	Asymptotic H channel opening gating variable [1]		range

tau_s	Time constant for channel opening [ms]		range

- States:

Name	Description	Initialization
c1	closed state of channel	1
o1	open state of channel without CB-bound	0
o2	CB-bound open state of channel	0
p0	resting CB	1
p1	Ca++-bound CB	0

- Equations:
  - First, update states according to the kinetic scheme:

~ c1 <-> o1	(alpha, beta)
~ p0 <-> p1	(k1ca, k2)
~ o1 <-> o2	(k3p, k4)

CONSERVE p0 + p1 = 1
CONSERVE c1 + o1 + o2 = 1

where the rate constants are computed by

$$\alpha = \frac{h_\infty}{\tau_s},\ \beta = \frac{1-h_\infty}{\tau_s}$$

$$h_\infty = \frac{1}{1+e^{(V+75-shift)/5.5}}$$

$$\tau_s = \frac{1}{T_{adj}}(\tau_m + \frac{1000}{e^{(V+71.5-shift)/14.2}+e^{-(V+89-shift)/11.6}})$$

(**shift** = **2** in all simulations of this paper)

$$T_{adj} = Q_{10}^{(T-36)/10}$$, where Q10 = **3.0**

$$k1ca = k2(\frac{[Ca]_i}{[Ca]_c})^{nca},\ k3p = k4(\frac{p1}{Pc})^{nexp}$$

  - Next, update percent of H channels opened and compute I_h:

$$m = o1 + ginc * o2$$

$$I_{Ca} = \overline{g}_h m(V - E_h)$$

- Procedures and functions:

Name & Arguments	Description	Called by
evaluate_fct(v (mV), cai (mM))	Update **h_inf**, **tau_s**, **alpha**, **beta**, **k1ca**, **k3p** based on current voltage and calcium concentration	INITIAL KINETIC activation()

activation(v (mV), cai (mM))	Evaluates the activation curve of I_h by first calling evaluate_fct(), then (easily derived from steady state values): $cc = \frac{1}{1+(\frac{[Ca]c}{[Ca]_i})^{nca}}$ (cc is the steady state value of p1) $m = \frac{1+ginc(\frac{cc}{Pc})^{nexp}}{1+\beta/\alpha+(\frac{cc}{Pc})^{nexp}}$	

- **IT.mod**
  - T-type calcium current responsible for low-threshold spikes (LTS) in **thalamocortical neurons**
  - Model based on the data of Huguenard & McCormick, J Neurophysiol 68: 1373-1383, 1992 & Huguenard & Prince, J Neurosci. 12: 3804-3817, 1992
  - The kinetics is described by **Nernst equations**, using a m^2h format
  - The activation function was empirically corrected to account for the contamination of inactivation:
    - An overall **hyperpolarizing shift** of **2 mV** was applied to compensate for **screening charge**
  - Inactivation curve is fit to data using a bi-exponential function.
  - Temperature dependence assumes a Q10 value of **3** for both m and h.
  - Suffix: “**it**”
  - Input/Output: reads **cai** [mM] & **cao** [mM], writes **ica** [mA/cm²]
  - Parameters:

Name	Description	Default value	Type
v	Membrane potential		global
celsius	Temperature [℃]	36	global
gcabar	Maximum calcium conductance [S/cm²]	**0.002**	range
shift	Shift towards hyperpolarization of *both activation & inactivation curves* [mV]	**2**	range
cai	Calcium concentration inside the cell [mM]	2.4e-4	global
cao	Calcium concentration outside the cell [mM]	2	global
q10	Q_10 for inactivation curve [1]	**3**	**global**

  - Other variables visible to hoc:

Name	Description	Default value	Type

m_inf	Asymptotic calcium activation gating variable [1]		range
h_inf	Asymptotic calcium inactivation gating variable [1]		range
tau_m	Time constant for calcium activation [ms]		range
tau_h	Time constant for calcium activation [ms]		range

- States:

Name	Description	Initialization
h	gating variable for inactivation of calcium current	0

- Equations:
  - First, update gating variables:

$$\frac{dh}{dt} = \frac{h_\infty - h}{\tau_h}$$

$$m_\infty = \frac{1}{1+e^{-(V+57+shift)/6.2}}$$

(Here, **V_1/2** is assumed to be **-57 mV**, but can be modified by **shift**, which is **2** in all simulations of this paper)

$$h_\infty = \frac{1}{1+e^{(V+81+shift)/4}}$$

(Here, **V_1/2** is assumed to be **-81 mV**, but can be modified by **shift**, which is **2** in all simulations of this paper)

$$\tau_h = \frac{1}{\Phi_h}(30.8 + \frac{211.4+e^{(V+113.2+shift)/5}}{1+e^{(V+84+shift)/3.2}})$$

(**shift** = **2** in all simulations of this paper)

$$\Phi_h = {Q_{10}}^{(T-24)/10}$$

In all simulations of this paper, Q_10 = **3**.

  - Next, update currents:

$$I_{Ca} = \overline{g}_{Ca} {m_\infty}^2 h(V - E_{Ca})$$

$$E_{Ca} = (10^3)\frac{RT}{ZF} log\frac{[Ca]_o}{[Ca]_i}$$

where Z = 2, T is in [K], V is in [mV]. This is based on the Nernst equation.

- Procedures and functions:

Name & Arguments	Description	Called by
evaluate_fct(v(mV))	Update **m_inf, h_inf, tau_h** based on current voltage	DERIVATIVE

- **IT2.mod**
  - T-type calcium current responsible for low-threshold spikes (LTS) in **reticular thalamic neurons**
  - Model based on the data of Huguenard & McCormick, J Neurophysiol 68: 1373-1383, 1992 & Huguenard & Prince, J Neurosci. 12: 3804-3817, 1992
  - The kinetics is described by **Nernst equations**, using a m^2h format
  - Temperature dependence assumes a Q10 value of **2.5** for both m and h.
  - Suffix: “**it2**”
  - Input/Output: reads **cai** [mM] & **cao** [mM], writes **ica** [mA/cm²]
  - Parameters:

Name	Description	Default value	Type
v	Membrane potential		global
celsius	Temperature [°C]	36	global
gcabar	Maximum calcium conductance [S/cm²]	**0.003**	range
shift	Shift towards hyperpolarization of *both activation & inactivation curves* [mV]	**0***	range
cai	Calcium concentration inside the cell [mM]	2.4e-4	global
cao	Calcium concentration outside the cell [mM]	2	global
qm	Q_10 for activation curve [1]	**2.5**	**range**
qh	Q_10 for inactivation curve [1]	**2.5**	**range**

***shift** = **2** in all simulations of this paper, set in **RE.tem**

  - Other variables visible to hoc:

Name	Description	Default value	Type
m_inf	Asymptotic calcium activation gating variable [1]		range
h_inf	Asymptotic calcium inactivation gating variable [1]		range
tau_m	Time constant for calcium activation [ms]		range
tau_h	Time constant for calcium activation [ms]		range

  - States:

Name	Description	Initialization
m	gating variable for activation of calcium current	m_inf

h	gating variable for inactivation of calcium current	h_inf

- Equations:
  - First, update gating variables:

$$\frac{dm}{dt} = \frac{m_\infty - m}{\tau_m}$$

$$\frac{dh}{dt} = \frac{h_\infty - h}{\tau_h}$$

$$m_\infty = \frac{1}{1+e^{-(V+50+shift)/7.4}}$$

(Here, **V_1/2** is assumed to be **-50 mV**, but can be modified by **shift**, which is **2** in all simulations of this paper)

$$h_\infty = \frac{1}{1+e^{(V+78+shift)/5.0}}$$

(Here, **V_1/2** is assumed to be **-78 mV**, but can be modified by **shift**, which is **2** in all simulations of this paper)

$$\tau_m = \frac{1}{\Phi_m}\left(3 + \frac{1}{e^{(V+25+shift)/10} + e^{-(V+100+shift)/15}}\right)$$

$$\tau_h = \frac{1}{\Phi_h}\left(85 + \frac{1}{e^{(V+46+shift)/4} + e^{-(V+405+shift)/50}}\right)$$

(**shift** = **2** in all simulations of this paper)

$$\Phi_m = Q_{10,m}{}^{(T-24)/10}$$

$$\Phi_h = Q_{10,h}{}^{(T-24)/10}$$

In all simulations of this paper, Q_10_m = Q_10_h = **2.5**.

  - Next, update currents:

$$I_{Ca} = \overline{g}_{Ca} m^2 h (V - E_{Ca})$$

$$E_{Ca} = (10^3)\frac{RT}{ZF} log\frac{[Ca]_o}{[Ca]_i}$$

where Z = 2, T is in [K], V is in [mV]. This is based on the Nernst equation.

- Procedures and functions:

Name & Arguments	Description	Called by
evaluate_fct(v(mV))	Update **m_inf, h_inf, tau_h** based on current voltage	DERIVATIVE

Plan for next week

1. Spike-wave detection codes
2. Go through the NEURON Hands-On Course
3. Read and take notes from Destexhe et al 1996
4. Reproduce figures in Destexhe et al 1996

Plan for the future

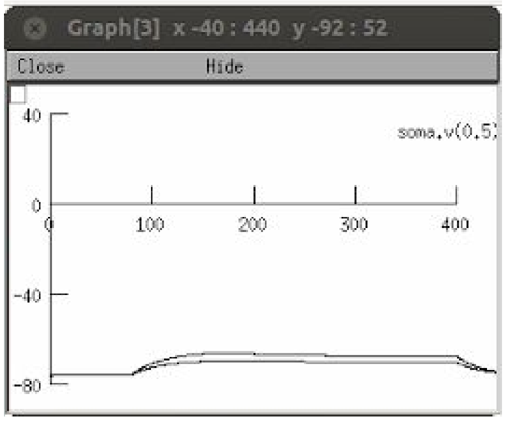
1. Reproduce figures in Destexhe et al 1998b
2. Reproduce figures in Destexhe 1998
3. Reproduce figures in Destexhe et al 2001
4. Reproduce figures in Traub et al 2005
5. Compile relevant papers that have cited Destexhe et al 1996 and/or have used the Destexhe et al 1996 model
6. Compile relevant papers that have cited Destexhe et al 1998b and/or Destexhe 1998 and/or Destexhe et al 2001 and/or have used the Destexhe et al 1998, 2001 model
7. Compile relevant papers that have cited Traub et al 2005 and/or have used the Traub et al 2005 model
8. Refine reproduction of figures in Destexhe et al 1998a
9. Compile relevant papers about absence epilepsy
10. Read and write down notes for Chen et al 2014 and Chen et al 2015
11. Read and write down notes for Zhao et al 2015
12. Examine the codes for the Chen et al 2014, 2015 model
13. Examine the codes for the Zhao et al 2015 model
14. Examine the codes for the Traub et al 2005 model
15. “Reproduce CSD graph” exercise
16. Examine Christine’s & Mark’s codes
17. Finish NEURON Book Appendix A1
18. Figure out how to export NEURON to Matlab
19. Complete the NEURON Tutorial
20. Understand NEURON Ch 7 & Ch 8
21. Resolve all NEURON Book questions
22. Read Abbott et al 2016 (“Building functional networks of spiking model neurons”)
23. Read Markram et al 2015 (“Reconstruction and Simulation of Neocortical Microcircuitry”)
24. Read Kragel & LaBar 2016 (“Decoding the Nature of Emotion in the Brain”)
25. Read Izhikevich: Dynamical Systems in Neuroscience
26. Read Dayan & Abbott: Theoretical Neuroscience
27. Derivation & shape of the **Goldman–Hodgkin–Katz flux equation**

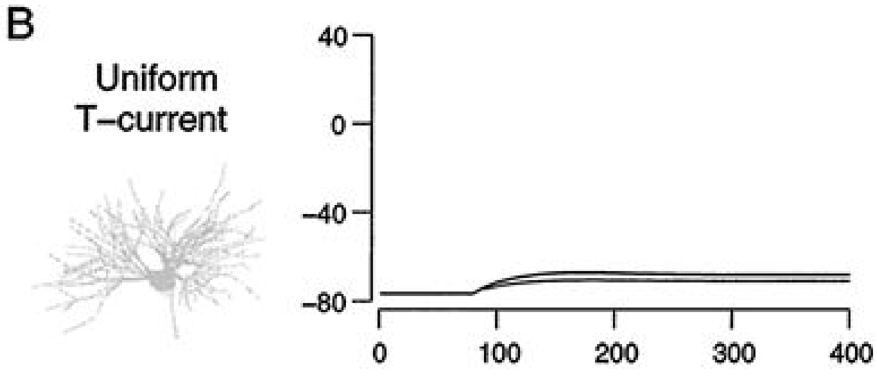
6/6/2016~6/7/2016

Reproduce Figures in Destexhe et al 1998a (part 3)

- **Figure 9**
    - **Current clamp** of the **detailed cell model**:
      Initialize v_init @ **-76.5/-74 mV**, after a delay of **80 ms**, inject current (**50 pA** and **75 pA**) for a duration of **320 ms**.
    - **9B**: Uniform T-channel density
      9C: T-channel density higher in distal dendrites
    - Code as in **tc200_cc.oc**, just toggle between:

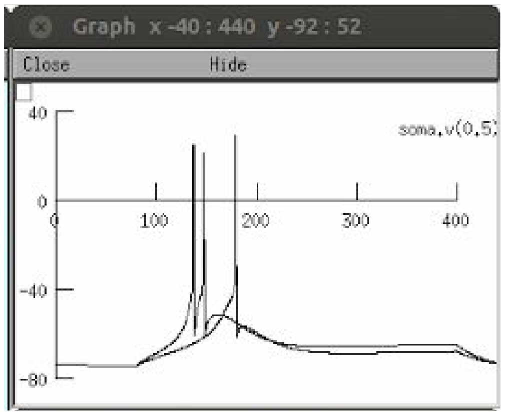
localize(1.7e-5,corrD*1.7e-5,corrD*1.7e-5):

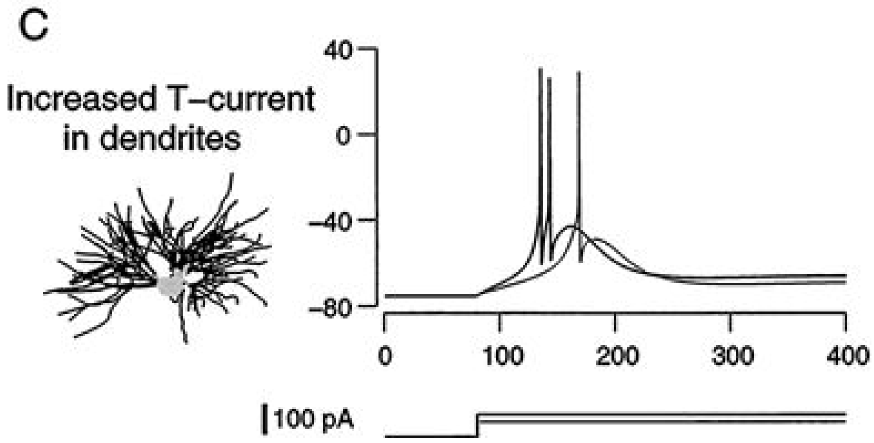




and

localize(1.7e-5,corrD*8.5e-5,corrD*8.5e-5)



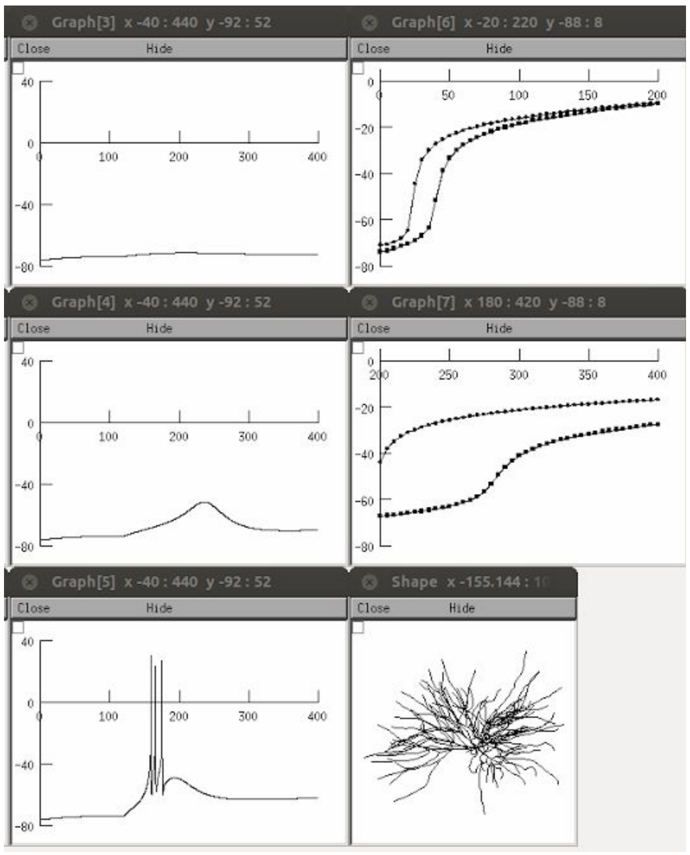


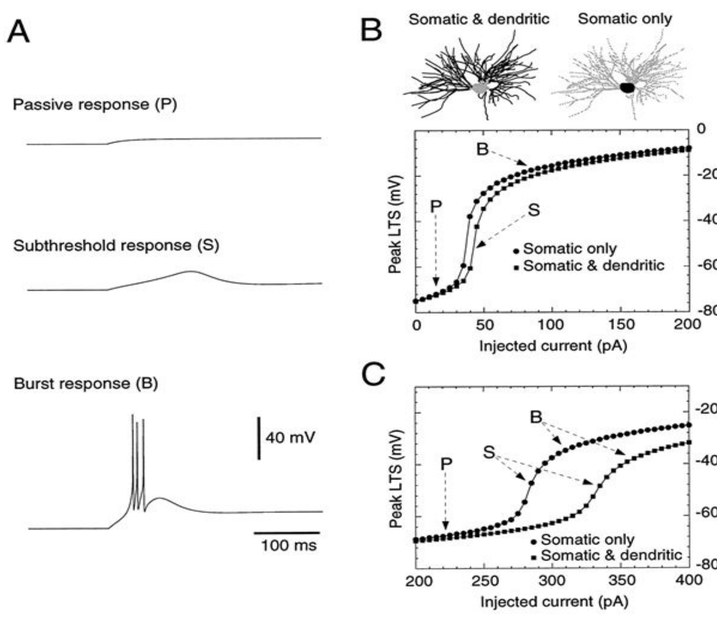
- **Figure 10**
    - Properties of **dendritic T currents** in the **detailed cell model**
    - **10A**: Example current clamp responses with HH intact:
      Initialize v_init @ E_pas (**76.5 mV**), after a delay of **120 ms**, inject current (**15 pA**, **40 pA**, **100 pA**) for duration of **280 ms**.
      10B: Peak low threshold spike amplitudes with

- Modified code of **tc200_cc.oc**:

```
proc addgraph() { local ii       // define subroutine to add a new graph for a variable that is
simulated
                                 // addgraph("variable", t_min, t_max, var_min, var_max,
window_left, window_top, window_width, window_height, KeepLines)
        ngraph = ngraph+1
        ii = ngraph-1
        g[ii] = new Graph(0)     // is not mapped; will be sized and placed with the .view() function
        g[ii].view($2, $4, $3-$2, $5-$4, $6, $7, $8, $9)
        g[ii].xaxis()
        g[ii].yaxis()
        g[ii].addvar($s1,1,0)
        g[ii].family($10)                // turn Keep Lines on if $10 == 1
        g[ii].save_name("graphList[0].")
        graphList[0].append(g[ii])
}

proc addgraph2() { local ii                     // Create graph for plotting in general
                                                // addgraph2(x_min, x_max, y_min, y_max,
window_left, window_top, window_width, window_height)
        ngraph = ngraph+1
        ii = ngraph - 1
        g[ii] = new Graph(0)
        g[ii].view($1, $3, $2-$1, $4-$3, $5, $6, $7, $8)
        g[ii].xaxis()
        g[ii].yaxis()
        g[ii].family($9)         // turn Keep Lines on if $10 == 1
        g[ii].save_name("graphList[0].")
        graphList[0].append(g[ii])
        last = ii
}

proc plotxy() {                                 // Plot vector $o1 against vector $o2 in graph $o3
as a line with color $4
                                                // Color: 0 white 1 black 2 red 3 blue 4 green 5
orange 6 brown 7 violet 8 yellow 9 gray
        $o2.line($o3, $o1, $4, 1)
        print $o2, " vs. ", $o1, " plotted!"    // for debugging
}

proc markxy() {                                         // Plot vector $o1 against vector $o2 in
graph $o3 with mark $s4
                                                // "S" is square, "T" is triangle, "O" is circle
        $o2.mark($o3, $o1, $s4, 4)
        print $o2, " vs. ", $o1, " marked!"        // for debugging
}

El.stim.del = 120
El.stim.dur = 280

npoints = 2000                                 // 400 ms is sufficient
v_init = E_pas                                 // Start from steady state potential

objref time, somav
proc simulate1() {                     // Representative responses to current injection of $1 pA,
plot in graph $o2
        time = new Vector()            // A vector of time points used in the simulation
        for ii = 0, tstop/dt time.append(ii*dt)
        somav = new Vector()           // For recordings of soma.v(0.5)
        somav.record(&soma.v(0.5)) // Records soma.v(0.5)
        El.stim.amp = $1/1000                  // Current injection amplitude in nA
        run()
        if (stoppedrun()) {
                break
        }
        plotxy(time, somav, $o2, 1)
}

objref ci, peakLTS
proc simulate2() { local n, x1, x2, dx         // Current injection amplitudes run from $1 to $2
with interval $3, plot peakLTS vs somav on graph $o4 with mark $s5
        // Set current ranges
        x1 = $1
        x2 = $2
        dx = $3
        n = (x2-x1)/dx + 1                     // Number of current injection amplitudes
        ci = new Vector(n,0)                   // For storing each current injection amplitude
        peakLTS = new Vector(n,0)              // For storing peak low threshold spike amplitude for
each current injection amplitude
        for i=0, n-1 ci.x[i] = x1 + dx*i
        for (x = x1; x <= x2; x = x + dx) {
                somav = new Vector()           // For recordings of soma.v(0.5)
                somav.record(&soma.v(0.5)) // Records soma.v(0.5)
                El.stim.amp = x/1000           // Current injection amplitude in nA
                run()
                if (stoppedrun()) {
                    break
                }
                left = gfloor(tstart/dt)
                right = gfloor(tstop/dt)
                peakLTS.x[(x-x1)/dx] = somav.max(left,right)
        }
        plotxy(ci, peakLTS, $o4, 1)
        markxy(ci, peakLTS, $o4, $s5)
}

// Fig 10B current clamp current range (pA)
w1 = 0
w2 = 200
dw = 5

// Fig 10C current clamp current range (pA)
y1 = 200
y2 = 400
dy = 5

// Prepare graphs
addgraph("soma.v(0.5)", tstart, tstop, -80, 40, 761, 102, 300.48, 200.32, 0)
addgraph("dend10[6].v(0.5)", tstart, tstop, -80, 40, 761, 365, 300.48, 200.32, 0)
addgraph("dend10[26].v(0.5)", tstart, tstop, -80, 40, 761, 628, 300.48, 200.32, 0)
addgraph2(tstart, tstop, -80, 40, 1080, 102, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 1080, 365, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 1080, 628, 300.48, 200.32, 1)
addgraph2(0, 200, -80, 0, 1399, 102, 300.48, 200.32, 1)
addgraph2(200, 400, -80, 0, 1399, 365, 300.48, 200.32, 1)
addshape()
//load_file("rig.ses")                    // Causes problems!

// Representative responses to current injection
simulate1(15 ,g[3])                       // Passive response (P)
simulate1(40 ,g[4])                       // Subthreshold response (S)
simulate1(100 ,g[5])                      // Burst response (B)

// Set Na & K currents to 0 so that only T-currents are present
soma {
        gnabar_hh2 = 0
        gkbar_hh2 = 0
}
```

```
// Somatic only
localize(56.53e-5,corrD*0,corrD*0)
simulate2(w1, w2, dw, g[6], "O")

// Increase dendritic shunt conductance to mimic the effects of mixed excitatory and inhibitory
inputs
forall { g_pas = 0.15e-3 }
soma { g_pas = G_pas * corrD }
simulate2(y1, y2, dy, g[7], "O")
forall { g_pas = G_pas * corrD }

// Somatic & dendritic
localize(1.7e-5,corrD*8.5e-5,corrD*8.5e-5)
simulate2(w1, w2, dw, g[6], "S")

// Increase dendritic shunt conductance to mimic the effects of mixed excitatory and inhibitory
inputs
forall { g_pas = 0.15e-3 }
soma { g_pas = G_pas * corrD }
simulate2(y1, y2, dy, g[7], "S")
forall { g_pas = G_pas * corrD }
```


Graph[3] x -40 : 440 y -92 : 52
Graph[6] x -20 : 220 y -88 : 8
Graph[4] x -40 : 440 y -92 : 52
Graph[7] x 180 : 420 y -88 : 8
Graph[5] x -40 : 440 y -92 : 52
Shape x -155.144 : 1
Close
Hide



- **Figure 11 & 12**
    - **Voltage clamp** & **current clamp** of the **3-compartment model** and the **1-compartment model**
    - Modified code of **tc3_cc.oc** as in **tc200_cc.oc**, except:

```
proc changeview() {                         // changeview(graph, t_min, t_max, var_min,
var_max, window_left, window_top, window_width, window_height, KeepLines)
        $o1.view($2, $4, $3-$2, $5-$4, $6, $7, $8, $9)
        $o1.family($10)                             // turn Keep Lines on if $10 == 1
}

El.stim.del = 80
El.stim.dur = 320

v_init = -74                    // approximate resting Vm

objref time, somav
proc simulate1() {                  // Representative responses to current injection of $1 pA,
plot in graph $o2 with color $3 and brush $4
        time = new Vector()         // A vector of time points used in the simulation
        for ii = 0, tstop/dt time.append(ii*dt)
        somav = new Vector()        // For recordings of soma.v(0.5)
        somav.record(&soma.v(0.5))  // Records soma.v(0.5)
        El.stim.amp = $1/1000               // Current injection amplitude in nA
        run()
        if (stoppedrun()) {
                break
        }
        plotxy(time, somav, $o2, $3, $4)
}

objref ci, peakLTS
proc simulate2() { local n, x1, x2, dx      // Current injection amplitudes run from $1 to $2
with interval $3, plot peakLTS vs somav on graph $o4 with mark $s5
        // Set current ranges
        x1 = $1
        x2 = $2
        dx = $3
        n = (x2-x1)/dx + 1                  // Number of current injection amplitudes
        ci = new Vector(n,0)                // For storing each current injection amplitude
        peakLTS = new Vector(n,0)           // For storing peak low threshold spike amplitude for
each current injection amplitude
        for i=0, n-1 ci.x[i] = x1 + dx*i
        for (x = x1; x <= x2; x = x + dx) {
                somav = new Vector()            // For recordings of soma.v(0.5)
                somav.record(&soma.v(0.5))  // Records soma.v(0.5)
                EI.stim.amp = x/1000            // Current injection amplitude in nA
                print "EI.stim.del = ", EI.stim.del, "EI.stim.dur = ", EI.stim.dur, "EI.stim.amp = ",
EI.stim.amp                     // for debugging
                run()
                if (stoppedrun()) {
                        break
                }
                print "The size of somav is ", somav.size()
                                // for debugging
                printf("The peak value of somav will be the maximum value between the indices
%d and %d\n", tstart/dt, tstop/dt)      // for debugging
                left = gfloor(tstart/dt)
                right = gfloor(tstop/dt)
                peakLTS.x[(x-x1)/dx] = somav.max(left,right)
                printf("peakLTS.x[%d] = %g\n", (x-x1)/dx, peakLTS.x[(x-x1)/dx])
                                        // for debugging
        }
        plotxy(ci, peakLTS, $o4, 1, 1)
        markxy(ci, peakLTS, $o4, $s5)
}

objref testvoltagelevel
proc simulate3() {                              // Test voltage level runs from $1 to $2 with interval
$3, plot peak current vs holding voltage level on graph $o4 with mark $s5
        // Erase first three graphs
        g[0].erase()
        g[1].erase()
        g[2].erase()
        // Set VClamp Family specifications (used in varyamp())
        EI.x1 = $1
        EI.x2 = $2
        EI.dx = $3
        n = (EI.x2-EI.x1)/EI.dx + 1             // number of test voltage levels
        testvoltagelevel = new Vector(n,0)      // For storing each test voltage level
        for i=0, n-1 testvoltagelevel.x[i] = EI.x1 + EI.dx*i
        EI.varyamp(1)                           // Varies the test voltage level and calls run()
repeatedly
        plotxy(testvoltagelevel, EI.vci_peak, $o4, 1, 1)
        markxy(testvoltagelevel, EI.vci_peak, $o4, $s5)
}
```

```
// Fig 12B current clamp current range (pA)
w1 = 0
w2 = 200
//dw = 100                          // for debugging
dw = 5

// Fig 12C current clamp current range (pA)
y1 = 200
y2 = 400
//dy = 100                          // for debugging
dy = 5

// Prepare graphs
addgraph("soma.v(0.5)", tstart, tstop, -80, 40, 442, 102, 300.48, 200.32, 0)
addgraph("dend1[1].v(0.5)", tstart, tstop, -80, 40, 442, 365, 300.48, 200.32, 0)
addgraph("El.vc.i", tstart, tstop, -80, 40, 442, 628, 300.48, 200.32, 0)
addgraph2(tstart, tstop, -0.6, 0.4, 761, 102, 300.48, 200.32, 1)
addgraph2(-82.5, -17.5, -5, 0.5, 761, 365, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 761, 628, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 1080, 102, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 1080, 365, 300.48, 200.32, 1)
addgraph2(0, 200, -80, 0, 1399, 102, 300.48, 200.32, 1)
addgraph2(200, 400, -80, 0, 1399, 365, 300.48, 200.32, 1)
addshape()
//load_file("rig.ses")              // Causes problems!

// Fig 12A1: Representative responses to current injection, uniform T-channel density
localize(1.7e-5,corrD*1.7e-5)
simulate1(50 ,g[6], 1, 1)
simulate1(75 ,g[6], 1, 2)

// Fig 11A4 & Fig 12A2: Representative responses to current injection, high dendritic T-channel
density
localize(1.7e-5,corrD*9.5e-5)
simulate1(50 ,g[7], 1, 1)
simulate1(75 ,g[7], 1, 2)
simulate1(50 ,g[5], 1, 1)
simulate1(75 ,g[5], 1, 2)

// Set Na & K currents to 0 so that only T-currents are present
soma {
        gnabar_hh2 = 0
        gkbar_hh2 = 0
}

// Fig 12B, Somatic only: peak low threshold spike amplitude for each current injection amplitude
localize(56.36e-5,corrD*0,corrD*0)
simulate2(w1, w2, dw, g[8], "O")

// Fig 12C, Somatic only: Increase dendritic shunt conductance to mimic the effects of mixed
excitatory and inhibitory inputs
forall { g_pas = 0.15e-3 * corrD }
soma { g_pas = G_pas }
simulate2(y1, y2, dy, g[9], "O")
forall { g_pas = G_pas * corrD }
soma { g_pas = G_pas }

// Fig 12B, Somatic & dendritic: peak low threshold spike amplitude for each current injection
amplitude
localize(1.7e-5,corrD*9.5e-5)
simulate2(w1, w2, dw, g[8], "S")

// Fig 12C, Somatic & dendritic: Increase dendritic shunt conductance to mimic the effects of
mixed excitatory and inhibitory inputs
forall { g_pas = 0.15e-3 * corrD }
soma { g_pas = G_pas }
simulate2(y1, y2, dy, g[9], "S")
forall { g_pas = G_pas * corrD }
soma { g_pas = G_pas }

//
// VOLTAGE-CLAMP MODE
//

trans = 1000

print " "
print ">> Transient time of ",trans," ms"
print " "

Dt = 0.2
npoints = 500                   // 100 ms is usually enough to find peak current

dt = 0.1                        // must be submultiple of Dt
tstart = trans
```

```
tstop = trans + npoints * Dt
runStopAt = tstop
steps_per_ms = 1/Dt

celsius = 24                    // temperature of John's experiments in VC
v_init = -70

soma El.vc.loc(0.5)             // put electrode in voltage-clamp mode

// Modify graph properties
changeview(g[0], tstart, tstop, -120, 20, 442, 102, 300.48, 200.32, 1)
changeview(g[1], tstart, tstop, -120, 20, 442, 365, 300.48, 200.32, 1)
changeview(g[2], tstart, tstop, -5, 0.5, 442, 628, 300.48, 200.32, 1)
changeview(g[3], tstart, tstop, -0.6, 0.4, 761, 102, 300.48, 200.32, 1)

// Fig 11A2
El.vc.dur[0] = trans
El.vc.dur[1] = 50
El.vc.dur[2] = 50
El.vc.amp[0] = -80
El.vc.amp[1] = -82.5
El.vc.amp[2] = -80

El.vc.rs = 5                    // default series resistance in tc200_vc.oc

// Simulate
time = new Vector()             // A vector of time points used in the simulation
for ii = 0, tstop/dt time.append(ii*dt)
objref vci
vci = new Vector()              // For recordings of El.vc.i
vci.record(&El.vc.i)            // Records El.vc.i
run()
if (stoppedrun()) {
        break
}
plotxy(time, vci, g[3], 1, 1)

// Fig 11A3
El.vc.dur[0] = trans
El.vc.dur[1] = 100
El.vc.dur[2] = 0
hold = -115
El.vc.amp[0] = hold
El.vc.amp[1] = hold
El.vc.amp[2] = hold
El.map()                        // This makes sure that vamp[3] & vdur[3] are updated to
user-specified values

El.vc.rs = 12                   // series resistance seems to be 12 MOhm in this figure

forall { g_pas = 0 }            // remove passive current everywhere

// Voltage clamp step voltage level range
c1 = -80
c2 = -20
//dc = 60                       // for debugging
dc = 5

// Same T-current density as Figure 9C
localize(1.7e-5,corrD*8.5e-5)
simulate3(c1, c2, dc, g[4], "T")

// Increased T-current density in dendrites
localize(1.7e-5,corrD*9.5e-5)
simulate3(c1, c2, dc, g[4], "O")
```

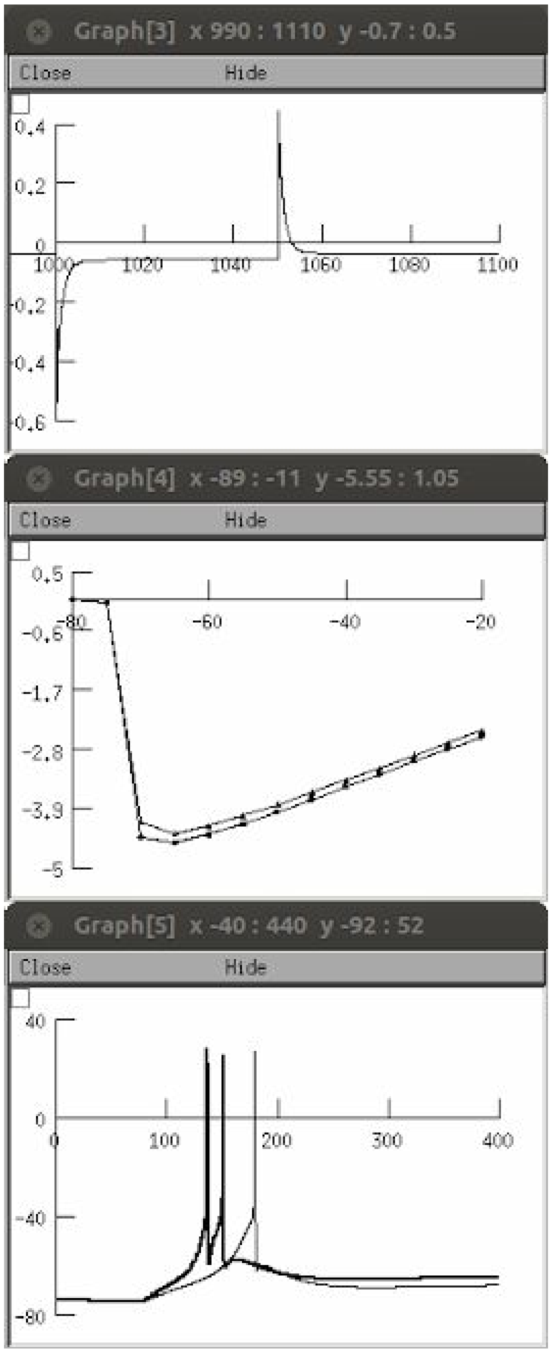
- Modified code of **tc1_cc.oc** as in **tc200_cc.oc**, except:

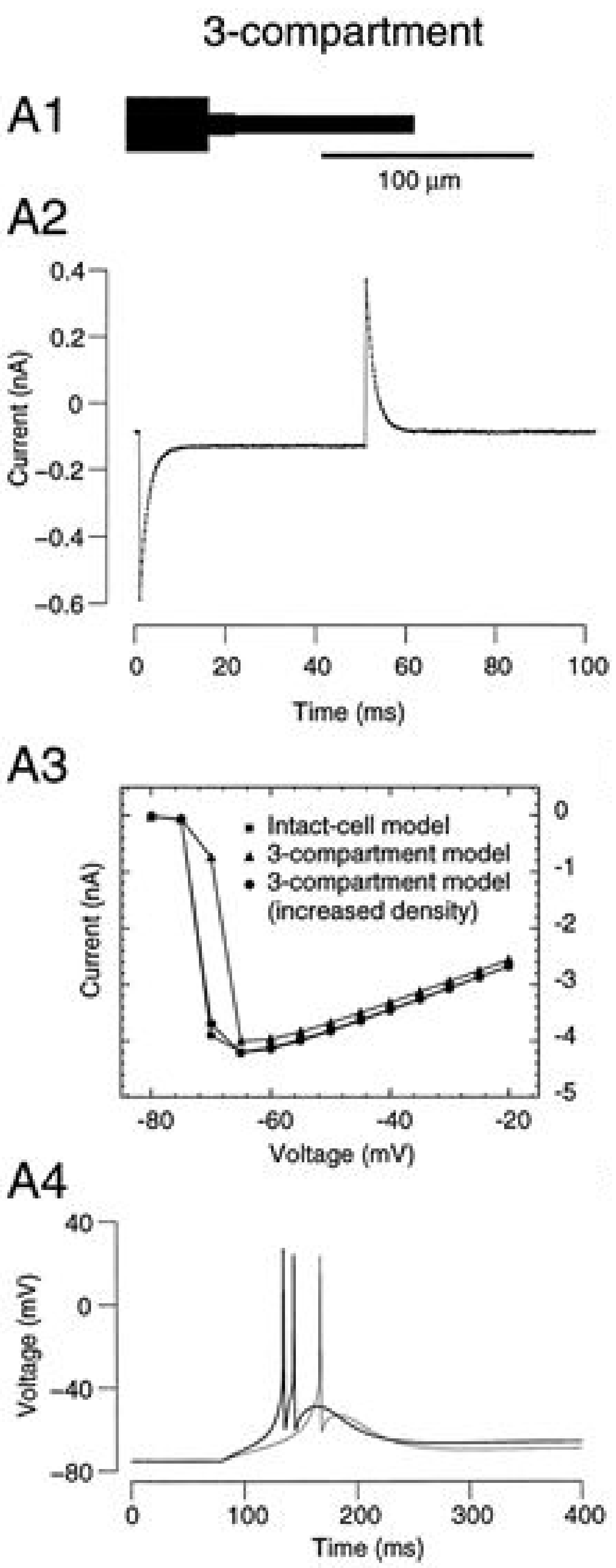
```
objref time, somav
proc simulate1() {                      // Representative responses to current injection of $1 pA,
plot in graph $o2 with color $3 and brush $4
        time = new Vector()             // A vector of time points used in the simulation
        for ii = 0, tstop/dt time.append(ii*dt)
        somav = new Vector()            // For recordings of soma.v(0.5)
        somav.record(&soma.v(0.5))  // Records soma.v(0.5)
        El.stim.amp = $1/1000                   // Current injection amplitude in nA
        run()
        if (stoppedrun()) {
                break
        }
        plotxy(time, somav, $o2, $3, $4)
}

objref ci, peakLTS
proc simulate2() { local n, x1, x2, dx          // Current injection amplitudes run from $1 to $2
with interval $3, plot peakLTS vs somav on graph $o4 with mark $s5
        // Set current ranges
```

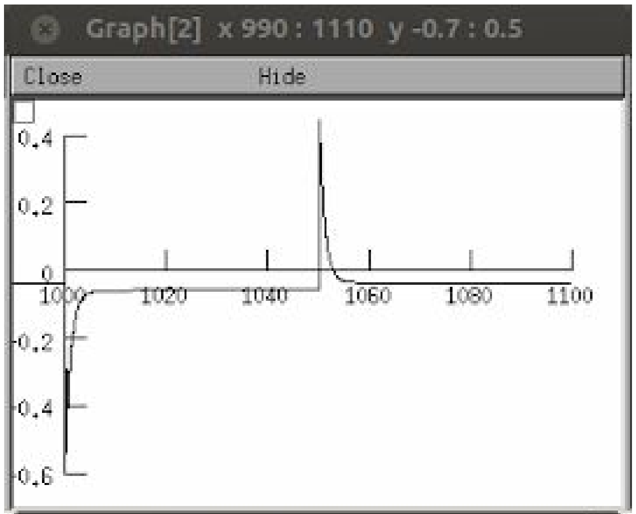
```
	x1 = $1
	x2 = $2
	dx = $3
	n = (x2-x1)/dx + 1			// Number of current injection amplitudes
	ci = new Vector(n,0)			// For storing each current injection amplitude
	peakLTS = new Vector(n,0)		// For storing peak low threshold spike amplitude for
each current injection amplitude
	for i=0, n-1 ci.x[i] = x1 + dx*i
	for (x = x1; x <= x2; x = x + dx) {
		somav = new Vector()		// For recordings of soma.v(0.5)
		somav.record(&soma.v(0.5))	// Records soma.v(0.5)
		El.stim.amp = x/1000		// Current injection amplitude in nA
		print "El.stim.del = ", El.stim.del, "El.stim.dur = ", El.stim.dur, "El.stim.amp = ",
El.stim.amp			// for debugging
		run()
		if (stoppedrun()) {
			break
		}
		print "The size of somav is ", somav.size()
				// for debugging
		printf("The peak value of somav will be the maximum value between the indices
%d and %d\n", tstart/dt, tstop/dt)	// for debugging
		left = gfloor(tstart/dt)
		right = gfloor(tstop/dt)
		peakLTS.x[(x-x1)/dx] = somav.max(left,right)
		printf("peakLTS.x[%d] = %g\n", (x-x1)/dx, peakLTS.x[(x-x1)/dx])
					// for debugging
	}
	plotxy(ci, peakLTS, $o4, 1, 1)
	markxy(ci, peakLTS, $o4, $s5)
}

objref testvoltagelevel
proc simulate3() {				// Test voltage level runs from $1 to $2 with interval
$3, plot peak current vs holding voltage level on graph $o4 with mark $s5
	// Erase first two graphs
	g[0].erase()
	g[1].erase()
	// Set VClamp Family specifications (used in varyamp())
	El.x1 = $1
	El.x2 = $2
	El.dx = $3
	n = (El.x2-El.x1)/El.dx + 1		// number of test voltage levels
	testvoltagelevel = new Vector(n,0)	// For storing each test voltage level
	for i=0, n-1 testvoltagelevel.x[i] = El.x1 + El.dx*i
	El.varyamp(1)				// Varies the test voltage level and calls run()
repeatedly
	plotxy(testvoltagelevel, El.vci_peak, $o4, 1, 1)
	markxy(testvoltagelevel, El.vci_peak, $o4, $s5)
}

// Fig 12B current clamp current range (pA)
w1 = 0
w2 = 200
//dw = 100				// for debugging
dw = 5

// Fig 12C current clamp current range (pA)
y1 = 200
y2 = 400
//dy = 100				// for debugging
dy = 5

// Prepare graphs
addgraph("soma.v(0.5)", tstart, tstop, -80, 40, 442, 102, 300.48, 200.32, 0)
addgraph("El.vc.i", tstart, tstop, -80, 40, 442, 365, 300.48, 200.32, 0)
addgraph2(tstart, tstop, -0.6, 0.4, 761, 102, 300.48, 200.32, 1)
addgraph2(-82.5, -17.5, -5, 0.5, 761, 365, 300.48, 200.32, 1)
addgraph2(tstart, tstop, -80, 40, 761, 628, 300.48, 200.32, 1)
addgraph2(0, 200, -80, 0, 1080, 102, 300.48, 200.32, 1)
addshape()
//load_file("rig.ses")			// Causes problems!

// Fig 11B4: Representative responses to current injection
simulate1(50 ,g[4], 1, 1)
simulate1(75 ,g[4], 1, 2)

// Set Na & K currents to 0 so that only T-currents are present
soma {
	gnabar_hh2 = 0
	gkbar_hh2 = 0
}

// Extra figure: peak low threshold spike amplitude for each current injection amplitude
w1 = 0
w2 = 200
```

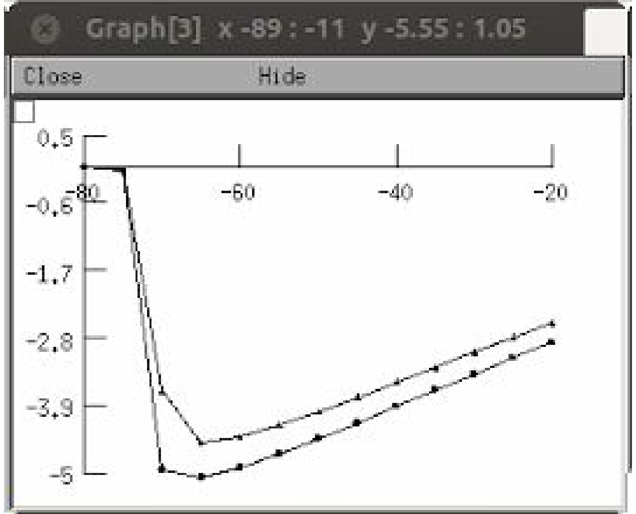
```
//dw = 100                         // for debugging
dw = 5
simulate2(w1, w2, dw, g[5], "O")

//
// VOLTAGE-CLAMP MODE
//

trans = 1000

print " "
print ">> Transient time of ",trans," ms"
print " "

Dt = 0.2
npoints = 500                // 100 ms is usually enough to find peak current

dt = 0.1                     // must be submultiple of Dt
tstart = trans
tstop = trans + npoints * Dt
runStopAt = tstop
steps_per_ms = 1/Dt

celsius = 24                 // temperature of John's experiments in VC
v_init = -70

soma El.vc.loc(0.5)          // put electrode in voltage-clamp mode

// Modify graph properties
changeview(g[0], tstart, tstop, -120, 20, 442, 102, 300.48, 200.32, 1)
changeview(g[1], tstart, tstop, -5, 0.5, 442, 365, 300.48, 200.32, 1)
changeview(g[2], tstart, tstop, -0.6, 0.4, 761, 102, 300.48, 200.32, 1)

// Fig 11A2
El.vc.dur[0] = trans
El.vc.dur[1] = 50
El.vc.dur[2] = 50
El.vc.amp[0] = -80
El.vc.amp[1] = -82.5
El.vc.amp[2] = -80

El.vc.rs = 5                 // default series resistance in tc200_vc.oc
```

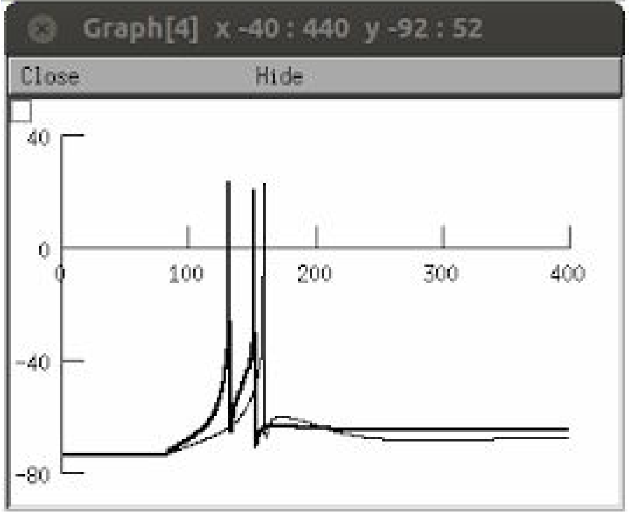
```
// Simulate
time = new Vector()             // A vector of time points used in the simulation
for ii = 0, tstop/dt time.append(ii*dt)
objref vci
vci = new Vector()              // For recordings of EI.vc.i
vci.record(&EI.vc.i)            // Records EI.vc.i
run()
if (stoppedrun()) {
        break
}
plotxy(time, vci, g[2], 1, 1)

// Fig 11A3
EI.vc.dur[0] = trans
EI.vc.dur[1] = 100
EI.vc.dur[2] = 0
hold = -115
EI.vc.amp[0] = hold
EI.vc.amp[1] = hold
EI.vc.amp[2] = hold
EI.map()                        // This makes sure that vamp[3] & vdur[3] are updated to
user-specified values

EI.vc.rs = 12                   // series resistance seems to be 12 MOhm in this figure

forall { g_pas = 0 }            // remove passive current everywhere

// Voltage clamp step voltage level range
c1 = -80
c2 = -20
//dc = 60                       // for debugging
dc = 5

// closest IV curve to detailed model wih dendritic density of 8.5e-5
// (total of 1.4434978)
soma.pcabar_itGHK = 6e-5
simulate3(c1, c2, dc, g[3], "T")

// increased density in order to get correct bursting behavior
// (total of 1.9246637)
soma.pcabar_itGHK = 8e-5
simulate3(c1, c2, dc, g[3], "O")
```

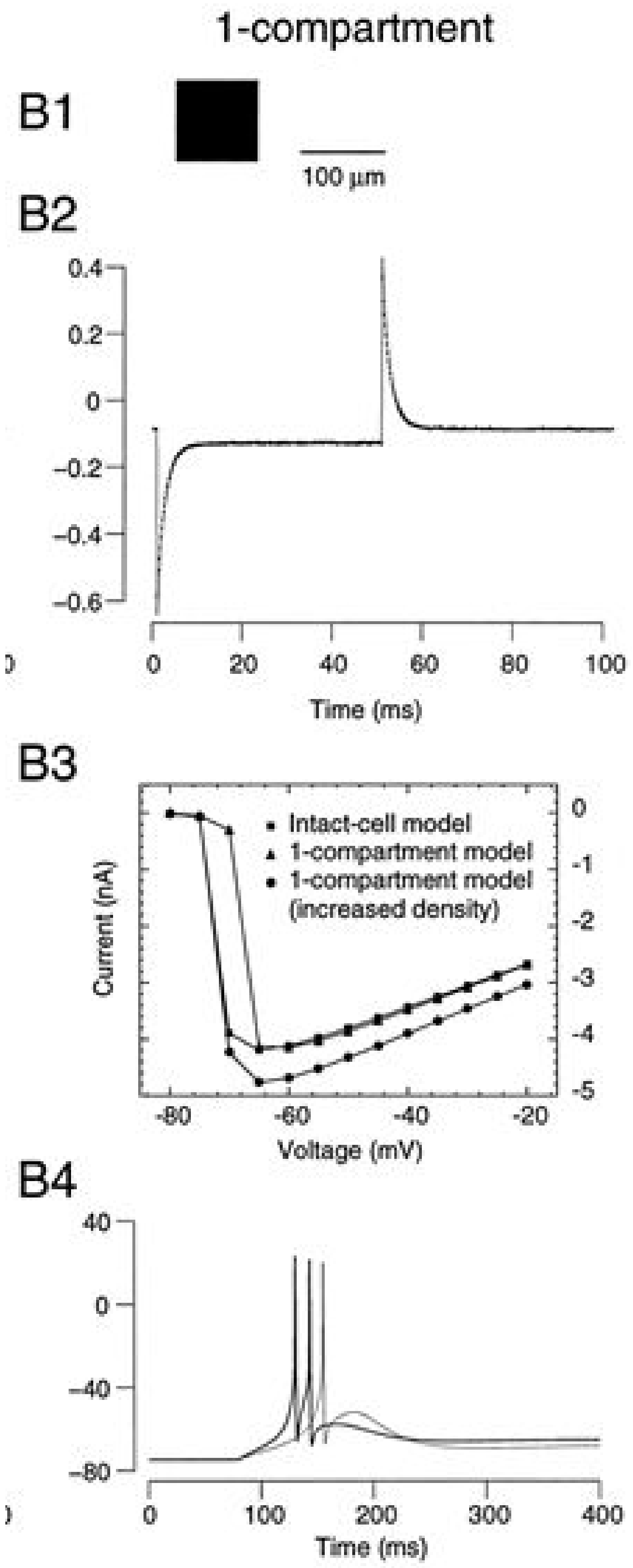

Graph[3] x 990 : 1110 y -0.7 : 0.5
Close
Hide
0.4
0.2
0
-0.2
-0.4
-0.6
1000
1020
1040
1060
1080
1100
Graph[4] x -89 : -11 y -5.55 : 1.05
Close
Hide
0.5
-0.6
-1.7
-2.8
-3.9
-5
-80
-60
-40
-20
Graph[5] x -40 : 440 y -92 : 52
Close
Hide
40
0
-40
-80
0
100
200
300
400

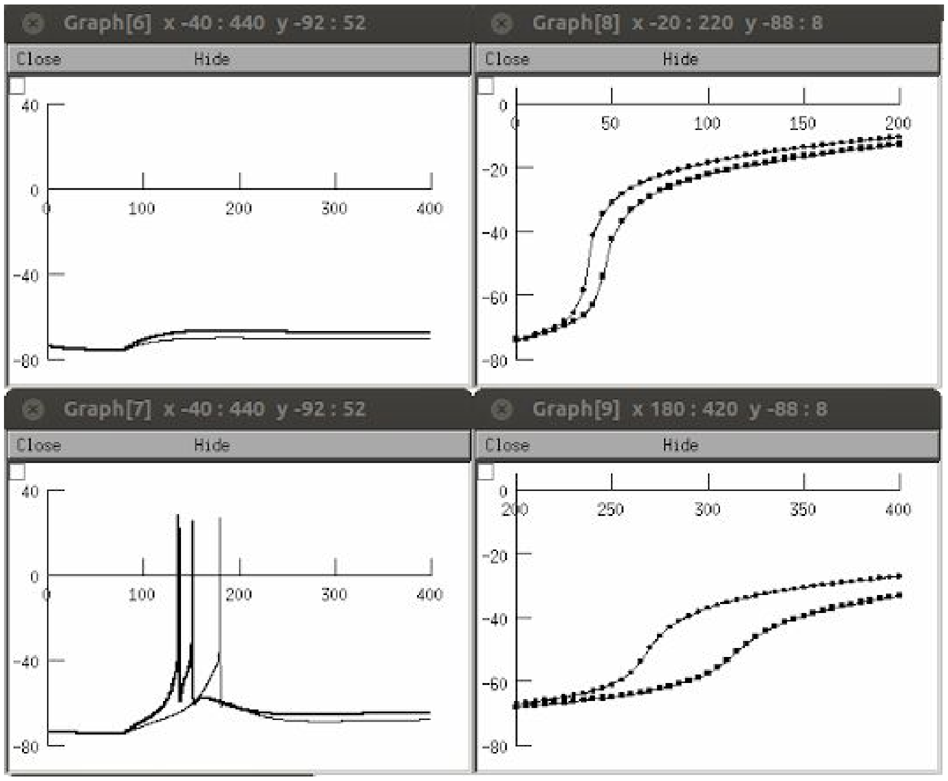

3-compartment
A1
100 µm
A2
Current (nA)
0.4
0.2
0
-0.2
-0.4
-0.6
0
20
40
60
80
100
Time (ms)
A3
Current (nA)
▪ Intact-cell model
▴ 3-compartment model
● 3-compartment model (increased density)
0
-1
-2
-3
-4
-5
-80
-60
-40
-20
Voltage (mV)
A4
Voltage (mV)
40
0
-40
-80
0
100
200
300
400
Time (ms)

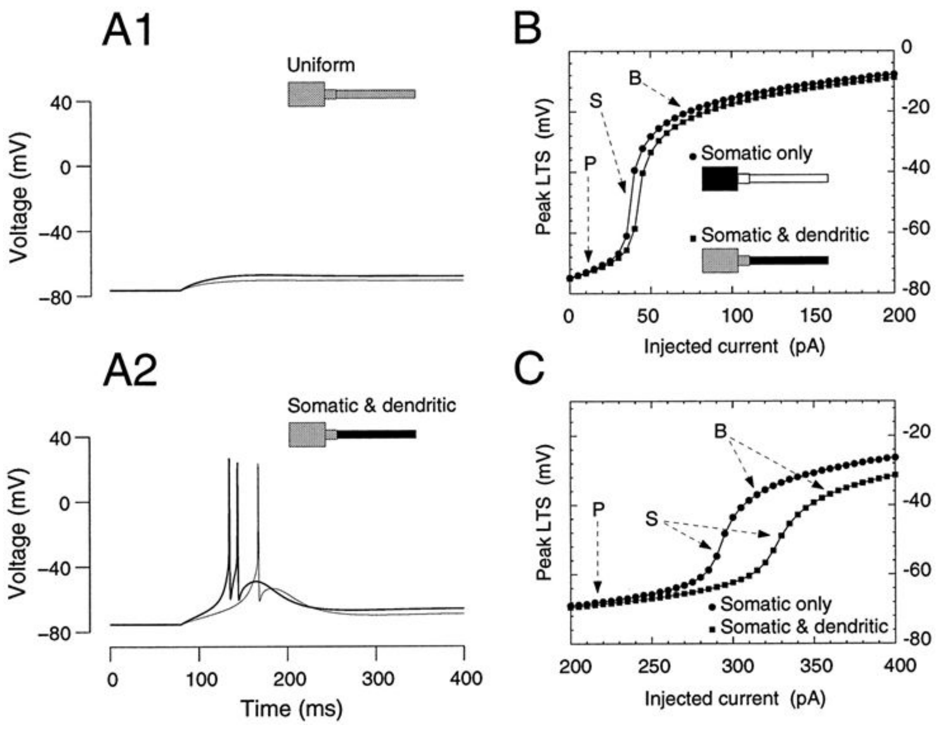

Graph[2] x 990 : 1110 y -0.7 : 0.5
Close
Hide
0.4
0.2
0
-0.2
-0.4
-0.6
1000
1020
1040
1060
1080
1100


Graph[3] x -89 : -11 y -5.55 : 1.05
Close
Hide
0.5
-0.6
-1.7
-2.8
-3.9
-5
-80
-60
-40
-20


Graph[4] x -40 : 440 y -92 : 52
Close
Hide
40
0
-40
-80
0
100
200
300
400


1-compartment
B1
100 μm
B2
0.4
0.2
0
−0.2
−0.4
−0.6
0
20
40
60
80
100
Time (ms)
B3
Intact-cell model
1-compartment model
1-compartment model (increased density)
Current (nA)
0
-1
-2
-3
-4
-5
-80
-60
-40
-20
Voltage (mV)
B4
40
0
−40
−80
0
100
200
300
400
Time (ms)


Graph[6] x -40 : 440 y -92 : 52
Close
Hide
Graph[8] x -20 : 220 y -88 : 8
Close
Hide
Graph[7] x -40 : 440 y -92 : 52
Close
Hide
Graph[9] x 180 : 420 y -88 : 8
Close
Hide


A1
Uniform
Voltage (mV)
A2
Somatic & dendritic
Time (ms)
B
Peak LTS (mV)
P
S
B
● Somatic only
■ Somatic & dendritic
Injected current (pA)
C
Peak LTS (mV)
P
S
B
● Somatic only
■ Somatic & dendritic
Injected current (pA)

5/23/2016~6/12/2016

Notes from Traub et al 2005

- Background Facts:
  1. Prior thalamocortical models:
     a. Tends to use small numbers of cells with one or a few compartments.
     b. Doesn't have multiple cell types and firing behaviors in cortical cells.
  2. **Gap junctions**:
     a. Experimental studies show that gap junctions are implicated in epileptogenesis.
     b. Modeling studies of hippocampal networks show that gap junctions:
        (1) can lower the extent of recurrent chemical synaptic excitation required for synchronization.
        (2) introduce a **very fast oscillation** (VFO, > **70 Hz**) that could occur on top of physiological sharp waves.
     c. There is evidence of gap junctions between **nRT** neurons in the form of **halothane-sensitive spikelets**.
  3. **Axonal coupling**:
     a. It is necessary in models for the occurrence of **gamma oscillations**.
     b. Spikelets occur in cortical interneurons
     c. There is staining for pannexin 2 throughout cortical layers 2-6
  4. **Very fast oscillations**:
     a. Superimposed on seizure burst complexes and interictal spikes in human epilepsy patients.
     b. Appears in somatosensory evoked potentials in rat barrel cortex.
     c. Superimposed on the "spike" component of spontaneous spike-wave seizures in ketamine-xylazine-anesthetized cats.
     d. Fast rhythmic bursting (**FRB**) cells do not exist in deep cortical layers.
  5. **Gamma/beta oscillations**:
     a. **Kainate** induces *persistent* (continues as long as the slice remains healthy) gamma oscillations in the rat **auditory cortex**, with maximum amplitude in superficial layers.
     b. Other *in vitro* gamma oscillations that have maximal amplitude in the superficial layers:
        - Interneuron gamma evoked by stimulating NMDA while AMPA is blocked.
        - Cortical gamma oscillations evoked by stimulating the thalamus in thalamocortical slices.
     c. **Gamma/beta oscillations** have different structure in **deep** versus **superficial** cortical layers, but have comparable amplitudes.
  6. **Sleep spindles**:

        a. **nRT** neurons burst on each spindle wave. They show a tonic depolarization *in vivo*, but a slight tonic hyperpolarization *in vitro* in ferret slices.
        b. **Multiphasic waves** are occasionally observed *in vivo*.
        c. *In vivo* in cats, sleep spindles are often followed by a run of **gamma oscillations** in both the cortex and the thalamus.
        d. An isolated portion of nRT appears to spindle on its own.
    7. **Seizures:**
        a. In the cat *in vivo*, **localized small-amplitude bursts** can be observed during epileptogenesis.
        b. There is an *in vivo* **genetic** rat model of spike-wave epilepsy in which intracortical inhibition is impaired.
        c. DIffuse cortical application of **penicillin** to the cat cortex can elicit a spike-wave-like epileptic pattern.
        d. A **0.1 Hz** spike-wave-like pattern has been observed *in vitro* during blockade of GABA_A and GABA_B receptors.
    8. **Fast runs:**
        a. *In vivo*, a **10 Hz fast run** can occur without the thalamus.
        b. In the penicillin cat model *in vivo*, **tonic-clonic seizures** occur once thalamic electrical activity is suppressed by injection of **hypertonic KCl**.
        c. In cat cortex *in vivo* after thalamectomy, fast runs can occur that go on for several minutes.
    9. **Cell types** in the **cortex**:
        a. Layer 2/3: Regular-spiking (**RS**) pyramidal cells, fast rhythmic bursting (**FRB**) pyramidal cells, fast-spiking (**FS**) interneurons, low-threshold spiking (**LTS**) interneurons
        b. Layer 4: **Stellate** cells (receives input from the thalamus)
        c. Layer 5: **Tufted** pyramidal cells (cortical outputs *not* to the thalamus), fast-spiking (**FS**) interneurons, low-threshold spiking (**LTS**) interneurons
        d. Layer 6: **Non-tufted** pyramidal cells (cortical outputs to the thalamus), fast-spiking (**FS**) interneurons, low-threshold spiking (**LTS**) interneurons
        e. Other: Neurogliaform cells, double bouquet cells, multipolar bursting neurons (not modeled)

- Hypothesis: Very fast oscillations are superimposed on epileptiform bursts and can be explained by the presence of gap junctions. A network model could be constructed to exhibit gamma oscillations, sleep spindles and epileptogenic bursts, very fast oscillations and spike-waves.
- Model:
    1. Basics:
        a. **3560** neurons.
        b. Each neuron has **dozens** of compartments (**50~100**)
        c. All neurons of a given type have the **same parameter set**

   d. The kinetics of submembrane [Ca2+] concentration was the same in all cell types. This is used to gate the slow AHP conductance (**sIAHP**) & one of the fast K conductances (**I_K(C)**)
   e. The same repertoire of **conductances** was used in all cell types
   f. Identical **kinetics** were used for same channels between different cell types, except:
      - **g_Na** & **g_K(DR)** were different between pyramidal cells and stellate/interneurons
      - **T-channel** kinetics were different between the reticular thalamus neurons (**nRT**) and thalamocortical relay neurons (**TCR**)
2. Channels:
   a. The "standard repertoire":
      g_Na (fast, transient)
      g_Na (persistent)
      g_K (**DR**; delayed rectifier)
      g_K (**A**; transient, inactivating)
      g_K (**sIAHP**; slow AHP)
      g_K (**C**; fast, voltage- and calcium-dependent)
      g_K (**K2**)
      g_K (**M**)
      g_Ca (high threshold)
      g_Ca (**T**; low threshold)
      g_h (relatively slow anomolous rectifier)
3. Network:
   a. The cortex is **1-dimensional** in depth; space is not defined within the thalamus.
   b. Cell types in each layer follow Background Fact **#6a~d**:
      - Layer 2/3:
        - **1000** regular-spiking (**RS**) pyramidal cells (74 compartments)
        - **50** fast rhythmic bursting (**FRB**) pyramidal cells (74 compartments)
        - Fast-spiking (**FS**) interneurons:
          - **90** basket cells (59 compartments)
          - **90** axoaxonic (chandelier) cells (59 compartments)
        - **90** low-threshold spiking (**LTS**) interneurons (59 compartments)
      - Layer 4:
        - **240 spiny stellate** cells (receives input from the thalamus, 59 compartments)
      - Layer 5:
        - **800 tufted IB** pyramidal cells (cortical outputs *not* to the thalamus, 61 compartments)

            - **200 tufted RS** pyramidal cells (cortical outputs *not* to the thalamus, 61 compartments)
        - Layer 6:
            - **500 non-tufted** pyramidal cells (cortical outputs to the thalamus)
            - Fast-spiking (**FS**) interneurons:
                - **100** basket cells (59 compartments)
                - **100** axoaxonic (chandelier) cells (59 compartments)
            - **100** low-threshold spiking (**LTS**) interneurons (59 compartments)
        - Thalamus:
            - **100** nucleus reticularis (**nRT**) neurons (59 compartments)
            - **100** thalamocortical relay (**TCR**) neurons (137 compartments)
    c. Synapses:
        - AMPA, NMDA, GABA_A (no GABA_B)
    d. Gap junctions are present between:
        - Dendrites of cortical interneurons
        - Dendrites of nRT cells
        - Dendrites of TCR cells
        - Axons of Layer ⅔ RS & FRB pyramidal cells
        - Axons of Layer 4 spiny stellates
        - Axons of Layer 5 **Tufted** pyramidal cells
        - Axons of Layer 6 **Non-tufted** pyramidal cells
    e. Inputs:
        - Only **steady bias currents**.
- Experimental Methods:
    1. *In vitro*: 450 um slices from adult male Wistar rats (150-250 g) that contain **primary** and **secondary auditory** regions and **secondary parietal** region.
    2. *In vivo*:
        a. Adult male Sprague-Dawley rats (350-450 g) anesthesized with **phenobarbital**. **Buprenorphine** was also administered to provide analgesia. Paralyzed with **gallamine triethiodide** and artificially ventilated.
        b. EEG: **Bipolar electrodes** inserted into cortex
        c. Intracellular recordings: Craniotomy exposed surface of **barrel cortex** (1.0-3.0 mm posterior to bregma, 4.0-7.0 mm lateral to the midline)
- Results:
    1. **Persistent gamma oscillations (~30 Hz)** could be generated in the cortical network with the thalamus removed.
        a. Principal cell axons spike => interneurons activated => GABA inhibition on principal cells for **tens of ms** (gamma period).

   b. Highest amplitude in the **superficial layers**.
   c. Superficial **FRB** pyramids discharge on approximately **every other burst**.
2. **Sleep spindles (~16Hz)** can be generated in the thalamic portion of the network (corticothalamic EPSCs are at most 0.2 nS).
   a. Model thalamic cells can generate rhythmic bursts at **~5 Hz**
   b. Spindles are initiated by a spontaneous burst in **nRT**, has a **waxing/waning** course, and stops on its own.
   c. Calcium-dependent **I_h** kinetics are **not needed** for cessation.
   d. **nRT** neurons burst on each wave, while **TCR** neurons only fire on a fraction of the waves.
   e. In Layer 4 **spiny stellates**, **coherent depolarizations** and **multiphasic waves** are observed.
   f. In Layer 5 **Tufted** pyramidal cells, a superposition of synaptic excitation and inhibition is observed.
   g. In the superficial layers, there is a mixture of spindles and gamma oscillations
   h. Sleep spindles are often followed by a run of **gamma oscillations** in just the cortex and not the thalamus (in contrast to Background Fact **#6c**).
   i. Isolating nRT from layer 6 pyramids and TCR neurons abolishes spindles, in contrast to Background Fact **#6d**.
   j. However, increasing the bias current to the isolated nRT produces a **6 Hz** oscillation that requires **gap junctions** but not within-RT inhibition
3. With the **thalamus removed**, **epileptiform bursts** with **VFO** could be generated as long as there is disinhibition and strong recurrent excitation of **spiny stellates**
   a. When IPSCs are reduced to **0.1x**, EPSCs reduced to **0.25x**, gap junctions present between **superficial pyramids**, between **spiny stellates**, between **Layer 6 RS pyramids**, and depolarizing bias currents of **0.1 nA** to deep pyramids, the following could be observed in the cortical network with thalamus removed:
      - **Interictal bursts**.
      - A large paroxysmal depolarization shift (**PDS**) at the Layer 6 pyramids.
      - Very fast oscillations (**VFO**, **~300 Hz**) in field recordings, especially at 1 mm.
      - Low-amplitude oscillations (**~20 Hz**) in field recordings.
      - Partially synchronized bursts
   b. When IPSCs are reduced to **0.05x**, EPSCs increased to **2x**, gap junctions conductances "high" between **superficial pyramids**, between **spiny stellates**, between **Layer 5 RS pyramids**, between **Layer 6 RS pyramids**, the following could be observed in the cortical network with thalamus removed:
      - A **10 Hz fast run (polyspike)** occurs and terminates spontaneously.

  - Synchronized bursts with VFO occurs when gap junctions exist only in **Layer 4 spiny stellates**.
  - When IPSCs are reduced only to **0.1x**, **double bursts** occur.
  - When IPSCs are reduced only to **0.25x**, epileptiform activity is abolished.
  - When IPSCs are reduced only to **0.75x**, **gamma oscillations** occur in the superficial layers.

c. Application of **kainate** (400 nM) + **picrotoxin** (40 μM, blocks GABA_A receptors) + **CGP-55845A** (10 μM, blocks GABA_B receptors) in the rat **auditory cortex** generated epileptiform **double bursts**.
  - **Layer 4 spiny stellates** fired throughout the double bursts, all other cells were in phase with field deflections.
  - Field recordings were similar to those in the model (IPSC reduced to **0.1x**), with prominent **VFO** in both.
  - However, simulations showed inter-burst spikes in and less-depolarized bursts in Layer 2/3, unlike the recordings.

d. When IPSCs are reduced to **0x**, AMPA conductances reduced to **0.25x**, NMDA conductances increased to **1.25x** (tau = 15 ms), gap junctions conductances “high” between **superficial pyramids**, between **spiny stellates**, between **Layer 6 RS pyramids**, the following could be observed in the cortical network with thalamus removed:
  - **Prolonged single epileptiform bursts** with **continuous spiny stellate** firing and **VFO** in field activity
  - Similar to experimental recordings when **kainate** + **picrotoxin** is applied (but not CGP-55845A)

e. When IPSCs are reduced to **0.2x**, AMPA conductances in spiny stellates reduced to **0.25x**, gap junctions present between **superficial pyramids**, between **spiny stellates**, between **Layer 6 RS pyramids**, and depolarizing bias currents of **0.35-0.45 nA** to deep pyramids, the following could be observed in the cortical network with thalamus removed:
  - **3-Hz spike-wave-like** oscillations with **VFO** in field activity.
  - Spiny stellate firing is brief, **layer 6 pyramid** bursting prolonged.
  - Only current with appropriate time course to gate the 3 Hz oscillation is the **slow calcium-mediated AHP current**.

f. VFO: The power spectrum over 6 “spikes” shows a peak near **100 Hz** and smaller peaks around 200 Hz & 300-400 Hz
  - Average signals over Layer 2/3 & Layer 4 show clear correlation between spike firing times **within populations**, but not between populations (interpretation: gap junctions only exist within each layer, not between layers)

4. *In vivo* recordings in anesthetized rats:
   a. Spontaneous seizures in anesthetized rats was a mixture of **10-15 Hz** fast runs and **1-4 Hz** spike-wave or polyspike-wave activity.

    b. Seizures were paroxysmal, associated with sleep spindles, and tend to recur every **1~3 min**.
    c. EEG spikes corresponded with **paroxysmal depolarizing shifts** (PDS) that were larger than **20 mV**.
    d. There is significant **spike inactivation** followed by a **long depolarizing tail** that is not present in simulations (possibly due to the omission of GABA_B receptors).
    e. In 5 cells the depolarization occurring during EEG spikes was preceded and crowned by bursts of **short spikelets** of **2-7 mV**, resembling the spikelets resulting from **antidromic spikes** in the model.
5. With the **thalamus connected**, the model shows similar spike-waves as before with nRT bursting with each “EEG spike” and TCR activity largely suppressed.
    a. TCR neurons fire a **single action potential** per “EEG spike.”
    b. When the simulation was repeated with axonal gap junctions between cortical principal cells removed, spike-waves did not occur.
    c. When the simulation was repeated with AMPA conductances in spiny stellates at **2x** instead of at **0.25x**, **1.6~2.5 Hz polyspikes** (containing spikes at about **13 Hz**) occur, with Layer 4 spiny stellates firing throughout the polyspike (other cells fire with each spike).
    d. With the thalamus disconnected again, the polyspike transforms into a **10 Hz** fast run, suggesting that the thalamus exerts a **restraining effect** on cortical epileptogenesis

- Discussions:
    1. The hypothesis is true.
    2. **Axonal coupling** between principal cortical neurons could explain **very fast oscillations** during seizures
    3. Recurrent excitatory interactions between **Layer 4 spiny stellate** cells appear important for epileptogenesis, allowing EEG spikes to occur, and allowing EEG spikes to become polyspikes
    4. Future directions:
        a. A clear epoch of **low-amplitude VFO** *before* epileptiform bursts has been observed in hippocampal slices and in children with seizures caused by a cortical dysplasia and in anesthetized cats, but this has not been reproduced in the current model.
        b. Recurrent excitatory interactions between spiny stellates may be counterbalanced by the ability of the synapses to undergo long-term depression that is dependent on **mGluR2** receptors. Are these receptors ineffective in individuals predisposed to seizures?
        c. A drug that targets **NR2C receptors** could have useful antiepileptic effects.

Plan for next week

1. Examine the codes for the Destexhe et al 1996 model
2. Read and take notes from Destexhe et al 1996
3. Reproduce figures in Destexhe et al 1996

Plan for the future

1. Reproduce figures in Destexhe et al 1998b
2. Reproduce figures in Destexhe 1998
3. Reproduce figures in Destexhe et al 2001
4. Reproduce figures in Traub et al 2005
5. Compile relevant papers that have cited Destexhe et al 1996 and/or have used the Destexhe et al 1996 model
6. Compile relevant papers that have cited Destexhe et al 1998b and/or Destexhe 1998 and/or Destexhe et al 2001 and/or have used the Destexhe et al 1998, 2001 model
7. Compile relevant papers that have cited Traub et al 2005 and/or have used the Traub et al 2005 model
8. Refine reproduction of figures in Destexhe et al 1998a
9. Compile relevant papers about absence epilepsy
10. Read and write down notes for Chen et al 2014 and Chen et al 2015
11. Read and write down notes for Zhao et al 2015
12. Examine the codes for the Chen et al 2014, 2015 model
13. Examine the codes for the Zhao et al 2015 model
14. Examine the codes for the Traub et al 2005 model
15. Go through the NEURON Hands-On Course
16. “Reproduce CSD graph” exercise
17. Examine Christine’s & Mark’s codes
18. Finish NEURON Book Appendix A1
19. Figure out how to export NEURON to Matlab
20. Complete the NEURON Tutorial
21. Understand NEURON Ch 7 & Ch 8
22. Resolve all NEURON Book questions
23. Read Abbott et al 2016 (“Building functional networks of spiking model neurons”)
24. Read Markram et al 2015 (“Reconstruction and Simulation of Neocortical Microcircuitry”)
25. Read Kragel & LaBar 2016 (“Decoding the Nature of Emotion in the Brain”)
26. Read Izhikevich: Dynamical Systems in Neuroscience
27. Read Dayan & Abbott: Theoretical Neuroscience
28. Derivation & shape of the **Goldman–Hodgkin–Katz flux equation**

5/26/2016~6/5/2016

Reproduce Figures in Destexhe et al 1998a (part 2)

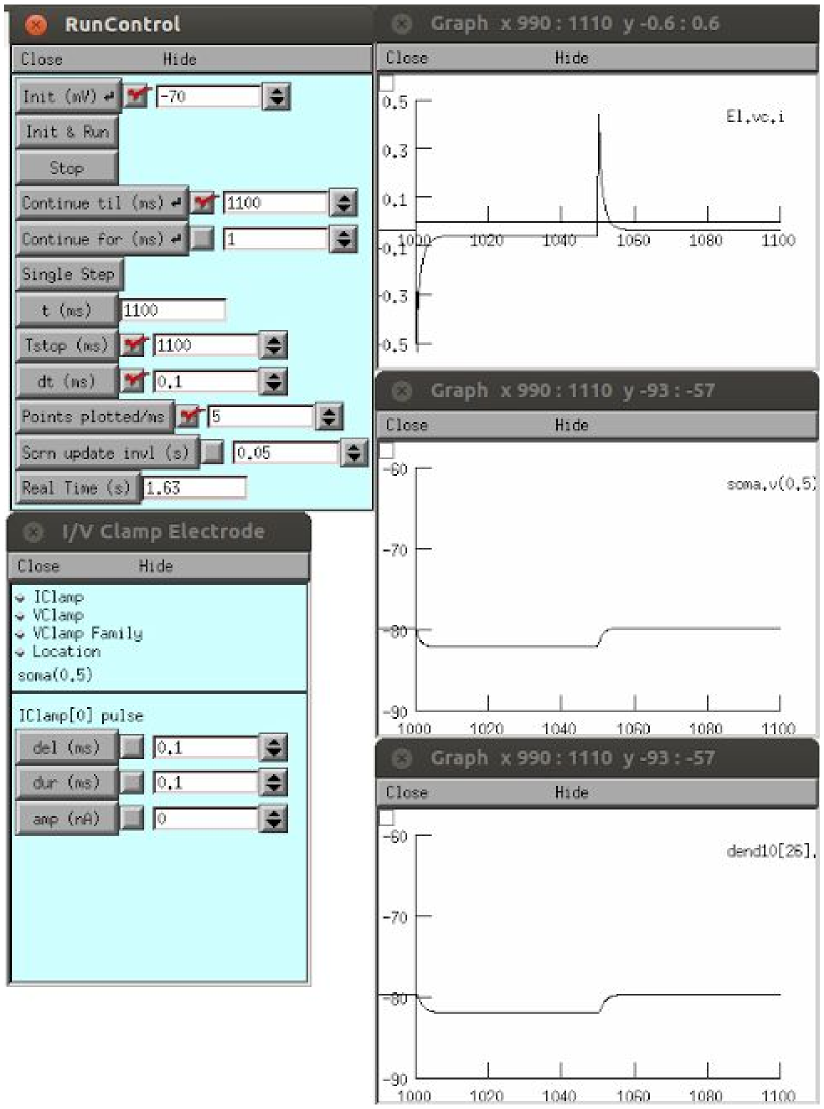
- **Figure 1C**
    - Voltage clamp of the detailed cell model thats fits to experimental data
- Modified code of **tc200_vc.oc**:

npoints = 500

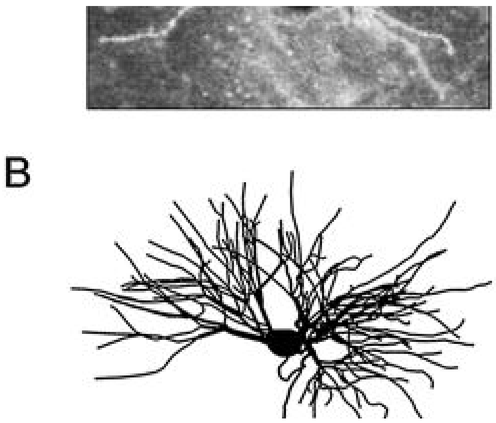
Restore **passive current** everywhere by commenting out **//**forall { g_pas = 0 }

Name	Description	New value	Notes
vc.dur[0]	Duration of voltage clamp level 1 [ms]	**1**	
vc.dur[1]	Duration of voltage clamp level 2 [ms]	**50**	
vc.dur[2]	Duration of voltage clamp level 3 [ms]	**50**	
vc.amp[0]	Amplitude of voltage clamp level 1 [mV]	**-80**	
vc.amp[1]	Amplitude of voltage clamp level 2 [mV]	**-82.5**	
vc.amp[2]	Amplitude of voltage clamp level 3 [mV]	**-80**	

addgraph("El.vc.i",**-0.5,0.5**)
addgraph("soma.v(0.5)",**-90,-60**)
addgraph("dend10[26].v(0.5)",**-90,-60**)

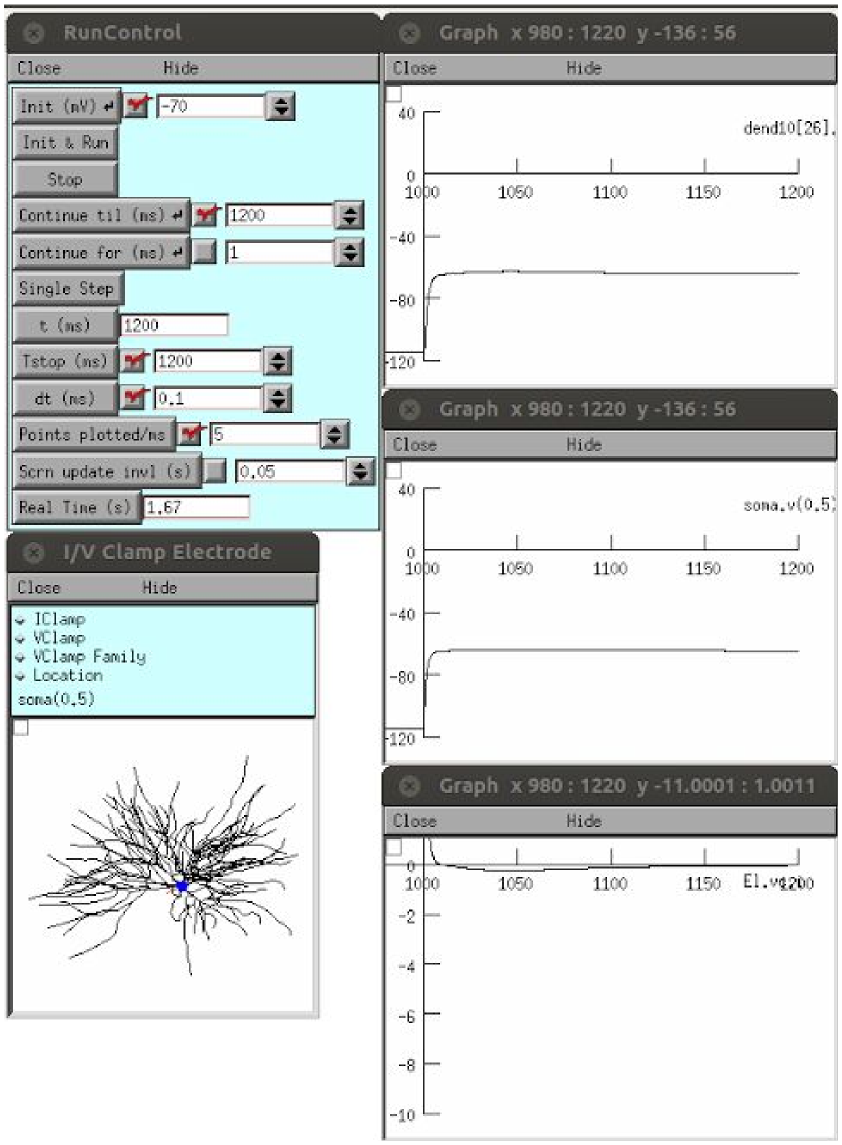


```
Init (mV)             -70
Continue til (ms)     1100
Continue for (ms)     1
t (ms)                1100
Tstop (ms)            1100
dt (ms)               0.1
Points plotted/ms     5
Scrn update invl (s)  0.05
Real Time (s)         1.63

soma(0.5)
IClamp[0] pulse
del (ms)   0.1
dur (ms)   0.1
amp (nA)   0
```



- **Figure 6A & 6B**
    - Voltage clamp of the detailed cell model:
      Condition at **-115 mV** for **1 sec** and stepping to **-65 mV**
    - **6A**: Uniform T-channel density
      6B: T-channel density higher in distal dendrites
    - Code as in **tc200_vc.oc**, just toggle between:

localize(1.7e-5,corrD*8e-5,corrD*8e-5):



RunControl

Close Hide

Init (mV) -70

Init & Run

Stop

Continue til (ms) 1200

Continue for (ms) 1

Single Step

t (ms) 1200

Tstop (ms) 1200

dt (ms) 0.1

Points plotted/ms 5

Scrn update invl (s) 0.05

Real Time (s) 1.67

I/V Clamp Electrode

Close Hide

- IClamp
- VClamp
- VClamp Family
- Location

soma(0.5)

Graph x 980 : 1220 y -136 : 56

Close Hide

dend10[26].

Graph x 980 : 1220 y -136 : 56

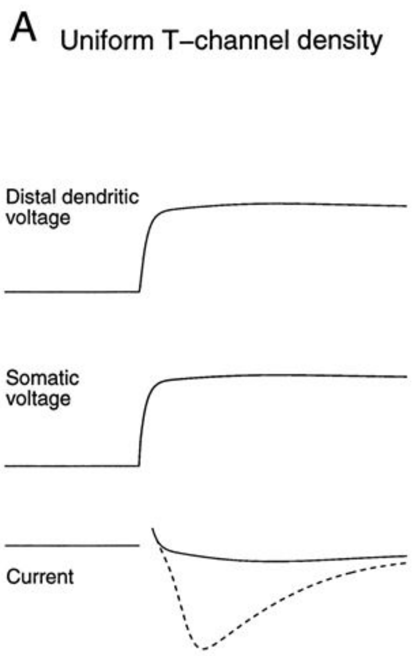
Close Hide

soma.v(0.5)

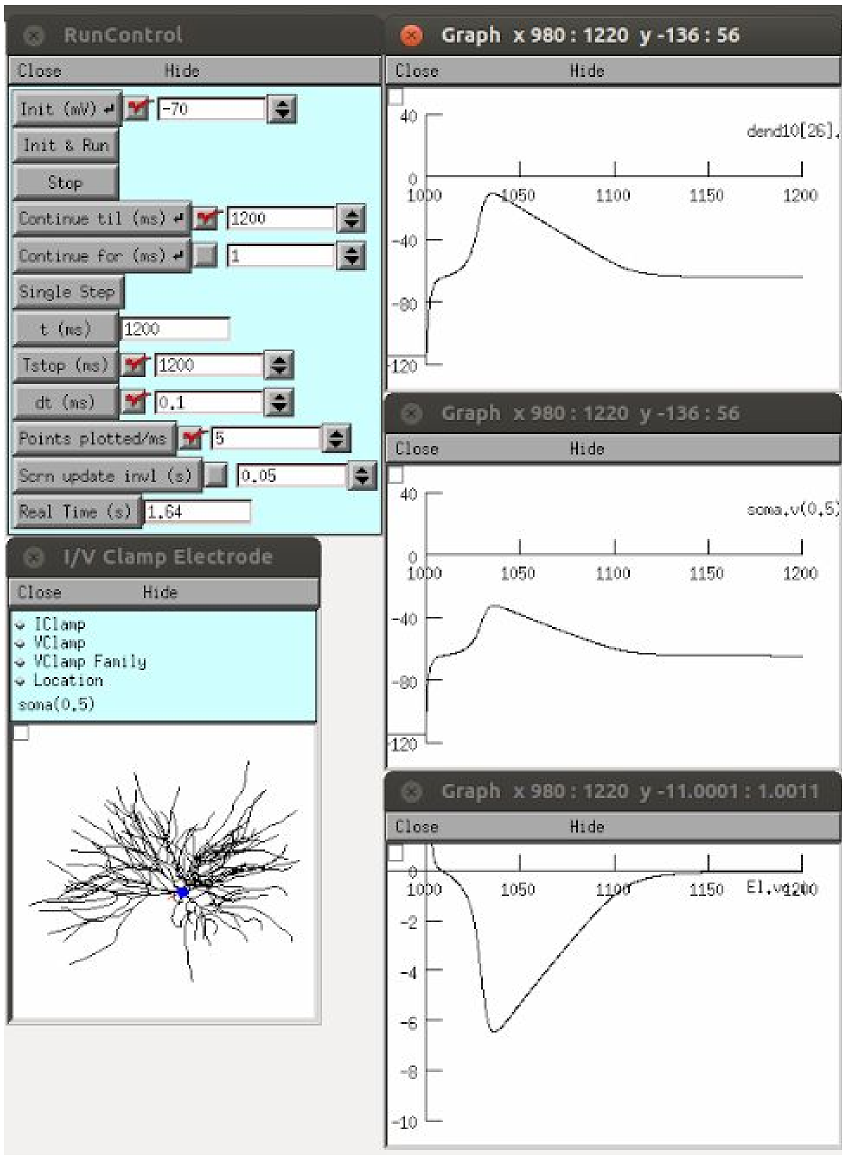
Graph x 980 : 1220 y -11.0001 : 1.0011

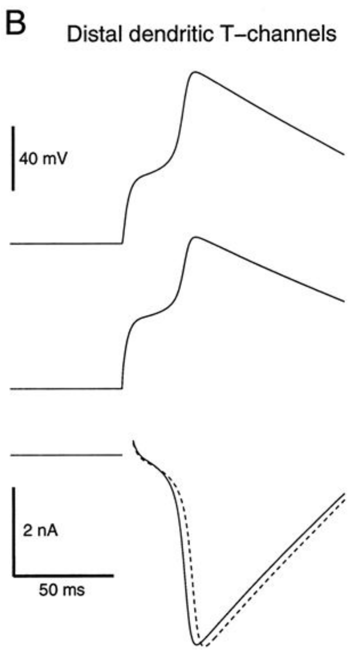
Close Hide

El.vi



and **localize(1.7e-5,corrD*1.7e-5,corrD*1.7e-5)**:





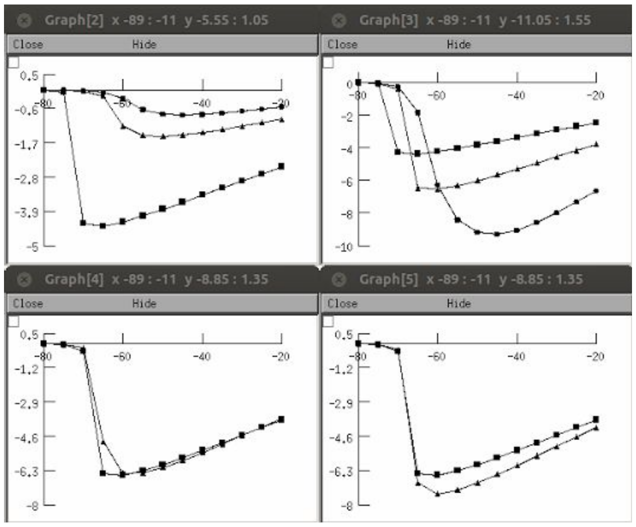
- **Figure 7A~7D**
    - I_V curves for the detailed cell model:
      Condition at **-115 mV** for **1 sec** and stepping to various values
    - **7A**: Vary **total number of T-channels**
      7B: Vary **series resistance**
      7C: Holding **peak current at soma** constant, compare having T-channels only at the soma or having more of them at distal dendrites
      7D: Holding **total number of T-channels** constant, compare having T-channels only at the soma or having more of them at distal dendrites
    - Modified code of **El.oc** as for Figure 5

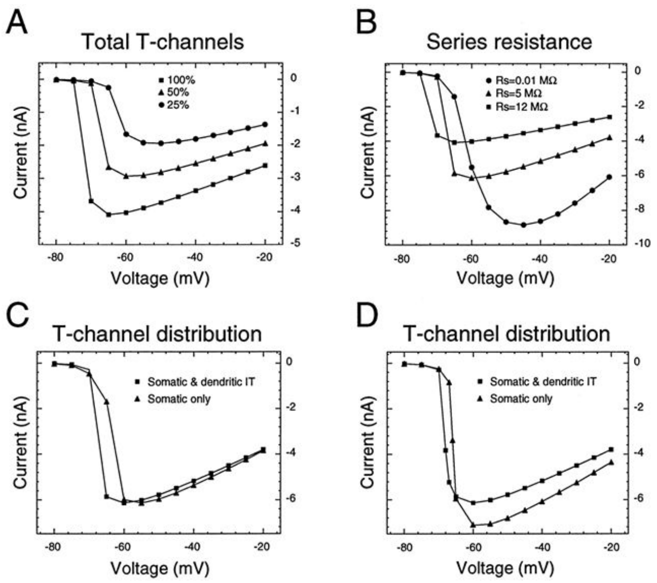
- Modified code of **loc200.oc**:
  Added procedure **localize2()** that changes soma and proximal dendrites separately (proximal dendrites are lumped together with middle dendrites)
- Modified code of **tc200_vc.oc** as for Figure 5, with the addition of:

```
proc addgraph2() { local ii
        ngraph = ngraph+1
        ii = ngraph - 1
        g[ii] = new Graph(0)
        g[ii].view($1, $3, $2-$1, $4-$3, $5, $6, $7, $8)
        g[ii].xaxis()
        g[ii].yaxis()
        g[ii].family(1)                         // turn Keep Lines on
        g[ii].save_name("graphList[0].")
        graphList[0].append(g[ii])
        last = ii
}
addgraph("El.vc.i", tstart, tstop, -10, 0.5, 761, 102, 300.48, 200.32)
addgraph("soma.v(0.5)", tstart, tstop, -120, 20, 761, 365, 300.48, 200.32)
load_file("rig.ses")
objref testvoltagelevel
proc simulate() {
        // Erase first two graphs
        g[0].erase()
        g[1].erase()
        // Set VClamp Family specifications (used in varyamp())
        El.x1 = $1
        El.x2 = $2
        El.dx = $3
        n = (El.x2-El.x1)/El.dx + 1             // number of test voltage levels
        testvoltagelevel = new Vector(n,0)      // For storing each test voltage level
        for i=0, n-1 testvoltagelevel.x[i] = El.x1 + El.dx*i
        El.varyamp(1)
}
proc plotIV() {
        El.vci_peak.line($o1, testvoltagelevel)
        El.vci_peak.mark($o1, testvoltagelevel, $s2, 6)
}
// Fig 7A
addgraph2(-82.5, -17.5, -5, 0.5, 1080, 102, 300.48, 200.32)// Create graph of peak current over
test voltage level
El.vc.rs = 12
// T-current density 100%
simulate(-80, -20, 5)
```

```
plotIV(g[last], "S")
// 50%
localize(1.7e-5*0.5,corrD*2.5e-5*0.5,corrD*2.5e-5*0.5)
simulate(-80, -20, 5)
plotIV(g[last], "T")
// 25%
localize(1.7e-5*0.25,corrD*2.5e-5*0.25,corrD*2.5e-5*0.25)
simulate(-80, -20, 5)
plotIV(g[last], "O")
// Reset T-current densities & series resistance
localize(1.7e-5,corrD*8e-5,corrD*8e-5)
El.vc.rs = 5
// Fig 7B
addgraph2(-82.5, -17.5, -10, 0.5, 1399, 102, 300.48, 200.32)          // Create graph of peak
current over test voltage level
// Series resistance = 5 MOhm
simulate(-80, -20, 5)
plotIV(g[last], "T")
// Series resistance = 0.01 MOhm
El.vc.rs = 0.01
simulate(-80, -20, 5)
plotIV(g[last], "O")
// Series resistance = 12 MOhm
El.vc.rs = 12
simulate(-80, -20, 5)
plotIV(g[last], "S")
// Reset series resistance
El.vc.rs = 5
// Fig 7C
addgraph2(-82.5, -17.5, -8, 0.5, 1080, 365, 300.48, 200.32)// Create graph of peak current over
test voltage level
// Somatic & dendritic IT
simulate(-80, -20, 5)
plotIV(g[last], "S")
// Somatic only
localize2(42.7e-5,corrD*0,corrD*0)
simulate(-80, -20, 5)
plotIV(g[last], "T")
// Reset dendritic densities
localize(1.7e-5,corrD*8e-5,corrD*8e-5)
// Fig 7D
addgraph2(-82.5, -17.5, -8, 0.5, 1399, 365, 300.48, 200.32)// Create graph of peak current over
test voltage level
```

```
// Somatic & dendritic IT
simulate(-80, -20, 5)
plotIV(g[last], "S")
// Somatic only
localize2(53.3e-5,corrD*0,corrD*0)
simulate(-80, -20, 5)
plotIV(g[last], "T")
// Reset dendritic densities
localize(1.7e-5,corrD*8e-5,corrD*8e-5)
```


Graph[2] x -89 : -11 y -5.55 : 1.05
Graph[3] x -89 : -11 y -11.05 : 1.55
Graph[4] x -89 : -11 y -8.85 : 1.35
Graph[5] x -89 : -11 y -8.85 : 1.35
Close
Hide



- **Figure 8**
    - Tail currents
    - Modified code of **EI.oc**:

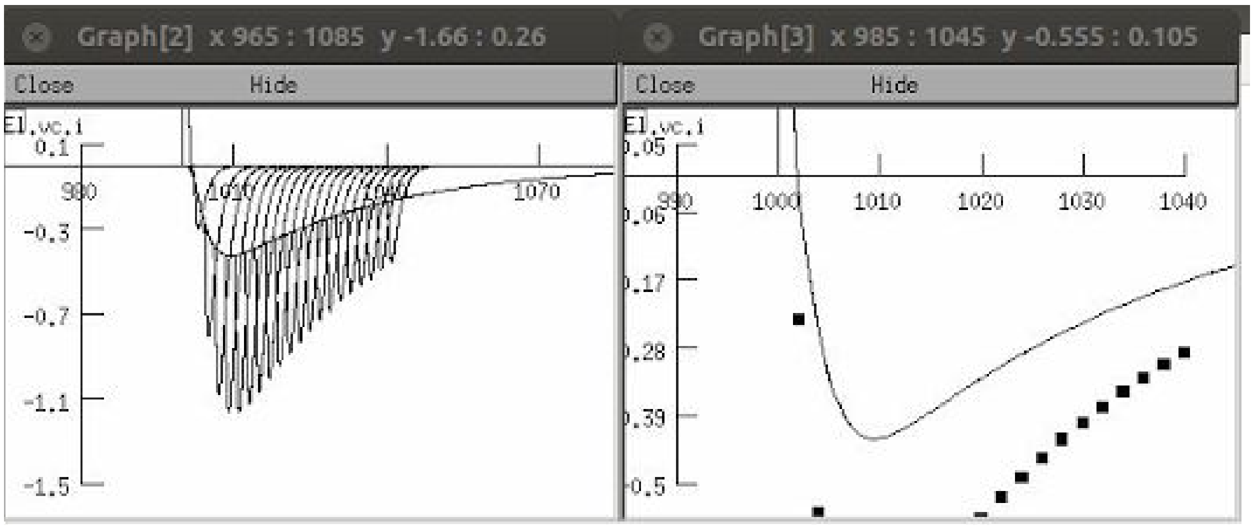
```
public stim, vc, unmap, map, v1, installIclamp, installVclamp, varyamp, varydur, vci_cap,
vci_peak, vci_tail, x1, x2, dx
external run, set_v_init, stoppedrun, addplot, addgraph2, tstop, plotxy, gvci, sl

objref vci_tail, time
proc varydur() {local i, ii, x, n, min, left, right, base, old1, old2
        i = $1
        set_vclamp()
        n = (x2-x1)/dx + 1              // number of step durations
        vci_tail = new Vector(n,0)      // For storing tail currents for each voltage step duration
        objref time
        time = new Vector()
        for ii = 0, tstop/dt time.append(ii*dt)
        for (x = x1; x <= x2; x = x + dx) {
                if (i == 1) {
                        old1 = vc.dur[1]
                        old2 = vc.dur[2]
                        vc.dur[1] = x
                        vc.dur[2] = old2 + (old1 - x)
                } else {
                        vc.dur[i] = x
                }
                vci = new Vector()      // For recordings of capacitave transients
                vci.record(&vc.i)
                forsec sl {
                        pcabar_old_itGHK = pcabar_itGHK
                        pcabar_itGHK = 0
                }
                run()
                if (stoppedrun()) {
                        break
                }
                vci_cap = new Vector()          // To save capacitave transients
                vci_cap.copy(vci)
                vci = new Vector()      // For recordings of vc.i
                vci.record(&vc.i)
                forsec sl {
                        pcabar_itGHK = pcabar_old_itGHK
                }
                run()
                if (stoppedrun()) {
                        break
                }
                vci.sub(vci_cap)        // Capacitive components subtracted
                print "The size of vci is ", vci.size()
                plotxy(time, vci, gvci, 1)
                left = gfloor(vc.dur[0]/dt)
                right = gfloor((vc.dur[0]+100)/dt)
                base = gfloor((vc.dur[0] + x)/dt)
                vci_tail.x[(x-x1)/dx] = vci.min(left,right) - vci.x[base]
        }
        print "The size of vci_tail is ", vci_tail.size()
}
```

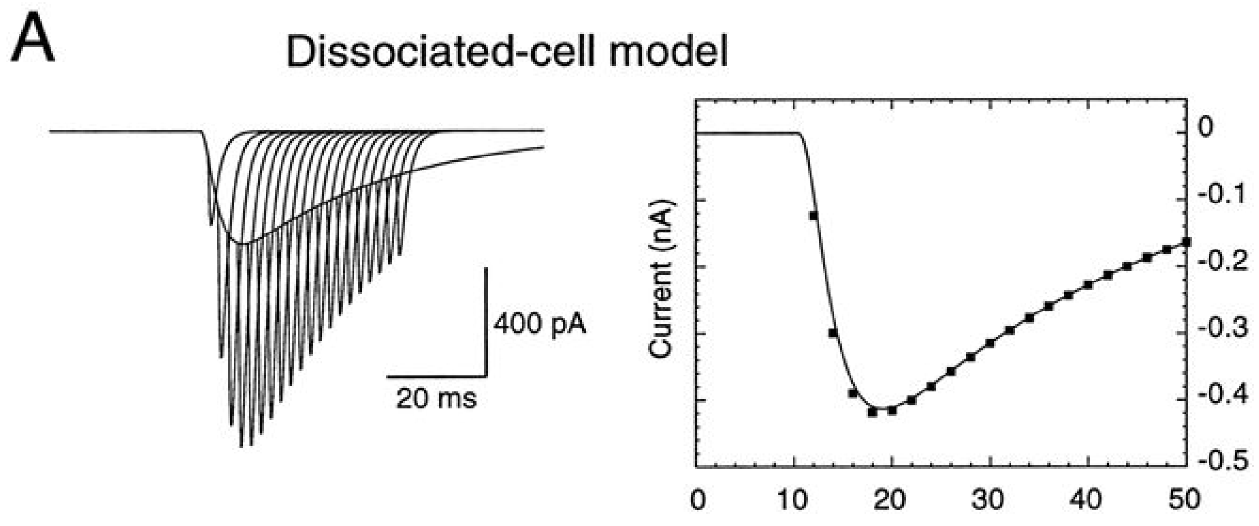
- Modified code of **tcD_vc.oc** & **tc200_vc.oc** (codes for former is shown, those for the latter is similar):

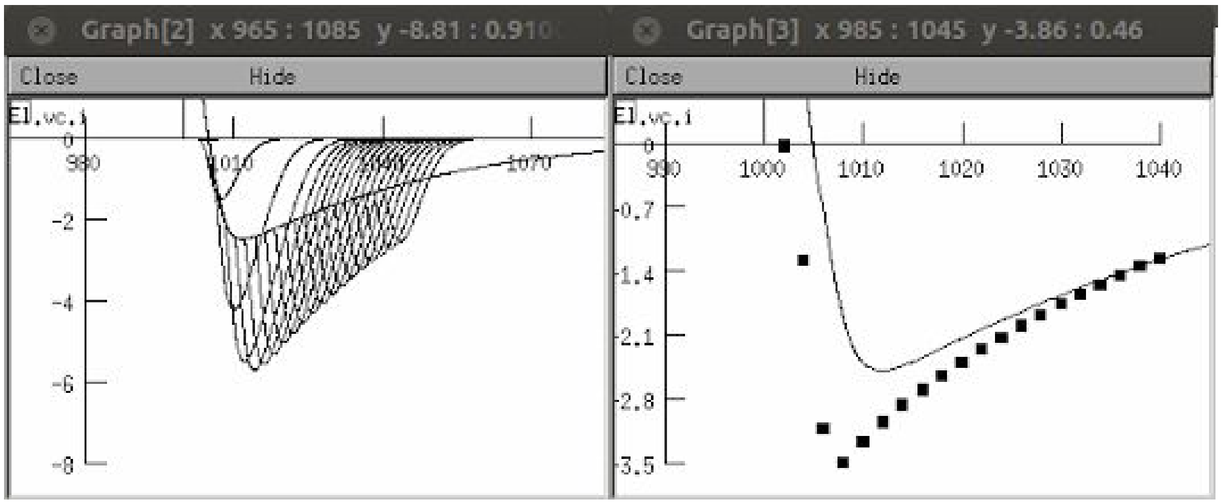
```
objectvar g[20], gvci                    // max 20 graphs
ngraph = 0

proc addgraph() { local ii      // define subroutine to add a new graph for a variable that is
simulated
                                // addgraph("variable", t_min, t_max, var_min, var_max,
window_left, window_top, window_width, window_height)
        ngraph = ngraph+1
        ii = ngraph-1
//      g[ii] = new Graph()
        g[ii] = new Graph(0)    // is not mapped; will be sized and placed with the .view() function
//      g[ii].size(tstart,tstop,$2,$3)
        g[ii].view($2, $4, $3-$2, $5-$4, $6, $7, $8, $9)
        g[ii].xaxis()
        g[ii].yaxis()
        g[ii].addvar($s1,1,0)
        g[ii].family(1)         // turn Keep Lines on
        g[ii].save_name("graphList[0].")
        graphList[0].append(g[ii])
}

proc addgraph2() { local ii                     // Create graph for plotting in general
                                                // addgraph2(x_min, x_max, y_min, y_max,
window_left, window_top, window_width, window_height)
        ngraph = ngraph+1
        ii = ngraph - 1
        g[ii] = new Graph(0)
        g[ii].view($1, $3, $2-$1, $4-$3, $5, $6, $7, $8)
        g[ii].xaxis()
        g[ii].yaxis()
        g[ii].family(1)                         // turn Keep Lines on
        g[ii].save_name("graphList[0].")
        graphList[0].append(g[ii])
        last = ii
}

proc addshape() { local ii      // define subroutine to add a new shape
                                // addshape()
        ngraph = ngraph+1
        ii = ngraph-1
        g[ii] = new PlotShape()
        g[ii].scale(-130,50)
}
```

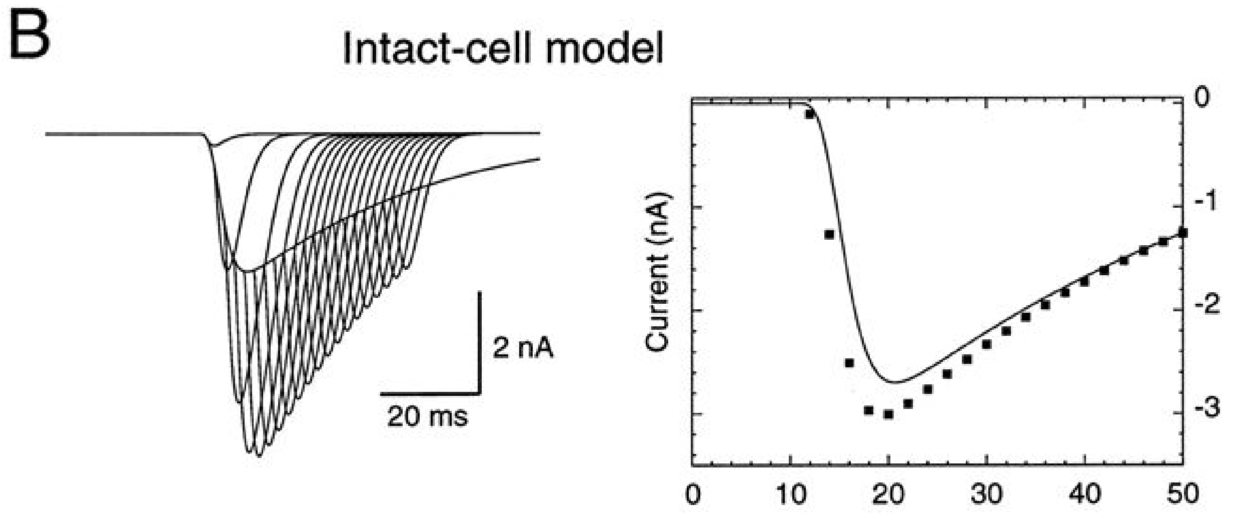
```
objref vsd
proc simulate() { local n                     // voltage step duration runs from $1 to $2 with
interval $3
        // Set VClamp Family specifications (used in varydur())
        El.x1 = $1
        El.x2 = $2
        El.dx = $3
        n = (El.x2-El.x1)/El.dx + 1           // number of test voltage levels
        vsd = new Vector(n,0)                 // For storing each voltage step duration
        for i=0, n-1 vsd.x[i] = El.x1 + El.dx*i
        El.varydur(1)                         // Varies the voltage step duration and calls run()
repeatedly
}

proc plotxy() {                               // Plot vector $o1 against vector $o2 in graph $o3
with color $4, used in El.oc
                                              // Color: 0 white 1 black 2 red 3 blue 4 green 5
orange 6 brown 7 violet 8 yellow 9 gray
        $o2.line($o3, $o1, $4, 1)
        print "Vector plotted"                // for debugging
}

objref vsd_shifted
proc plottail() {                     // Plot tail currents on graph $o1 with mark $s2 ("S" is
square, "T" is triangle, "O" is circle)
        El.vci_tail.printf                    // for debugging
        vsd_shifted = vsd.c.add(trans)
        El.vci_tail.mark($o1, vsd_shifted, $s2, 6)
}

El.vc.dur[0] = trans
El.vc.dur[1] = 100
El.vc.dur[2] = 1000
El.vc.amp[0] = -115
El.vc.amp[1] = -30
El.vc.amp[2] = -115

El.vc.rs = 5                  // series resistance

El.map()                      // This makes sure that vamp[3] & vdur[3] are updated to
user-specified values
```

```
// Voltage clamp step duration range
x1 = 2
x2 = 40
//dx = 38                                  // for debugging
dx = 2

// Prepare graphs
addgraph("El.vc.i", tstart - 25, tstart + 75, -1.5, 0.1, 761, 102, 300.48, 200.32)
addgraph("soma.v(0.5)", tstart - 25, tstart + 75, -120, 20, 761, 365, 300.48, 200.32)
addgraph2(tstart - 25, tstart + 75, -1.5, 0.1, 1080, 102, 300.48, 200.32)
objref gvci
gvci = g[last]
addgraph2(tstart - 10, tstart + x2, -0.5, 0.05, 1399, 102, 300.48, 200.32)
load_file("rig.ses")

// Simulate all conditions
simulate(x1, x2, dx)
plottail(g[last], "S")
g[last-1].addvar("El.vc.i",1,0)
g[last].addvar("El.vc.i",1,0)
El.installVclamp()
run()
if (stoppedrun()) {
        break
}
```






Graph[2] x 965 : 1085 y -8.81 : 0.910
Close
Hide
El.vc.i
Graph[3] x 985 : 1045 y -3.86 : 0.46
Close
Hide
El.vc.i



- Tail current plots do not match. Wrong definition?

Plan for next week

1. Continue to reproduce figures in Destexhe et al 1998a
2. Read and write down notes for Traub et al 2005
3. Examine the codes for the Destexhe et al 1998, 2001 model
4. Examine the codes for the Traub et al 2005 model
5. Reproduce figures in Destexhe et al 1998b
6. Reproduce figures in Destexhe 1998
7. Reproduce figures in Destexhe et al 2001
8. Reproduce figures in Traub et al 2005
9. Compile relevant papers that have cited Destexhe et al 1996 and/or have used the Destexhe et al 1996 model
10. Compile relevant papers that have cited Destexhe et al 1998b and/or Destexhe 1998 and/or Destexhe et al 2001 and/or have used the Destexhe et al 1998, 2001 model
11. Compile relevant papers that have cited Traub et al 2005 and/or have used the Traub et al 2005 model
12. Optional? Examine the codes for the Destexhe et al 1996 model

Plan for the future

1. Compile relevant papers about absence epilepsy
2. Read and write down notes for Chen et al 2014 and Chen et al 2015
3. Read and write down notes for Zhao et al 2015
4. Examine the codes for the Chen et al 2014, 2015 model
5. Examine the codes for the Zhao et al 2015 model
6. Examine the codes for the Destexhe et al 1996 model
7. Go through the NEURON Hands-On Course
8. “Reproduce CSD graph” exercise
9. Examine Christine’s & Mark’s codes
10. Finish NEURON Book Appendix A1
11. Figure out how to export NEURON to Matlab
12. Complete the **NEURON Tutorial**
13. Understand NEURON Ch 7 & Ch 8
14. Resolve all NEURON Book questions
15. Read Abbott et al 2016 (“Building functional networks of spiking model neurons”)
16. Read Markram et al 2015 (“Reconstruction and Simulation of Neocortical Microcircuitry”)
17. Read Kragel & LaBar 2016 (“Decoding the Nature of Emotion in the Brain”)
18. Read Izhikevich: Dynamical Systems in Neuroscience
19. Read Dayan & Abbott: Theoretical Neuroscience
20. Derivation & shape of the **Goldman–Hodgkin–Katz flux equation**

5/20/2016~5/22/2016

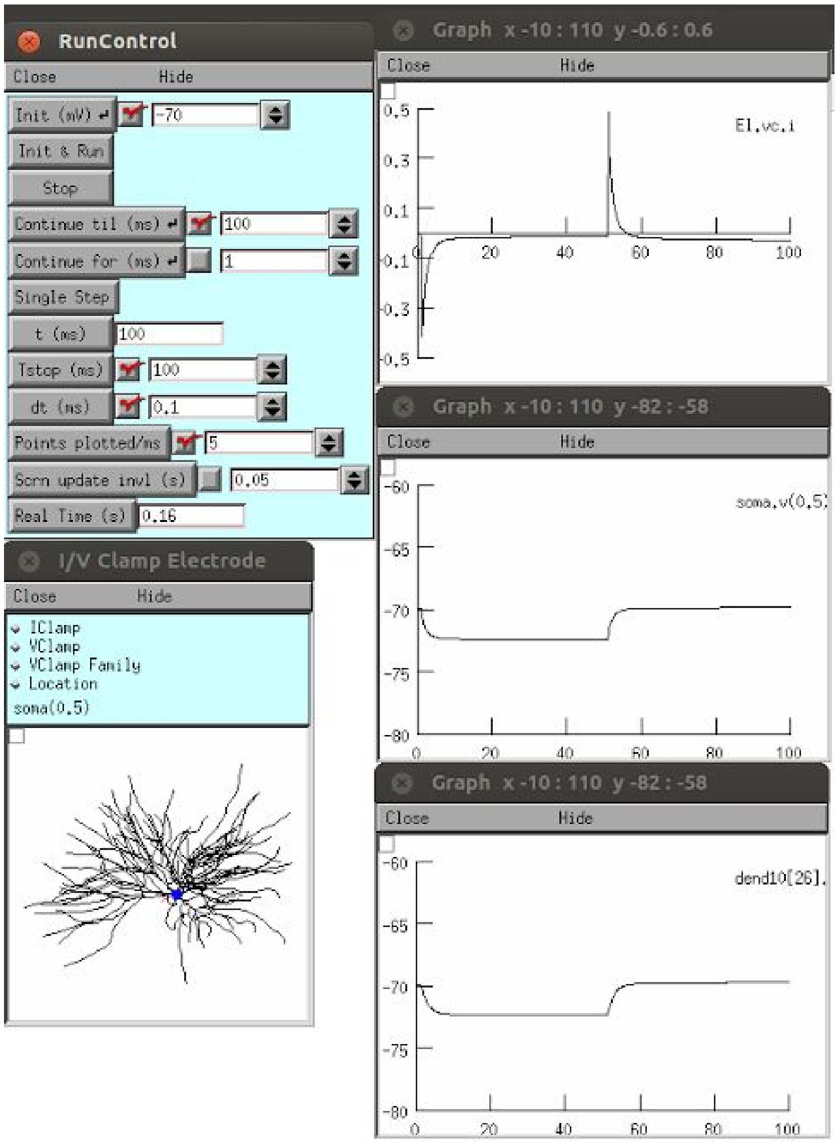
Reproduce Figures in Destexhe et al 1998a (part 1)

- **Figure 1C**
  - Voltage clamp of the detailed cell model thats fits to experimental data
  - Modified code of **tc200_vc.oc**:

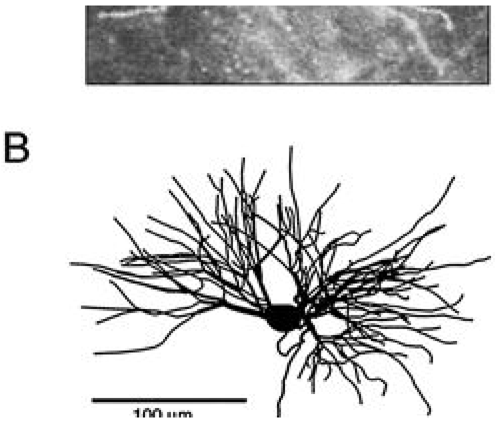
trans = 0
npoints = 500

Name	Description	New value	Notes
vc.dur[0]	Duration of voltage clamp level 1 [ms]	**1**	
vc.dur[1]	Duration of voltage clamp level 2 [ms]	**50**	
vc.dur[2]	Duration of voltage clamp level 3 [ms]	**50**	
vc.amp[0]	Amplitude of voltage clamp level 1 [mV]	**-70**	
vc.amp[1]	Amplitude of voltage clamp level 2 [mV]	**-72.5**	
vc.amp[2]	Amplitude of voltage clamp level 3 [mV]	**-70**	

addgraph("El.vc.i",**-0.5,0.5**)
addgraph("soma.v(0.5)",**-80,-60**)
addgraph("dend10[26].v(0.5)",**-80,-60**)



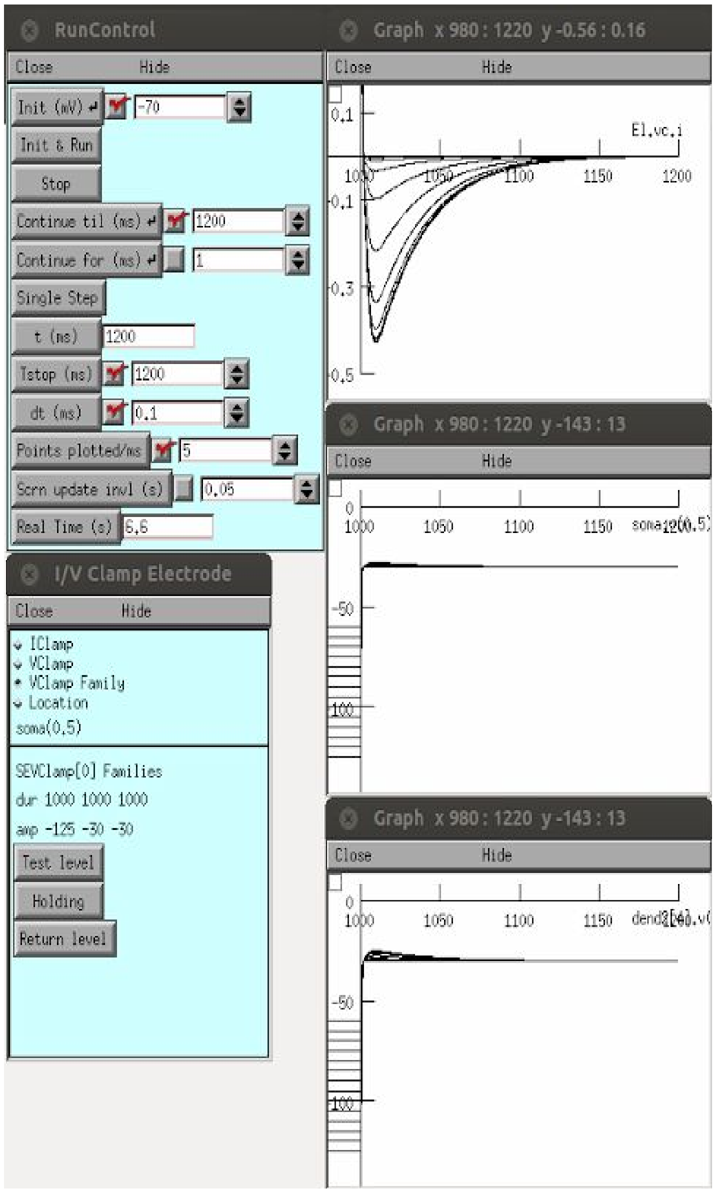
Init (mV) -70
Continue til (ms) 100
Continue for (ms) 1
t (ms) 100
Tstop (ms) 100
dt (ms) 0.1
Points plotted/ms 5
Scrn update invl (s) 0.05
Real Time (s) 0.16
soma(0.5)

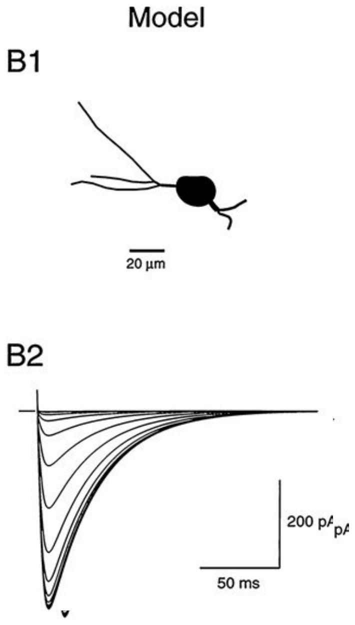


- Did not reproduce offset

- **Figure 2B**
  - Voltage clamp of the dissociated cell model thats fits to experimental data
  - Modified code of **El.oc**, in **varyamp()**:
    if (i == 0) {
    x1 = -125 x2 = -60   dx = 5
    This ramps **vc.amp[0]** = **-125**:**5**:**-60** when “**Holding**” is clicked under **VClamp Family**
  - Modified code of **tcD_vc.oc**:
    In **addgraph()**:
    g[ii].**family(1)**   // turn Keep Lines on
    El.vc.amp[0] = **-125**
    El.**map()**   // This makes sure that vamp[3] & vdur[3]

```
                                          // are updated to user-specified values in
                                          // hoc
addgraph("El.vc.i",-0.5,0.1)              // current
addgraph("soma.v(0.5)",-130,0)            // soma voltage
addgraph("dend2[4].v(0.5)",-130,0)        // dendrite voltage
```


```
RunControl
Init (mV)            -70
Init & Run
Stop
Continue til (ms)    1200
Continue for (ms)    1
Single Step
t (ms)               1200
Tstop (ms)           1200
dt (ms)              0.1
Points plotted/ms    5
Scrn update invl (s) 0.05
Real Time (s)        6.6

I/V Clamp Electrode
IClamp
VClamp
VClamp Family
Location
soma(0.5)
SEVClamp[0] Families
dur 1000 1000 1000
amp -125 -30 -30
Test level
Holding
Return level

Graph x 980 : 1220 y -0.56 : 0.16
Graph x 980 : 1220 y -143 : 13
Graph x 980 : 1220 y -143 : 13
```



- **Figure 3 & 4**
    - Voltage clamp data of the intact TC cell (**Figure 3**)
    - Voltage clamp of the detailed cell model thats fits to experimental data (**Figure 4**)
    - Modified code of **El.oc**, in **varyamp()**:

```
if (i == 0) {
        x1 = -105  x2 = -40     dx = 5
```

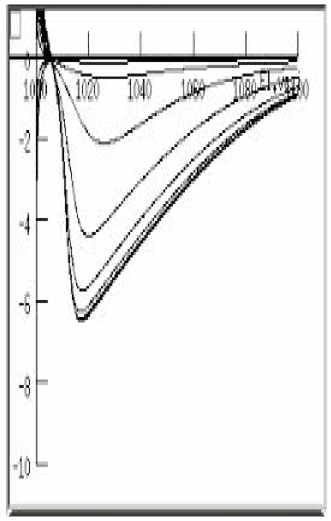
This ramps **vc.amp[0]** = **-105**:**5**:**-40** when “**Holding**” is clicked under **VClamp Family**

- Modified code of **tc200_vc.oc**:
  In **addgraph()**:

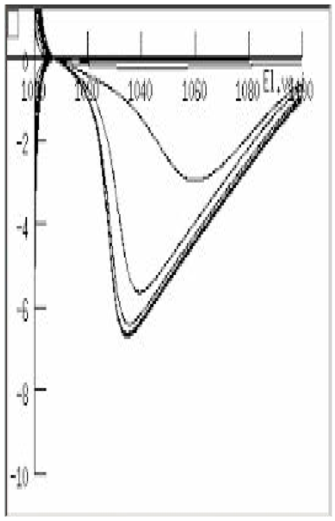
```
        g[ii].family(1)              // turn Keep Lines on
// No T-current in distal dendrites (Fig 4A)
localize(1.7e-5,corrD*0,corrD*0)
// Uniform T-current throughout the neuron (Fig 4B)
localize(1.7e-5,corrD*1.7e-5,corrD*1.7e-5)
// Density of T-current in distal dendrites is twice of that in the perisomatic area
(Fig 4C)
localize(1.7e-5,corrD*3.4e-5,corrD*3.4e-5)
// Density of T-current in distal dendrites is 5 times of that in the perisomatic area
(Fig 4D and all mimics of Figure 3)
localize(1.7e-5,corrD*8.5e-5,corrD*8.5e-5)
El.vc.amp[0] = -105
El.vc.amp[1] = -55 or -65 or -60 or -45
El.vc.amp[2] = -55 or -65 or -60 or -45
                                     // To mimic Fig 3A, 3B, 3C, 3D, respectively
El.map()                             // This makes sure that vamp[3] & vdur[3]
                                     // are updated to user-specified values in
                                     // hoc
npoints = 500                        // for Figure 3 mimics
npoints = 1000                       // for Figure 4
addgraph("El.vc.i",-10,0.1)          // For Figure 3 mimics
addgraph("El.vc.i",-6,0.1)           // For Figure 4
addgraph("soma.v(0.5)",-110,0)       // soma voltage
addgraph("dend2[4].v(0.5)",-110,0)   // dendrite voltage
```

Figure 3

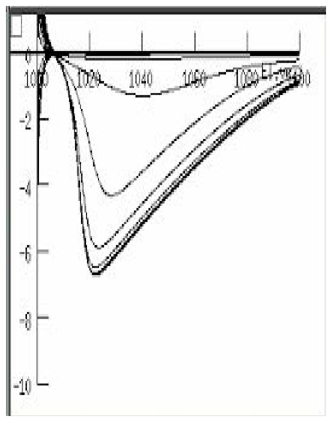
El.vc.amp[1] = **-55**



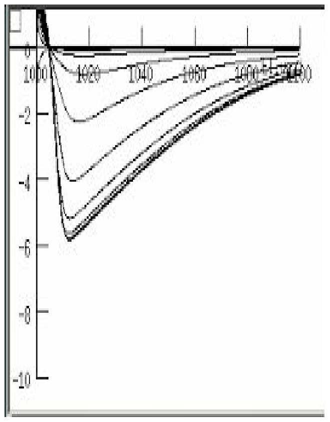
El.vc.amp[1] = **-65**



El.vc.amp[1] = **-60**



El.vc.amp[1] = **-45**



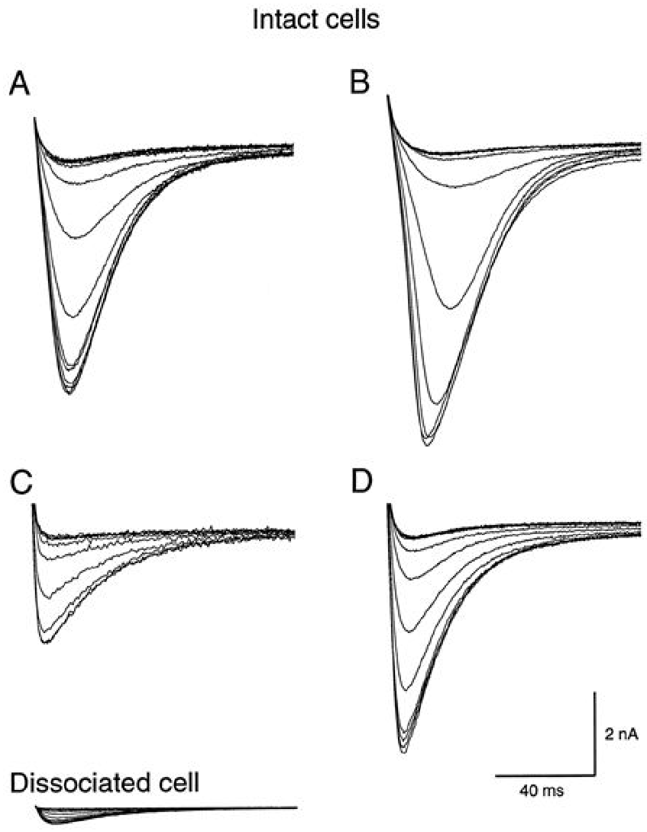
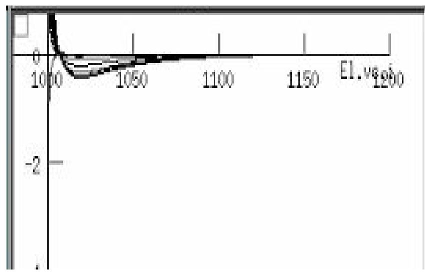
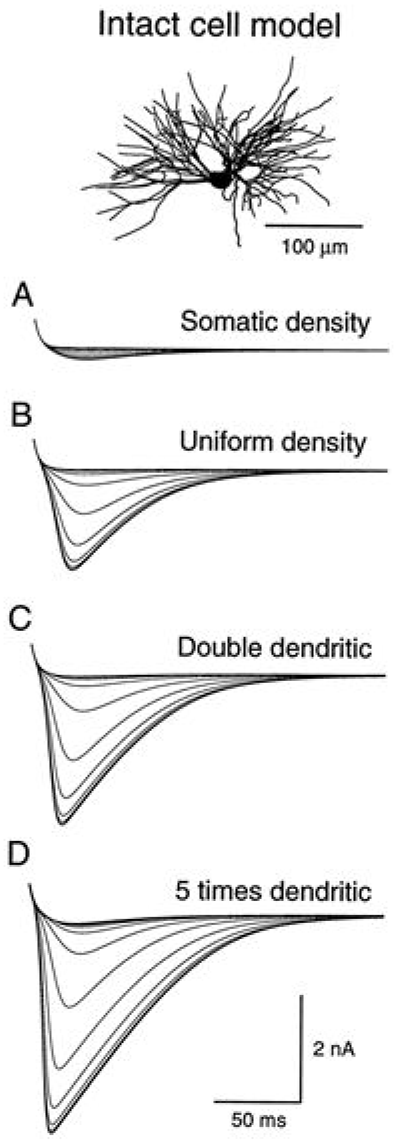


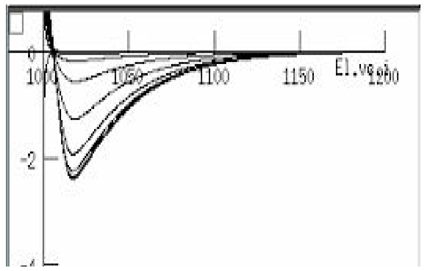
Figure 4

No T-current in distal dendrites (only in **soma & proximal dendrites**)

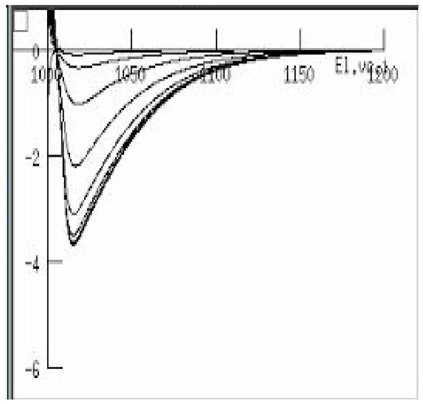




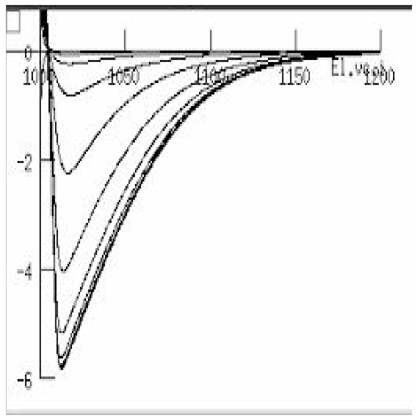
Uniform T-current throughout the neuron



Density of T-current in distal dendrites is **twice** of that in the perisomatic area



Density of T-current in distal dendrites is **5 times** of that in the perisomatic area



- **Figures 5C & 5D**
    - Voltage clamp data of the intact TC cell (**Figure 3**)
    - Voltage clamp of the detailed cell model thats fits to experimental data (**Figure 4**)
    - Modified code of **El.oc**
        - Added public declaration:
          ```
          public varyamp, vci, vci_peak, x1, x2, dx
          ```
        - In **init()**:
          ```
                  x1 = -100
                  x2 = -50
                  dx = 5
          ```
        - **varyamp()** is modified to:
          ```
          objref vci, vci_peak
          proc varyamp() {local i, x, old, n, min
                  i = $1
                  set_vclamp()
                  n = (x2-x1)/dx + 1              // number of voltage levels
                  vci_peak = new Vector(n,0)      // For storing peak currents for each
                                                  // voltage level
                  for (x = x1; x <= x2; x = x + dx) {
                          vci = new Vector()      // For recordings of vc.i
                          vci.record(&vc.i)       // Records vc.i
                          vc.amp[i] = x
                          run()
                          if (stoppedrun()) {
                                  break
                          }
                          vci_peak.x[(x-x1)/dx] = vci.min(10000,11000)
                  }
          }
          ```
    - Modified code of **tc200_vc.oc**:
        - Modified **addgraph()**:
          ```
          g[ii] = new Graph(0)
          g[ii].view($2, $4, $3-$2, $5-$4, 1228, 120 + ii*263, 300, 200)
          g[ii].family(1)
          ```
        - Voltage clamp defaults:
          El.vc.dur[1] = **100**
          El.vc.dur[2] = **0**
          hold = **-105** or **-115**
          El.vc.amp[0] = hold
          El.vc.amp[1] = hold

```
EI.vc.amp[2] = hold
EI.map()
```

- Modified simulation parameters:

```
npoints = 500
```

- Add graphs:

```
addgraph("EI.vc.i",tstart,tstop,-3,0.1)
addgraph("soma.v(0.5)",tstart,tstop,-120,20)
// Create graph of peak current over test voltage level
ngraph = ngraph+1
last = ngraph-1
g[last] = new Graph(0)
g[last].view(-100, -3, 100, 3, 1228, 120 + last*263, 300, 200)
g[last].xaxis()
g[last].yaxis()
g[last].family(1)                              // turn Keep Lines on
g[last].save_name("graphList[0].")
graphList[0].append(g[last])
load_file("rig.ses")
```

- Specific for this figure:

```
objref testvoltagelevel
proc simulate() {                              // Test voltage level runs
from
                                               // $1 to $2 with interval $3
        EI.x1 = $1
        EI.x2 = $2
        EI.dx = $3
        n = (EI.x2-EI.x1)/EI.dx + 1            // number of test voltage
                                               // levels
        testvoltagelevel = new Vector(n,0)     // For storing each test
                                               // voltage level
        for i=0, n-1 testvoltagelevel.x[i] = EI.x1 + EI.dx*i
        EI.varyamp(1)                          // Varies the test voltage level
                                               // and calls run() repeatedly
}

proc plotIV() {                                // Plot I-V curve
        EI.vci_peak.printf                     // for debugging
        EI.vci_peak.line(g[last], testvoltagelevel)
        EI.vci_peak.mark(g[last], testvoltagelevel, "O", 6)
}

// Uniform T-current throughout the neuron (Fig 5D, upper curve)
localize(1.7e-5,corrD*1.7e-5,corrD*1.7e-5)
```

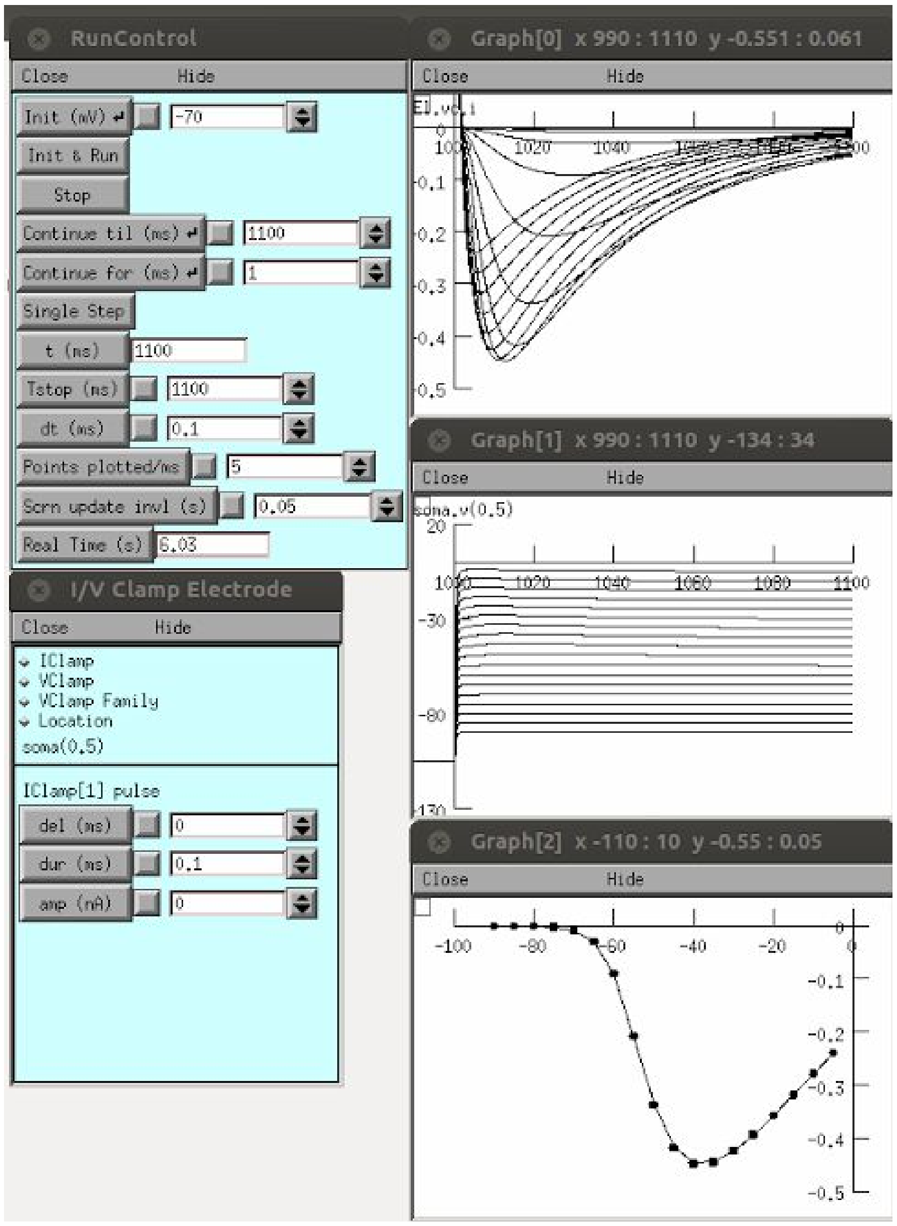
simulate(**-80**, **-20**, **5**)
plotIV()

// Erase first two graphs
g[0].erase()
g[1].erase()

// Density of T-current in distal dendrites is higher than that in the
// perisomatic area (Fig 5D, lower curve)
localize(1.7e-5,corrD***2.5e-5**,corrD***2.5e-5**)
simulate(**-80**, **-35**, **5**)
plotIV()

- Modified code of **tcD_vc.oc**:
  - Similar as the modifications to tc200_vc.oc, except:
    addgraph("El.vc.i",tstart,tstop,-0.5,0.01)
    g[last].view(-100, **-0.5**, 100, **0.5**, 1228, 120 + last*263, 300, 200)
    simulate(**-90**, **-5**, **5**)

Figure 5C (Holding level **-105 mV**)



RunControl

Init (mV) -70

Init & Run

Stop

Continue til (ms) 1100

Continue for (ms) 1

Single Step

t (ms) 1100

Tstop (ms) 1100

dt (ms) 0.1

Points plotted/ms 5

Scrn update invl (s) 0.05

Real Time (s) 6.03

I/V Clamp Electrode

IClamp

VClamp

VClamp Family

Location

soma(0.5)

IClamp[1] pulse

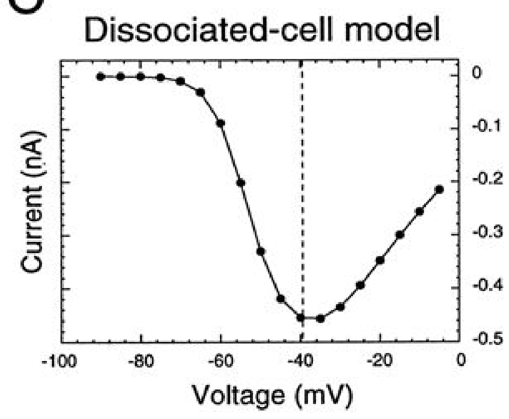
del (ms) 0

dur (ms) 0.1

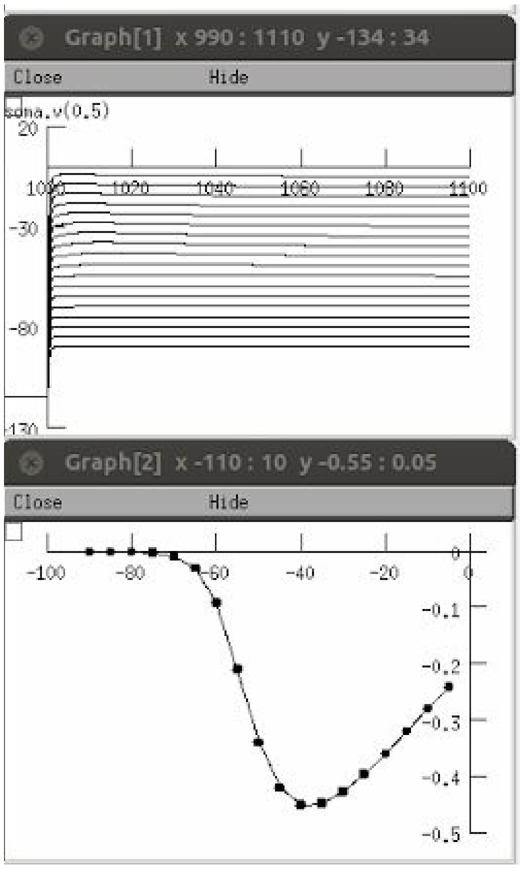
amp (nA) 0

Graph[0] x 990 : 1110 y -0.551 : 0.061
El.vc.i
Graph[1] x 990 : 1110 y -134 : 34
soma.v(0.5)
Graph[2] x -110 : 10 y -0.55 : 0.05
Close
Hide

C



(Holding level **-115 mV**)



C

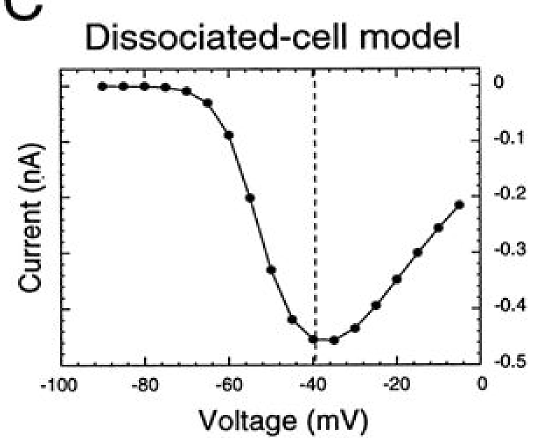
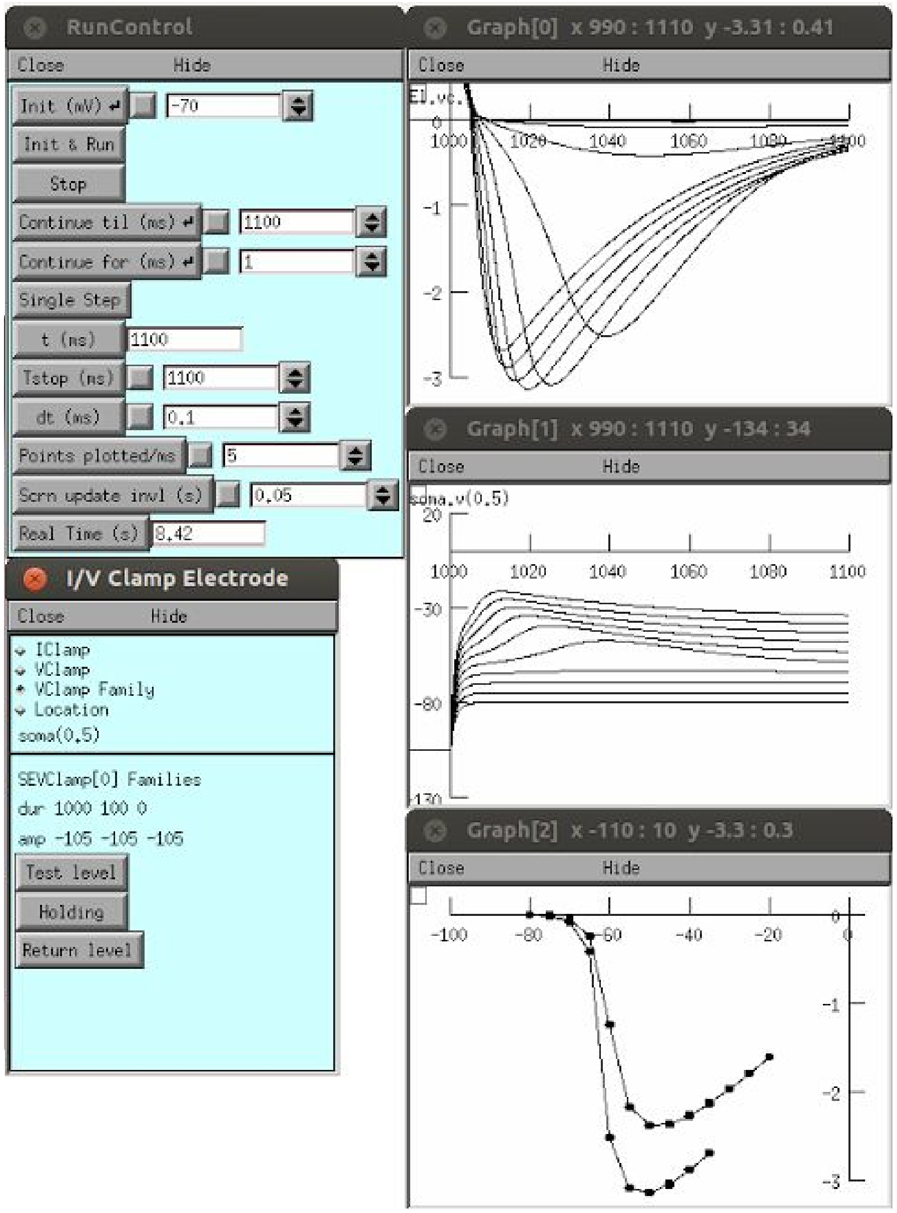
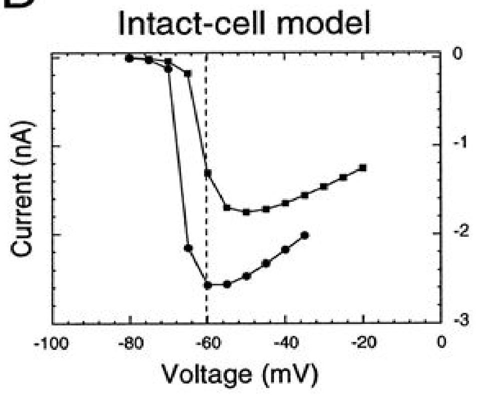


Figure 5D (Holding level **-105 mV**)
Top curve: Uniform T-current density
Lower curve: Higher T-current density distally (**2.5e-5** cm/sec vs **1.7e-5** cm/sec)

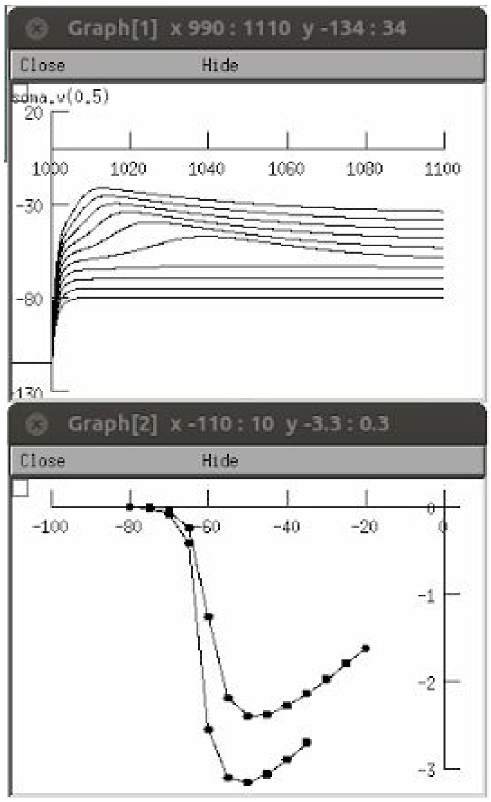


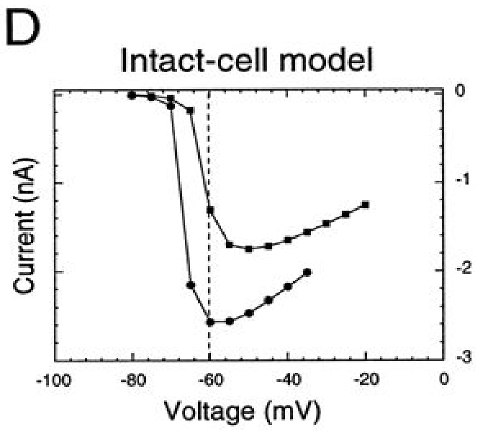
```
RunControl
Close    Hide
Init (mV)            -70
Init & Run
Stop
Continue til (ms)    1100
Continue for (ms)    1
Single Step
t (ms)               1100
Tstop (ms)           1100
dt (ms)              0.1
Points plotted/ms    5
Scrn update invl (s) 0.05
Real Time (s)        8.42

I/V Clamp Electrode
Close    Hide
IClamp
VClamp
VClamp Family
Location
soma(0.5)

SEVClamp[0] Families
dur 1000 100 0
amp -105 -105 -105
Test level
Holding
Return level

Graph[0] x 990 : 1110 y -3.31 : 0.41
Graph[1] x 990 : 1110 y -134 : 34
Graph[2] x -110 : 10 y -3.3 : 0.3
```

D



(Holding level **-115 mV**)





Not exactly the same

5/13/2016~5/22/2016

Notes from Destexhe et al 1998b

- Background Facts:
    1. **Decortication** => sleep spindle oscillations still exist.
    2. *In vivo*: the **rostral pole of the reticular thalamus (RE)**, deafferented, generates spindle rhythms by itself (no oscillations occur when the RE nucleus is isolated *in vitro* though).
    3. *In vitro*: Ferret **lateral geniculate** slices shows spindle oscillations that behave as traveling waves (progressive recruitment).
    4. Spindles recorded in **distant sites** in the cortex or thalamus *in vivo* are "**nearly simultaneous**," whereas those recorded in distant sites in the thalamic slices *in vitro* are rarely simultaneous (but shows **systematic propagation**).
    5. **Deep cortical incisions** does not affect synchronization. However, **sectioning the corpus callosum** reduces synchronization.
    6. In thalamic slices, many thalamocortical (TC) cells are observed to be **spontaneous oscillators**.
    7. **Thalamic refractoriness**: Caged Ca2+ experiments show that intracellular Ca2+ enhances I_h currents in TC cells, which reduces the tendency of TC cells to display rebound bursts.
    8. Anatomical facts:
        a. Ascending thalamocortical fibers project mostly to layers **I**, **IV** and **VI** of the cortex.
        b. All corticothalamic fibers originating from layer **VI** leave **axon collaterals** in **RE**.
        c. In area 5 of cat cerebral cortex, axon collaterals from pyramidal cells are profuse and dense but remain localized within a **few hundreds of microns**.
        d. Intra-thalamic projections & projections between the thalamus and cortex are both **local** and **topographic**. However, the latter has **diverge** more.
        e. Some corticothalamic fibers originates from lower layer **V** and *does not* leave axon collaterals in RE (not modeled).
        f. Some **intralaminar nuclei** project diffusely to the cortex and receive projections from it (not modeled).
        g. **PY→TC** synapses end on the most **distal** part of the dendritic tree, whereas **RT→TC** inhibitory synapses end more **proximally**.
        h. A large part of synaptic terminals in the thalamus arise from brainstem structures instead of the cortex
        i. Divergence: ascending TC axons to the somatosensory cortex extend to **~600 µm**, with neighboring TC cells projecting up to **1500 µm**.
    9. Electrophysiological evidence:
        a. There is evidence that thalamic interneurons do not play a role in the generation of spindles.

   b. PY cells have a **fluctuating voltage trace** with occasional **spontaneous firing**.
   c. Intracellular recordings of **RE** cells show a high sensitivity to **EPSPs** of cortical or internal capsular origin.
   d. Intracellular recordings of **TC** relay neurons show a **dominance of inhibition**, but is transformed to a powerful EPSP after lesioning the RE nucleus. This is true also with other anesthetics such as **ketamine-xylazine**.
   e. Stimulating the cortex (even contralaterally) evokes spindle oscillations.
   f. During spontaneous oscillations, TC cells are entrained by an initial **IPSP**.
   g. During spindle oscillations, anatomically related cortical and thalamic territories discharged **in phase**.
   h. During cortical seizures, **60%** of TC cells are **hyperpolarized**, while cortical neurons produce paroxysmal discharges.
   i. RE cells may have a powerful **dendritic T-current**.
   j. IPSPs are seen in TC cells even with low-intensity cortical stimuli

- Hypothesis: The discrepancies between in vivo and in vitro studies can be explained by the presence or lack of a **corticothalamic feedback** that acts predominantly on exciting RE cells and therefore recruiting TC cells via **dominant inhibition**.
- Past Models:
  1. Detailed biophysical models have already explained fact #2
- Current Model:
  1. **Single compartments** for 4 types of cells:
     cortical pyramidal cells (PY)
     cortical interneurons (IN)
     reticular thalamic cells (RE)
     thalamocortical cells (TC)
  2. Membrane equation:

$$C_m \frac{dV_i}{dt} = -g_L(V_i - E_L) - \sum_j I^{int}{}_{ji} - \sum_k I^{syn}{}_{ki}$$

     where V_i is the membrane potential of cell i,
     C_m = 1 μF/cm² is the specific membrane capacitance,
     g_L & E_L are the conductance and reversal potential of the **leak current**
     I^int_ji and I^syn_ki are the **intrinsic** and **synaptic currents**, respectively, going to cell i
  3. Intrinsic currents:
     a. Membrane equation:

$$I^{int}{}_{ji} = \overline{g}_j m_j{}^M h_j{}^N (V_i - E_j)$$

     PY: I_Na, I_K, **I_M** (**slow voltage-dependant K+ current**, contributes to adapting trains of action potentials)
     IN: I_Na, I_K

RE: I_Na, I_K, **I_T** (contributes to bursts)
TC: I_Na, I_K, **I_T** (contributes to bursts), **I_h** (contributes to waxing-and-waning properties of oscillations)

b. I_h is regulated by **intracellular Ca2+** using a kinetic model involving **Ca2+-binding proteins**.
c. Activation and inactivation gates follow:

$$\text{(closed)} \underset{\beta(V)}{\overset{\alpha(V)}{\rightleftarrows}} \text{(open)}$$

d. Parameter values were obtained by fitting to voltage-clamp data:

Parameter	Value
Cortical pyramidal cells (PY)	
Membrane Area	29,000 μm2
ḡL	0.1 mS/cm2
EL	−70 mV
ḡNa	50 mS/cm2
ḡK	5 mS/cm2
ḡM	0.07 mS/cm2
Cortical interneurons (IN)	
Membrane Area	14,000 μm2
ḡL	0.15 mS/cm2
EL	−70 mV
ḡNa	50 mS/cm2
ḡK	10 mS/cm2
Thalamic reticular cells (RE)	
Membrane Area	14,000 μm2
ḡL	0.05 mS/cm2
EL	−90 mV
ḡNa	200 mS/cm2
ḡK	20 mS/cm2
ḡTs	3 mS/cm2
Thalamocortical cells (TC)	

Membrane Area	29,000 µm2
ḡL	0.01 mS/cm2
EL	−70 mV
ḡKL	3–5* nS
ḡNa	90 mS/cm2
ḡK	10 mS/cm2
ḡT	2 mS/cm2
ḡh	0.015–0.02* mS/cm2

***Randomization** of **I_h** and **I_KL** conductances models **spontaneous oscillators** of TC cells (Background Fact **#7**)

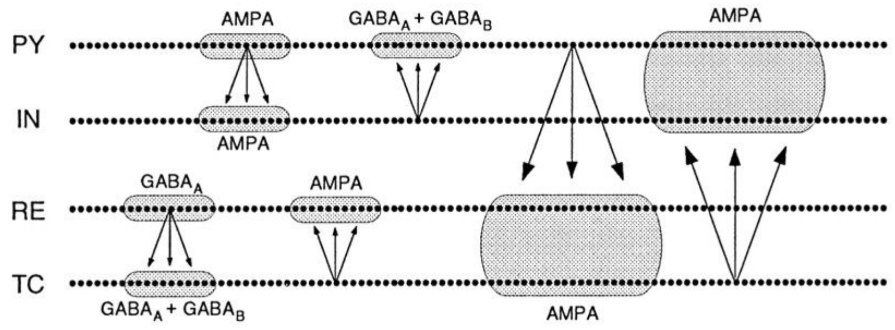
4. Synaptic currents:
   a. Membrane equation:

$$I^{syn}_{ki} = \overline{g_{ki}} m_{ki}(V_i - E_{ki})$$

   b. m_ki is the fraction of open receptors according to the kinetic scheme:

$$\text{(closed)} + T(V\_k) \underset{\beta(V)}{\overset{\alpha(V)}{\rightleftarrows}} \text{(open)}$$

   c. When a spike occurs, T(V_k) is set to **0.5 mM** for **0.3 ms**
   d. See Destexhe et al 1996 for equations of α(V) and β(V)
   e. A decay time constant of **12.5 ms** was used for GABA_A
5. Network structure and synaptic connections:
   a. Based on Background Facts **#8a~e**
   b. 4 one-dimensional layers, **100** cells each:
      Layer VI cortical pyramidal cells (**PY**)
      cortical interneurons (**IN**)
      reticular thalamic cells (**RE**)
      thalamocortical cells (**TC**)
   c. The synaptic connections are modeled as:



where each small box consists of **11** cells & each large box consists of **21** cells.

d. For a given type of cell, all axonal projections and synaptic conductances are equal.

e. **Reflexive boundary conditions** (see Destexhe et al 1996) were used to minimize boundary effects.

f. In some simulations, every PY cell had an extra **20** AMPA & **20** GABA_A synapses, with conductance values of **0.01 μS** and **0.0025 μS**, respectively. These synapses are randomly activated according to a **Poisson process** with a mean rate of **15 Hz**, producing voltage traces that explain Background Fact **#9b**.

g. Synaptic conductances used:

Type of Receptor	Location	Optimal Conductance Value	Range Tested
AMPA	PY → PY	0.6 μS	0–0.9 μS
AMPA	PY → IN	0.2 μS	0.1–0.4 μS
GABAA	**IN → PY**	0.15 μS	0.09–0.2 μS
GABAB	IN → PY	0.03 μS	0–0.2 μS
AMPA	TC → RE	0.2 μS	0.1–1 μS
GABAA	RE → RE	0.2 μS	0.05–0.4 μS
GABAA	RE → TC	0.02 μS	0.01–0.04 μS
GABAB	RE → TC	0.04 μS	0–0.15 μS
AMPA	TC → PY	1.2 μS	0.4–2.5 μS
AMPA	TC → IN	0.4 μS	0.1–0.6 μS
AMPA	**PY → RE**	1.2 μS	0.4–2 μS
AMPA	**PY → TC**	0.01 μS	0–0.07μS

- Experimental Methods:
  1. Adult cats anesthetized with **pentobarbital** (**35 mg/kg**)
  2. Intracellular recordings
     a. Recording electrode: **TC** cells of the **lateral posterior** nucleus
     b. Stimulating electrode: Bipolar electrodes in the depth of the **suprasylvian cortex**
     c. Internal solution: 3 M potassium acetate, final DC resistance: **30-40 MΩ**
  3. Field recordings
     a. In the **suprasylvian gyrus**
     b. Details in Contreras et al 1997a
  4. Spatial coherence analysis:

   a. Details in Contreras et al 1996a

- Results
    1. **Corticothalamic feedback** creates **dominant inhibition** on TC cells by **exciting RE cells**
        a. Intracellular recordings of **TC** cells show an EPSP-IPSP sequence dominated by the **IPSP** component. This is in contrast to that of RE (Background Fact **#9c**) and corroborates Background Fact **#9d~f**.
        b. In a small circuit of **2** TC & **2** RE cells, EPSPs on RE and TC cells could reproduce Result **#1a** as long as the EPSPs on RE cells were **stronger** than those on TC cells.
    2. Dominance inhibition is optimal for triggering **thalamic oscillations**
        a. In a simplified thalamocortical model of **2** cells each, **9-11 Hz** spindles could be generated either spontaneously (triggered by a rebound burst of a **TC** neuron) or from stimulating a **PY** neuron. Within one cycle, all cell types discharged in phase, in agreement with Background Fact **#9g**.
        b. When the strength of PY→TC is weak, IPSPs dominate and spindles can occur. With increasing PY→TC strength, no oscillations could be evoked. When PY→TC is very strong, cortical discharges can evoke spikes in TC cells directly.
        c. All IPSPs were dominated by **GABA_A** receptors.
        d. TC cells burst **once every two cycles**, similar as in purely thalamic circuits.
        e. The model was extremely **robust** to changes in conductance parameters within the range indicated in the table above. Testing condition: spindle oscillations could be generated by both **intrinsic oscillatory behavior of TC** cells and **cortical stimulation**
        f. **Local average** membrane potentials over **21** adjacent PY cells show that spindle oscillations began approximately simultaneously (within **~0.2 s**) in several (**1-3**) different sites, then merges into a unique synchronized oscillation
    3. Dominance inhibition determines **thalamic coherence**
        a. Individual thalamic cells as well as local average potentials were considerably **more simultaneous** in the presence of cortical feedback.
        b. Without cortex, initiation sites led to **local patterns of propagation** and **colliding waves**, agreeing with Background Fact **#4**.
        c. Power spectrum analysis: **7-15 Hz power** increases concomitantly in distant sites only if cortex is present.
        d. **Spatial correlations** from thalamic cells decays more with distance when cortical feedback was removed.
    4. The cortex can trigger thalamic oscillations only after some period of silence (**refractoriness**)
        a. Oscillations could be evoked by cortical stimulation only if preceded by a **~2-8 s** period of silence

        b. For a moderate stimulus intensity, a periodic stimuli of every **4 s** entrained the network once every **2** stimuli, indicating a **refractory period** of **8~12 s**.
    5. Patterns of **systemic propagations** can be generated at the end of the refractory period by exciting a localized area of the thalamus
        a. Due to **reciprocal corticothalamic connectivity** and not to horizaontal intracortical connections.
        b. Synchronization after **high-intensity** cortical stimulation is due to the activation of a more extended population of cortical cells at once
    6. If cortical PY cells were subject to **random synaptic bombardment**, **spontaneous spindles** could occur
        a. Generalized spindles recur with a period of **4-10 s**, with great variability.
        b. Occasionally, local spindles occur when spontaneous cortical discharges occurred before complete recovery from refractoriness.
- Discussion
    1. The hypothesis is true. The **divergence** of projections and thalamic **refractoriness** cause the corticothalamic loops to synchronize the entire network and generate spindle oscillations.
    2. Future directions:
        a. The model predicts that local injection of GABA_A blockers, but not AMPA blockers, in thalamic relay nuclei should lead to dramatic changes in the large-scale coherence of oscillations.
        b. **Long-range synchrony** of slow oscillations in **deep sleep** versus **local synchrony** of fast oscillations (20-60 Hz) during **activated periods**. Possibly, **cholinergic synapses** depolarize TC cells but hyperpolarize RE cells, switching between two different types of thalamic responsiveness.

5/18/2016~5/22/2016

Notes from Destexhe 1998

- Background Facts:
    1. Role of **thalamus** in absence seizures:
        a. Field recordings in humans: During absence attacks, **thalamic recordings** in humans show **spike-and-wave (SW)** patterns.
        b. Field recordings in animal models: SW patterns disappear after **thalamic lesions** or **inactivation of the thalamus**.
        c. Intracellular recordings in animal models: Cortical and thalamic cells fire prolonged discharges in phase with the EEG "**spike**" component and all cells are silent during the "**wave**" component. Some TC cells stay silent throughout entire oscillation.
        d. Intracellular recordings in animal models: Spindles generated by thalamic circuits can transform gradually into SW discharges.
    2. Role of **GABA_B receptors** in the genesis of SW discharges:
        a. GABA_B **agonists** exacerbate SW discharges.
        b. GABA_B **antagonists** suppress SW discharges.
        c. **Clonazepam** acts on GABA_A receptors but seems to diminish GABA_B-mediated IPSPs in thalamocortical cells.
        d. Ferret thalamic slices:
        GABA_A antagonists => spindles transform into **3 Hz** oscillations
        Then **GABA_B antagonists** => these oscillations are suppressed
    3. Computational models of thalamic circuits:
        a. Can replicate Fact #2d
    4. Role of **cortex** in absence seizures:
        a. Injection of **penicillin** or **bicuculline** (both GABA_A antagonists) in the thalamus led to **3-4 Hz** oscillations but no SW discharge. However, injection in the **cortex** resulted in seizures with **SW discharge**.
        b. In cats treated with penicillin, SW discharges could be transformed back to spindle oscillations if the **cortex is inactivated**.
    5. Other features of absence seizures:
        a. Bursts of several cycles of SW oscillations are interleaved with **long periods of silence** (~20 seconds).
        b. In a rat model of absence epilepsy, **T-current** is increased selectively in **RE** cells.
        c. **Ethosuximide**, which is thought to block **T-currents** selectively in **TC** cells, reduces absence seizures.
        d. Different experimental models of absence seizures generate **different frequencies** of SW bursts.
        e. Some GABA_A agonists, such as **barbiturates**, may increase the frequency of seizures
    6. Anatomical observations:

   a. **Cortical synapses** contact only the distal dendrites of TC cells, but proximal dendrites of RT cells.

- Hypothesis: By the use of a computational model based on the intrinsic firing properties of thalamic & cortical neurons, a **thalamocortical loop** mechanism can explain the genesis of SW oscillations.
- Methods:
  1. Computational model:
     a. Synaptic currents:
        - AMPA, NMDA, GABA_A receptors are modeled by:

$$I_{syn} = \bar{g}_{syn} m (V - E_{syn})$$
$$\frac{dm}{dt} = \alpha [T](1-m) - \beta m$$

          where [T] is the transmitter concentration.
        - In addition, NMDA receptors had a voltage-dependent term corresponding to an extracellular Mg2+ concentration of 2 mM (see Kinetic Models of Synaptic Transmission Chapter 1)
        - Parameters obtained by fitting to postsynaptic currents recorded experimentally:

Receptor	Parameter	Value
AMPA	E_syn [mV]	0
	α [M^{-1}s^{-1}]	0.94e6
	β [s^{-1}]	180
NMDA	E_syn [mV]	0
	α [M^{-1}s^{-1}]	11e4
	β [s^{-1}]	6.6
GABA_A	E_syn [mV]	-80
	α [M^{-1}s^{-1}]	20e6
	β [s^{-1}]	160

        - GABA_B receptors act through a G-protein to open K+ channels, so are modeled by:

$$I_{GABA_B} = \bar{g}_{GABA_B} \frac{s^n}{s^n + K_D} (V - E_K)$$
$$\frac{dr}{dt} = K_1 [T](1-r) - K_2 r$$

$$\frac{ds}{dt} = K_3 r - K_4 s$$

where r is [GABA_B receptors]_activated
s is the normalized [G-protein]_activated

- Parameters obtained by fitting to postsynaptic currents recorded experimentally:

Receptor	Parameter	Value
GABA_B	K_D [1]	100
	K_1 [$M^{-1}s^{-1}$]	9e4
	K_2 [s^{-1}]	1.2
	K_3 [s^{-1}]	180
	K_4 [s^{-1}]	34
	n	4

b. Field potentials from a single cell:
- Calculated from a single cell receiving 200 synapses (100 excitatory with AMPA & NMDA; 100 inhibitory with GABA_A & GABA_B).
- A **random jitter** of ±1 msec was included in the timing of each presynaptic action potential.
- Equations from the model of Nunez (1981):

$$V_{ext} = \frac{R_e}{4\pi} \sum_j \frac{I_j}{r_j}$$

where R_e = 230 Ω·cm is the extracellular resistivity,
I_j is the postsynaptic current,
r_j is the distance to the postsynaptic current

c. Intrinsic currents: HH type equations (same as Destexhe et al 1998b)

d. Thalamocortical network: Same model & parameter values as Destexhe et al 1998b

e. Field potentials in the network:

$$V_{ext} = \frac{R_e}{4\pi} \sum_{i,k} \frac{{I_{syn}}^{ki}}{r_i}$$

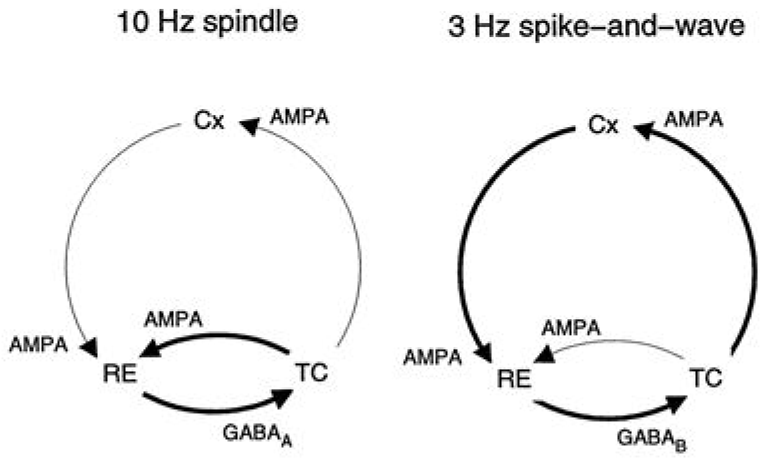
$$\text{or } V_{ext} = \frac{R_e}{4\pi} \sum_i \frac{{I_M}^i + \sum_k {I_{syn}}^{ki}}{r_i}$$

where I_syn^ki is the current from the ith cell, kth synapse
r_i is the distance to the ith cell

- Results:
  1. **GABA_B** receptors are **nonlinearly activated**:
     a. The model assumes that cooperative binding of **4** G-proteins are needed to activate the potassium channel.
     b. A burst of **5-10 high-frequency spikes**, but not an isolated presynaptic spike, can evoke a GABA_B current.
  2. **Spike-and-wave field potentials** can be generated from a single cell:
     a. See Method 1b for details.
     b. Presynaptic trains of single spikes resulted in a field potential of **negative deflections**, whereas bursts of high-frequency spikes resulted in a field potential of **spike-and-wave patterns**.
     c. **Spikes** are more pronounced when excitatory synapses **discharged earlier** than inhibitory synapses.
     d. 90% reduction of AMPA & NMDA receptors, GABA_A receptors and GABA_B receptors diminished the **negative peak**, the **positive peak** and the **wave**, respectively.
  3. In a thalamic circuit with **RE** and **TC** neurons (see thalamic part of Destexhe et al 1998b model), **cortical input** can generate **~3 Hz oscillations** through enhancing **GABA_B** inhibition from **RE→TC**:
     a. This cortical control requires **stronger excitatory input** to RE than to TC, as explained in Destexhe et al 1998b. In Figure 3, the ratio is **120** (1.2 μS vs 0.01 μS).
     b. **Weak** stimulation (either a single event or a 3 Hz stimulation) generates **10 Hz spindles** by evoking **GABA_A**-mediated IPSPs
     c. **Strong** stimulation at **3 Hz** entrains the network into **3 Hz oscillations** by evoking both **GABA_A** and **GABA_B**-mediated IPSPs (the latter lasts **~300 ms**)
     d. **Strong** stimulation at **10 Hz** leads to **quiescence** in TC cells (100 ms < 300 ms)
  4. In a full corticothalamic circuit with **PY, IN, RE** and **TC** neurons (see Destexhe et al 1998b model), **suppression of GABA_A** in the **cortex**, but not in the thalamus, leads to **spike-and-wave oscillations** (this explains Background Fact **#4a**):
     a. Control => **10 Hz spindles** (see Destexhe et al 1998b)
     b. Suppression of GABA_A in **TC** cells
        => **3-5 Hz spindle-like oscillations**
     c. 50% suppression of GABA_A in **PY** cells
        => **2-3 Hz SW-like discharges**
     d. 100% suppression of GABA_A in **PY** cells
        => **2-3 Hz SW oscillations**

   e. 100% suppression of GABA_A in **all** cells (data not shown)
      => **2-3 Hz SW oscillations**
   f. **Spikes**: **high-frequency discharges** in all cells (**TC** cells fire first)
      Waves: **neuronal silence** in all cells for a period of **300-500 ms** (via 2 types of **K+** currents: **GABA_B** from IN→PY and RE→TC & **I_M** in PY cells)
      Some TC cells stay silent throughout entire oscillation
      (this explains Background Fact **#1c**)
   g. Field potentials show progressive transformation from spindles to SW discharges (with decreasing frequency and decreasing amounts of positive spike) as **IN→PY** (**GABA_A** in cortex) is gradually suppressed (this explains Background Fact **#1d**).
      This is because less GABA_A => stronger PY bursts => more GABA_B => hyperpolarizing "waves."
   h. Similarities between spindles and SW discharges:
      Both follow a **waxing and waning envelope**, likely due to **calcium**-dependent upregulation of **I_h** in **TC** cells (an *intrathalamic* mechanism), which may explain Background Fact **#5a**.
      To generate both spindles & SW discharges, **stronger excitatory input** to RE than to TC is required (at least **4 times**). This is consistent with Background Fact **#6a**.

5. Conceptual model: **spindles** are generated by an **intrathalamic loop**, whereas **SW discharges** are generated by a **thalamocortical loop**:
   a. Without the cortex, only spindles can be generated, which explains Background Fact **4b**.



6. The **major determinants** of spindles vs SW discharges are **PY→PY**, **PY→IN**, **IN→PY**, **RE→RE**, **RE→TC (GABA_B)**, **PY→RE**, **T-current** conductance in **RE & TC** cells
   a. Strengthening the following increases **SW discharges** at the expense of spindles:
      PY→PY (increases cortical excitation)

RE→TC (GABA_B) (increases rebound bursts in TC cells)
PY→RE, i.e., **corticothalamic feedback** (increases RE bursts => increases GABA_B IPSPs on TC cells)
T-current conductance in **RE** cells (increases rebound bursts in TC cells; this explains Background Fact **#5b**)
T-current conductance in **TC** cells (increases rebound bursts in TC cells; this explains Background Fact **#5c**)

  b. Strengthening the following reduces SW discharges in favor of **spindles**:
  PY→IN (decreases cortical excitation)
  IN→PY (decreases cortical excitation)
  RE→RE (decreases RE bursts => decreases GABA_B IPSPs on TC cells; this explains Background Fact **#2c** if **clonazepam** acts specifically on GABA_A receptors in RE)

7. SW frequency can be modulated by the kinetics of **GABA_B** and the **T-current** amplitude in **TC** cells
    a. Specifically, changing **K_4** affected only the frequency but not the spiking pattern of SW discharges. This explains Background Fact **#5d**.

- Discussions:
    1. The hypothesis is true. The key biophysical components that generate SW discharges are:
        a. The activation properties of **GABA_B receptors**
        b. The intrinsic firing properties of **RE** & **TC** cells (**T currents**, etc.)
        c. A strong **corticothalamic feedback** (e.g., from diminished intracortical inhibition)
    2. Future directions:
        a. Background Fact #**5e** could possibly be explained by a strengthened **RE→TC (GABA_A)**, but this doesn't seem to affect SW discharges in the model significantly. This type of data might require modeling the **variants of GABA_A receptor types**.
        b. There are purely intracortical SW discharges that could be modeled by pyramidal cells with I_T currents (Note: this has already been done in Destexhe et al 2001)

5/18/2016

Notes from Destexhe et al 2001

- Background Facts:
    1. EEG: **spike-and-wave (SW) complexes** of **2-4 Hz** are found during seizures
    2. **GABA_A antagonists** induce SW paroxysms when injected in the **cerebral cortex**, but not when injected in the thalamus.
    3. During cortical SW seizures, a majority of thalamic neurons are **hyperpolarized and silent**
    4. **Thalamic inactivation** or **thalamectomy** => cortical seizures
- Hypothesis: There are seizures of purely intracortical origin through **intrinsic rebound mechanisms** in some cortical cells.
- Methods:
    1. Background Fact **#2** was repeated with **multisite field potential recordings** from **areas 5-7** of **cat** cerebral cortex under **barbiturate** anesthesia. Seizures were induced by injecting **bicuculline** (0.1 uL, 0.2 mM) into deep layers of the cerebral cortex
    2. Computational model was the same as Destexhe 1998 but with the thalamic layers removed
- Results:
    1. "**Corticothalamic SW**" vs "**intracortical SW**":
        a. Multisite field potential recordings
        b. Barbiturate control: **7-14 Hz** barbiturate spindles
        c. Bicuculline in cortex: **2-4 Hz** "corticothalamic SW"
        d. Bicuculline in cortex after thalamectomy: **1.8 Hz** "intracortical SW"
    2. **LTS activity** are present in cortical pyramidal cells:
        a. In vivo intracellular recordings in same area of cortex (presumably pyramidal cells).
        b. About **10%** of cells show **low-threshold spike (LTS)** activity:
        Depolarizing current injection => adapting trains of action potentials
        Hyperpolarizing current injection => rebound bursting
    3. **Rebound bursts** can be recapitulated by a model of the pyramidal neuron:
        a. Modified from Destexhe 1998 that includes I_T in addition to I_K, I_Na, I_M.
        b. With a depolarizing current, regular spiking behavior is recapitulated.
        c. At the offset of a hyperpolarizing current, rebound bursts are generated. A T-channel density of **0.8 mS/cm^2** was needed to match the data.
        d. With a depolarizing current from hyperpolarized levels, an initial burst is generated followed by an adapting train of action potentials.
    4. **SW discharges** can be recapitulated by a model of the intracortical SW:
        a. Modified from Destexhe 1998 but with thalamic layers removed, and with 20% of pyramidal cells having LTS properties as above.

        b. At baseline, **no** oscillations are generated (in contrast to the full thalamocortical model, where spindles could be generated).
        c. When **GABA_A was suppressed**, sustained oscillations of **1.3 Hz** are generated. **Prolonged discharge** in interneurons generate **GABA_B**-mediated IPSPs in pyramidal neurons with I_T, causing rebound bursting.
        d. Percentage of LTS pyramidal neurons could be as low as **5%**, depending on the connectivity used.
        e. **Extracellular field potentials** calculated shows **SW patterns** that have a **lower frequency** and **less prominent spikes** compared to the thalamocortical model (see Destexhe 1998), in agreement with experimental observations.

- Discussions:
    1. The hypothesis is true.
    2. Future directions:
        a. This model should be **integrated with a thalamocortical model** to see under what conditions intracortical loops prevail over corticothalamic loops.

Plan for next week

1. Continue to reproduce figures in Destexhe et al 1998a
2. Read and write down notes for Traub et al 2005
3. Examine the codes for the Destexhe et al 1998, 2001 model
4. Examine the codes for the Traub et al 2005 model
5. Reproduce figures in Destexhe et al 1998b
6. Reproduce figures in Destexhe 1998
7. Reproduce figures in Destexhe et al 2001
8. Reproduce figures in Traub et al 2005
9. Compile relevant papers that have cited Destexhe et al 1996 and/or have used the Destexhe et al 1996 model
10. Compile relevant papers that have cited Destexhe et al 1998b and/or Destexhe 1998 and/or Destexhe et al 2001 and/or have used the Destexhe et al 1998, 2001 model
11. Compile relevant papers that have cited Traub et al 2005 and/or have used the Traub et al 2005 model
12. Optional? Examine the codes for the Destexhe et al 1996 model

Plan for the future

1. Compile relevant papers about absence epilepsy
2. Read and write down notes for Chen et al 2014 and Chen et al 2015
3. Read and write down notes for Zhao et al 2015
4. Examine the codes for the Chen et al 2014, 2015 model
5. Examine the codes for the Zhao et al 2015 model
6. Examine the codes for the Destexhe et al 1996 model
7. Go through the NEURON Hands-On Course
8. “Reproduce CSD graph” exercise
9. Examine Christine’s & Mark’s codes
10. Finish NEURON Book Appendix A1
11. Figure out how to export NEURON to Matlab
12. Complete the NEURON Tutorial
13. Understand NEURON Ch 7 & Ch 8
14. Resolve all NEURON Book questions
15. Read Abbott et al 2016 (“Building functional networks of spiking model neurons”)
16. Read Markram et al 2015 (“Reconstruction and Simulation of Neocortical Microcircuitry”)
17. Read Kragel & LaBar 2016 (“Decoding the Nature of Emotion in the Brain”)
18. Read Izhikevich: Dynamical Systems in Neuroscience
19. Read Dayan & Abbott: Theoretical Neuroscience
20. Derivation & shape of the **Goldman–Hodgkin–Katz flux equation**

5/5/2016~5/15/2016

Details of Destexhe et al 1998a model (continued)

- **cadecay.mod**
  - Fast mechanism for submembranal Ca++ concentration (cai)
  - Suffix: "**cad**" (same as calcium pump)
  - Input/Output: reads **ica** ([mA/cm²]) & **cai**, writes **cai**
  - Parameters:

Name	Description	Default value	Range/ global
depth	Depth of the shell just beneath the membrane [μm]	0.1	range
cainf	Equilibrium concentration of calcium [mM]	2e-4	range
taur	Time constant of calcium extrusion, must be fast) [ms]	5	range
kt, kd	Dummy parameters (parameters specific to the calcium pump)		range

  - States & initialization:

Name	Description	Initialization
cai	submembranal Ca++ concentration [mM]	cainf

  - Equations:
    - **Inward rectification**:
      drive_channel = - (10000) * ica / (2 * FARADAY * depth)
      : 10000 μm/cm
      if (drive_channel <= 0.) { drive_channel = 0. } : cannot pump inward
      : (ica should be negative)
    - Differential equations:
      $$\frac{d[Ca]_i}{dt} = -\frac{I_{Ca}}{2Fd} + \frac{[Ca]_\infty - [Ca]_i}{\tau_r}$$
      where F is Faraday's constant, d is the depth of the shell just beneath the membrane
    - cai' = drive_channel + (cainf - cai)/taur

- **hh2.mod**
  - Hippocampal Hodgkin-Huxley channels
  - **Q10** was assumed to be **3** for both currents
  - Suffix: "**hh2**"
  - Input/Output: reads **ena** ([mV]) & **ek** ([mV]), writes **ina** [mA/cm²] & **ik** [mA/cm²]
  - Parameters:

Name	Description	Default	Range/

		value	global
gnabar	Maximum sodium conductivity [S/cm²]	.003	range
gkbar	Maximum potassium conductivity [S/cm²]	.005	range
ena	Reversal potential of sodium channel [mV]	50	global
ek	Reversal potential of potassium channel [mV]	-90	global
celsius	Temperature [℃]	36	global
vtraub	Threshold v_T [mV]	-63	range

- Other variables visible to hoc:

Name	Description	Default value	Range/ global
m_inf	Asymptotic sodium activation gating variable		range
h_inf	Asymptotic sodium inactivation gating variable		range
n_inf	Asymptotic potassium activation gating variable		range
tau_m	Time constant for sodium activation		range
tau_h	Time constant for sodium activation		range
tau_n	Time constant for sodium activation		range
m_exp	$1 - e^{-(t_{i+1}-t_i)/\tau_m}$		range
h_exp	$1 - e^{-(t_{i+1}-t_i)/\tau_h}$		range
n_exp	$1 - e^{-(t_{i+1}-t_i)/\tau_n}$		range

- States:

Name	Description	Initialization
m	gating variable for activation of sodium current	0
h	gating variable for inactivation of sodium current	0
n	gating variable for activation of potassium current	0

- Equations:
  - First, update gating variables:

$$T_{adj} = {Q_{10}}^{(T-36)/10}, Q_{10} = 3$$

$$m_{i+1} = m_i + (1 - e^{-(t_{i+1}-t_i)/\tau_m})(m_\infty - m_i)$$
$$a_m = 0.32(\frac{13-(V-V_T)}{e^{(13-(V-V_T))/4}-1})$$
$$b_m = 0.28(\frac{(V-V_T)-40}{e^{((V-V_T)-40)/5}-1})$$
$$\tau_m = \frac{1}{T_{adj}(a_m+b_m)}$$
$$m_\infty = \frac{a_m}{a_m+b_m}$$
$$h_{i+1} = h_i + (1 - e^{-(t_{i+1}-t_i)/\tau_h})(h_\infty - h_i)$$
$$a_h = 0.128(\frac{17-(V-V_T)}{18})$$
$$b_h = \frac{4}{1+e^{(40-(V-V_T))/5}}$$
$$\tau_h = \frac{1}{T_{adj}(a_h+b_h)}$$
$$h_\infty = \frac{a_h}{a_h+b_h}$$
$$n_{i+1} = n_i + (1 - e^{-(t_{i+1}-t_i)/\tau_n})(n_\infty - n_i)$$
$$a_n = 0.032(\frac{15-(V-V_T)}{e^{(15-(V-V_T))/5}-1})$$
$$b_n = 0.5e^{(10-(V-V_T))/40}$$
$$\tau_n = \frac{1}{T_{adj}(a_n+b_n)}$$
$$n_\infty = \frac{a_n}{a_n+b_n}$$

- Next, update currents:
$$I_{Na} = \overline{g}_{Na} m^3 h(V - E_{Na})$$
$$I_K = \overline{g}_K n^4 (V - E_K)$$
- Equations modified by Traub, for Hippocampal Pyramidal cells, in: Traub & Miles, Neuronal Networks of the Hippocampus, Cambridge, 1991

○ Procedures and functions:

Name & Arguments	Description	Called by
states()	Updates state variables **m, h, n** based on current voltage	
evaluate_fct(v(mV))	Updates **m_inf, h_inf, n_inf, tau_m, tau_h, tau_n, m_exp, h_exp, n_exp** based on	states()

	current voltage	

- **ITGHK.mod**
  - T-type calcium current responsible for low-threshold spikes (LTS)
  - Model of Huguenard & McCormick, J Neurophysiol 68: 1373-1383, 1992.
  - The kinetics is described by **Goldman-Hodgkin-Katz equations**, using a m^2h format, according to the voltage-clamp data (whole cell patch clamp) of Huguenard & Prince, J. Neurosci. 12: 3804-3817, 1992."
  - The activation function was empirically corrected to account for the contamination of inactivation:
    - An overall **hyperpolarizing shift** of **2 mV** was applied to compensate for **screening charge**
    - However, an overall **depolarizing shift** of **3 mV** was necessary to reproduce the current-clamp simulations of TC cells in the paper, but the same shift was applied to voltage clamp simulations as well
  - All voltage-clamp simulations were done at **24°C** assuming Q10 values of **2.5** for both m and h, whereas current-clamp behavior was simulated at **34°C**.
  - Suffix: “**itGHK**”
  - Input/Output: reads **cai** [mM] & **cao** [mM], writes **ica** [mA/cm²]
  - Parameters:

Name	Description	Default value	Range/ global
celsius	Temperature [°C]	36*	global
pcabar	Maximum calcium permeability in the GHK equation [cm/s]	0.2e-3	range
shift	Shift towards hyperpolarization of *both activation & inactivation curves* [mV]	2**	range
actshift	Shift towards hyperpolarization of *activation curve only* [mV]	0	range
cai	Calcium concentration inside the cell [mM]	2.4e-4	global
cao	Calcium concentration outside the cell [mM]	2	global
qm	Q_10 for activation curve [1]	5***	global
qh	Q_10 for inactivation curve [1]	3***	global

    - *Default temperature mimics physiological conditions. However, **celsius = 24** for all voltage-clamp simulations & **celsius = 34** for all current-clamp simulations
    - **Default shift corresponds to 2 mM ext Ca++ (compensates for screening charge). However, shift = **-1 mV** in all simulations of this paper.

- ***Default Q_10s are from Coulter et al., J Physiol 414: 587, 1989. However, they are both changed to **2.5** in all simulations of this paper

- Other variables visible to hoc:

Name	Description	Default value	Range/ global
m_inf	Asymptotic calcium activation gating variable		range
h_inf	Asymptotic calcium inactivation gating variable		range
tau_m	Time constant for calcium activation		range
tau_h	Time constant for calcium activation		range

- States:

Name	Description	Initialization
m	gating variable for activation of calcium current	m_inf
h	gating variable for inactivation of calcium current	h_inf

- Equations:
  - First, update gating variables:

$$\frac{dm}{dt} = \frac{m_\infty - m}{\tau_m}$$

$$\frac{dh}{dt} = \frac{h_\infty - h}{\tau_h}$$

$$m_\infty = \frac{1}{1+e^{-(V+57+shift+actshift)/6.2}}$$

(Here, **V_1/2** is assumed to be -57 mV, but can be modified by **shift** & **actshift**, which is **-1** and **0**, respectively, in all simulations of this paper)

$$h_\infty = \frac{1}{1+e^{(V+81+shift)/4}}$$

(Here, V_1/2 is assumed to be -81 mV, but can be modified by **shift**, which is **-1** in all simulations of this paper)

$$\tau_m = \frac{1}{\Phi_m}\left(0.612 + \frac{1}{e^{-(V+132+shift+actshift)/16.7}+e^{(V+16.8+shift+actshift)/18.2}}\right)$$

(**shift** = **-1** in all simulations of this paper)

$$\tau_h = \frac{1}{\Phi_h} e^{(V+467+shift)/66.6} \quad \text{for V < -80 mV}$$

$$\tau_h = \frac{1}{\Phi_h}\left(28 + e^{-(V+22+shift)/10.5}\right) \quad \text{for V ≥ -80 mV}$$

(**shift** = **-1** in all simulations of this paper)

$$\Phi_m = Q_{10,m}^{(T-24)/10}$$

In all simulations, Q_10,m = **2.5**. Since T = **24°C** & **34°C** for voltage-clamp & current-clamp simulations, respectively, 1/phi_m = 1 & **0.333**, respectively.

$$\Phi_h = Q_{10,h}^{(T-24)/10}$$

In all simulations, Q_10,h = **2.5**. Since T = **24°C** & **34°C** for voltage-clamp & current-clamp simulations, respectively, 1/phi_h = 1 & **0.333**, respectively.

- Next, update currents:

$$I_{Ca} = \overline{P}_{Ca} m^2 h \; G(v, [Ca]_o, [Ca]_i)$$

$$G(V, [Ca]_o, [Ca]_i) = Z^2 F^2 V/RT \frac{[Ca]_i - [Ca]_o e^{-ZFV/RT}}{1-e^{-ZFV/RT}}$$

where Z = 2, T is in [K], V is in [V]. This is based on the **Goldman–Hodgkin–Katz flux equation**

- Procedures and functions:

Name & Arguments	Description	Called by
evaluate_fct(v(mV))	Update **m_inf, h_inf, tau_m, tau_h** based on current voltage	INITIAL, DERIVATIVE
ghk(v(mV), ci(mM), co(mM)) (.001 coul/cm3)	Computes the **Goldman-Hodgkin-Katz flux** based on current voltage, concentration inside the cell, concentration outside the cell	BREAKPOINT, **nongat()**
efun(z)	z/(exp(z) - 1) with Taylor approximation when |z| < 1e-4, z is a floating point number (uses NMODL intrinsic function **fabs**)	**ghk()**
nongat(v,cai,cao)	**Non-gated** version of the calcium current nongat = pcabar * ghk(v, cai, cao)	NONE

- **ghk** has the structure:

    (.001)*2*FARADAY*(ci*efun(-z) - co*efun(z))

  where

    efun(z) = z/(exp(z) - 1)

  and

    z = (1e-3 [V/mV])*2*FARADAY*v/(R*(celsius+273.15))

  For |z| < **1e-4**, the **1st order Taylor approximation**

    z/(exp(z) - 1) ~ 1 - z/2 is used

  However, since ci & co are in [mM] = [μmol/cm³]=10^-6 [mol/cm³], I don't think the units match up…

- **VClamp.mod**
    - Single electrode Voltage clamp with three levels
    - “Do not insert several instances of this model at the same location in order to make level changes. That is equivalent to independent clamps and they will have incompatible internal state values.”
    - Here, **i** is an electrode current, so positive values of i depolarize the cell
    - Electrical circuit for the clamp:

```
              rs           Rin
     vc ---'\/\/`--- o  ---'\/\/`  ---   o
                     |                   |
                     |_____| |_____|
                           | |
                           Cm
```

-
    - Suffix: “SEVClamp”
    - Parameters:

Name	Description	Default value	Range/ global
rs	Series resistance [MΩ]	1	range

-
    - Other variables visible to hoc:

Name	Description	Default value	Range/ global
dur[3]	Duration for each voltage clamp level [ms]		range
amp[3]	Amplitude for each voltage clamp level [mV]		range
vc	[mV]		range
i	[nA]		range

-
    - Equations:
        - Specification:

$$I = \frac{V_c - V}{R_i}$$

where V_c is the voltage level that is clamped

-
    - Procedures:

Name & Arguments	Description	Called by
vstim()	Sets voltage clamp level based on current time **t**	BREAKPOINT

- **el.oc**

- Defines the class **Electrode**
- "A current injection electrode inserted in the middle of the current section which can be switched between current and voltage clamp modes and can do simple voltage clamp families."
- "Electrode can be saved in a .session file and is best used anonymously so that it is dismissed and point processes deleted when the graphic is dismissed."
- Usage:
    - section e = new Electrode([xplacement, yplacement])
- External functions needed (all in **stdrun.hoc**):
  run(), **set_v_init()**, **stoppedrun()**, **addplot()**
- Public objects:

Name	Description	Class	Variables (default values) and procedures used
stim	Current clamp	IClamp	**del** (**0.1** ms) **dur** (**0.1** ms) **amp** (**0** mV)
vc	Voltage clamp	SEVClamp	**dur[3]** (**0.1** ms, **5** ms, **100** ms) **amp[3]** (**-65** mV, **10** mV, **-65** mV)

v1	The main window that contains a panel and a deck	VBox	**map()**
			unmap()
			ref()
			save()
			intercept()

- Public functions/procedures:

Name & Arguments	Description	Called by
installIclamp()	1. Switches deck to **IClamp** mode 2. Restores the **amplitude** for **stim** 3. Makes **vc.dur[i]** equal to **-1** for all i	glyph()
map()	Stores current current clamp and voltage clamp parameters into **samp**, **vdur[3]** & **vamp[3]** (the latter by calling **store_vclamp()**) Creates **v1** & **d1** by calling **glyph()** Creates a window for **v1** with the label "**I/V Clamp Electrode**"; if there are 2 arguments, $1 is the left x-coordinate of the window, $2 is the top y-coordinate of the window and the width & height of the window are both **100 pixels**	init()
unmap()	Dismisses the last mapped window depicting **v1**	NONE

- Private objects:

Name	Description	Class	Variables and procedures used

d1	A deck of 4 cards (IClamp, VClamp, VClamp Family, Location)	Deck	**flip_to()**
			intercept()
			map()

this	Refers to the instance of the current electrode	Electrode	

shape	Shape plot	Shape	**point_mark()**
			action()
			select()

grbox	The window that contains the graph **g**	VBox	**save()**
			intercept()
			map()

g	Graph of current versus time	Graph	**addvar()**

- Private variables:

Name	Description	Type	Notes
durstr	"dur *vdur[0] vdur[1] vdur[2]*"	string	
ampstr	"dur *vamp[0] vamp[1] vamp[2]*"	string	
tempstr	temporary string	string	
grname	"*vc* Graph" where **vc** is the name of the voltage clamp	string	
vdur[3]	Stores voltage clamp durations	double	
vamp[3]	Stores voltage clamp amplitudes	double	
samp	Stores current clamp amplitude	float	
sec	The name of the currently accessed section	string	
xloc	The location of the electrode in the section	float	
location	The string "*sec*(*xloc*)"	string	

- Private functions/procedures:

Name & Arguments	Description	Called by

set_vclamp()	Sets **vc** parameters according to **vdur**[3] & **vamp**[3]	installVclamp()
		glyph()

store_vclamp()	Stores **vc** parameters into **vdur**[3] & **vamp**[3]	map()
		save()

installVclamp()	1. Switches deck to **VClamp** mode
	2. Saves the **amplitude** for **stim** but make it 0
	3. Calls **set_vclamp()** to set vc parameters
		installFamily()
		glyph()

installFamily()	1. Calls **installVclamp()** to set vc, etc. 2. Switches deck to **VClamp Family** mode 3. Prints **vdur**[3] and **vamp**[3] values in **durstr** & **ampstr**	glyph()
init()	1. Stores the name of the currently accessed section in **sec** (built-in function **sectionname()**) 2. Sets the location of the electrode (**xloc**) at **0.5** 3. Creates the IClamp **stim** and set default parameters 4. Creates the SEVClamp **vc**, prints its name in **grname** and set default parameters 5. Creates a window for the electrode (calls **map()**)	
locate()	1. If $1==1, flip deck to **Location** 2. Otherwise, [don't know what this code is for]	glyph()
move()	Update **location** based on selected section (**hoc_ac_** contains the arc position 0 - 1 of the nearest node to the mouse click)	locate(0) & shape.action() in glyph()

glyph()	Sets up a panel (**v1**) that has 4 radio buttons (**xradiobutton()**):
	1. IClamp (calls **installIclamp()**)
	2. VClamp (calls **installVclamp()**)
	3. VClamp Family (calls **installFamily()**)
	4. Location (calls **locate(1)**)
	Sets up a deck (**d1**) that has 4 cards:
	1. VClamp
	a. Shows the string "*vc* pulse," where *vc* is the name of **vc**
	b. Creates field editors for **vdur[0]**, **vdur[1]**, **vdur[2]**, **vamp[0]**, **vamp[1]**,
		map()
		save()

	vamp[2] c. Changing values in the field editor calls **set_vclamp()** d. Creates a button called "VClamp.i graph" that calls **mkgraph()** 2. IClamp a. Shows the string "*stim* pulses," where *stim* is the name of **stim** b. Creates field editors for **stim.del**, **stim.dur**, **stim.amp** 3. VClamp Family a. Shows the string "*vc* Families," where *vc* is the name of **vc** b. Prints **durstr**, **ampstr** c. Creates 3 buttons: "Test level" calls **varyamp(1)** "Holding" calls **varyamp(0)** "Return level" calls **varyamp(2)** 4. Location a. Creates a shape plot (**shape**) b. Draw a blue dot (**3** in shape.**pointmark()**) where the electrode is located on the cell c. Whenever the mouse clicks, call **move()** d. Calls **locate(0)** Maps **d1** inside **v1** Calls **installVclamp()** & **installIclamp()** Flips the deck to Location	
mkgraph()	1. Creates a graph of the voltage clamp current (**vc.i**) versus time in a window named **grname** 2. Calls **addplot()** from stdrun.hoc: Changes the axis to [0,tstop,-1,1] and adds the graph to GraphList[0]	glyph()
varyamp()	1. $1 == 0 will vary the "Holding level" a. Ramps **vc.amp[0]** = **-100**:**5**:**-50** b. Sets v_init to the new holding level temporarily by calling **set_v_init()** from stdrun.hoc 2. $1 == 1 will vary the "Test level" () a. Ramps **vc.amp[1]** = **-50**:**10**:**40** 3. $1 == 2 will vary the "Return level" a. Ramps **vc.amp[2]** = **-100**:**10**:**0**	glyph()

	Calls **run()** from stdrun.hoc for each condition; breaks if Stop button is clicked	
save()	How the VBox **v1** is to be saved: [specifics that I haven't understood]	

- Other functions/procedure outside template:

Name & Arguments	Description	Called by
makeelectrode()	Creates an Electrode object	

- **loc3.oc**
    - Procedure to **localize T-type calcium channels differentially** in soma and dendrites for the **3-compartment model**
    - Functions/procedures:

Name & Arguments	Description	Called by
localize()	Changes the maximum calcium permeability in the GHK equation for T-type calcium current (**pcabar_itGHK**) to: ● $1 for **soma** & **dend1[0]** ● $2 for **dend1[1]**	

- **loc200.oc**
    - Procedure to **localize T-type calcium channels differentially** in soma and dendrites for the **detailed cell model**
    - Functions/procedures:

Name & Arguments	Description	Called by
localize()	Changes the maximum calcium permeability in the GHK equation for T-type calcium current (**pcabar_itGHK**) to: ● $1 for **soma**, **dend3[0]**, **dend4[0]**, **dend6[0]**, **dend7[0]**, **dend8[0]**, **dend9[0]**, **dend10[0]**, **dend11[0]** ● $2 for dend2[0], dend3[1], dend3[8], dend4[1], dend4[4], dend6[1], dend6[2], dend6[3], dend6[8], dend6[13], dend6[20], dend7[1], dend7[4], dend7[5], dend8[1], dend8[14], dend8[24], dend9[7], dend9[8], dend9[22], dend9[26], dend10[2], dend10[6], dend10[7], dend10[8], dend10[11], dend10[16], dend11[1], dend11[10], dend11[13], dend11[14], dend11[15], dend11[16], dend11[18]	

	• $3 for **all other sections**	

- **locD.oc**
  - Procedure to **localize T-type calcium channels differentially** in soma and dendrites for the **dissociated cell model**
  - Functions/procedures:

Name & Arguments	Description	Called by
localize()	Changes the maximum calcium permeability in the GHK equation for T-type calcium current (**pcabar_itGHK**) to: • $1 for **soma**, **dend1[0] & dend2[0]** • $2 for **all other sections**	

- **tc1_cc.oc**
  - **Current clamp** simulations of the **single-compartment model**
  - Reproduces parts of **Figure 11** of the paper
  - Creates a maximum of **20** graphs
  - Geometry is set up by loading **cell/tc1.geo**
  - Soma has leak channels (**pas**) inserted and have the following parameter values:

Name	Description	Default value	Notes
g_pas	Leak current conductance [S/cm²]	**G_pas**	
e_pas	Leak current reversal potential [mV]	**E_pas**	
cm	Specific capacitance [μF/cm²]	**0.88**	
Ra	Cytoplasmic resistivity [Ω cm]	**173**	

  - Soma has HH channels (**hh2**) inserted and has the following parameter values:

Name	Description	Default value	Notes
ena	Reversal potential of sodium channel [mV]	**50**	
ek	Reversal potential of potassium channel [mV]	**-100**	
vtraub_hh2	Threshold v_T [mV]	**-52**	
gnabar_hh2	Maximum sodium conductivity [S/cm²]	**0.01**	

gkbar_hh2	Maximum potassium conductivity [S/cm²]	**0.01**	

- Soma has T-type calcium channels (**itGHK**) inserted and has the following parameter values:

Name	Description	Default value	Notes
cai	Calcium concentration inside the cell [mM]	**2.4e-4**	
cao	Calcium concentration outside the cell [mM]	**2**	
eca	Reversal potential of calcium channel [mV]	**120**	
shift_itGHK	Shift towards hyperpolarization of *both activation & inactivation curves* [mV]	**-1**	
gcabar_itGHK	**??**	**0.0002**	
pcabar_itGHK	Maximum calcium permeability in the GHK equation [cm/s]	**8e-5***	
qm_itGHK	Q_10 for activation curve [1]	**2.5**	
qh_itGHK	Q_10 for inactivation curve [1]	**2.5**	

* This is the value “in order to get correct bursting behavior.” However, the “closest IV curve to detailed model with dendritic density of 8.5e-5” requires soma.pcabar_itGHK = **6e-5**, whereas “same amount of T-channels as the intact-cell model” gives soma.pcabar_itGHK = **7.452e-5**

- Soma has a calcium extrusion mechanism (**cadecay**) inserted and has the following parameter values:

Name	Description	Default value	Notes
depth_cad	Depth of the shell just beneath the membrane [μm]	**0.1**	
cainf_cad	Equilibrium concentration of calcium [mM]	**2.4e-4**	
taur_cad	Time constant of calcium extrusion, must be fast) [ms]	**5**	
kt_cad	Dummy parameters (parameters specific to the calcium pump)	1e-4	

kd_cad	Dummy parameters (parameters specific to the calcium pump)	**0** (no pump)	

- Soma has an Electrode (**E1**) placed at x = **0.5** and has the following parameter values:

Name	Description	Default value	Notes
stim.del	Current clamp delay [ms]	**480**	
stim.dur	Current clamp duration [ms]	**900**	
stim.amp	Current clamp amplitude [mV]	**0.05**	

- Simulation parameters:

Name	Description	Default value	Notes
Dt	Time interval between points plotted [ms]	**0.2**	
npoints	Number of points plotted	**4000**	
dt	Time step of integration [ms]	**0.1**	
tstart	Start time to plot [ms]	trans	
tstop	End time to plot [ms]	trans + npoints*Dt	
runStopAt		tstop	
steps_per_ms	Points plotted per ms	1/Dt	
celsius	Temperature of experiment [°C]	**34**	
v_init	Resting membrane potential [mV]	**-74**	

- Other Global variables/objects:

Name	Description	Type/ Class	Notes
g[i]	Graphs (including voltage vs time and shape plots)	Graph	
ngraph	Number of graphs created so far	float	
trans	Sets a random start time [ms]	float	Default: **0**
G_pas	Leak current conductance [S/cm²]	float	Default: **3.79e-5**

E_pas	Leak current reversal potential [mV]	float	Default: **-76.5**
electrodes_present	==1 if electrodes are present	float	

- Functions/procedures:

Name & Arguments	Description	Called by
addgraph("*variable*", minvalue, maxvalue)	Creates a graph with axes == [**tstart**,**tstop**,**$2**,**$3**], variable == **$s1**, color index == **1** (**black**), brush index == 0 g[ii].save_name("graphList[0].")	
addshape()	Creates a Shape Plot (Class **PlotShape**) whose default y-axis for time and space plots are [**-130**, **50**]	

- Creates the following graphs:
    - A voltage vs time plot of **soma.v(0.5)** with y-axis [**-120**,**40**]
    - A shape plot

- **tc3_cc.oc**
    - **Current clamp** simulations of the **3-compartment model**
    - Reproduces parts of **Figure 11** of the paper
    - Creates a maximum of 20 graphs
    - Geometry is set up by loading **cell/tc3.geo**
    - **All sections** have leak channels (**pas**), T-type calcium channels (**itGHK**) & calcium extrusion mechanisms (**cadecay**) inserted and share the same parameter values as in **tc1_cc.oc** except:

Name	Description	Default value	Notes
gcabar_itGHK	**??**	**0.0002*corrD**	
depth_cad	Depth of the shell just beneath the membrane [µm]	**0.1*corrD**	

- **soma** has HH channels (**hh2**) inserted and has the same parameter values as in **tc1_cc.oc** except:

Name	Description	Default value	Notes
gnabar_hh2	Maximum sodium conductivity [S/cm²]	**0.1**	
gkbar_h	Maximum potassium conductivity [S/cm²]	**0.1**	

h2			

- **dend1[0] & dend1[1]** share the following parameter values:

Name	Description	Default value	Notes
g_pas	Leak current conductance [S/cm²]	**G_pas*corrD**	
cm	Specific capacitance [μF/cm²]	**0.88*corrD**	

- **soma & dend1[0]** share the following parameter values (using **localize()** from **loc3.oc**):

Name	Description	Default value	Notes
pcabar_itGHK	Maximum calcium permeability in the GHK equation [cm/s]	**1.7e-5**	

- **dend1[1]** has the following parameter values (using **localize()** from **loc3.oc**):

Name	Description	Default value	Notes
pcabar_itGHK	Maximum calcium permeability in the GHK equation [cm/s]	**9.5e-5*corrD**	

  - This is the value for "**high distal density**, match detailed model in voltage-clamp."
  - The value for "**uniform T-current** with same density as in dissociated cells" is **1.7e-5*corrD**
  - The value for "high distal density, same total T-channels as intact cell" is **9.634e-5*corrD**
- **soma** has an Electrode (**E1**) placed at x = **0.5** and has the same parameter values as in **tc1_cc.oc**
- Simulation parameters are the same as in **tc1_cc.oc**
- Other Global variables/objects are the same as in **tc1_cc.oc** except:

Name	Description	Type/ Class	Notes
corrD	Correction factor for dendritic surface estimated by fitting voltage-clamp data	float	Default: **7.954**

- Functions/procedures are the same as in **tc1_cc.oc**
- Creates the following graphs:
  - A voltage vs time plot of **soma.v(0.5)** with y-axis [**-120**,**40**]
  - A voltage vs time plot of **dend1[0].v(0.5)** with y-axis [**-120**,**40**]
  - A voltage vs time plot of **dend1[1].v(0.5)** with y-axis [**-120**,**40**]

        - A shape plot

- **tc200_cc.oc**
    - **Current clamp** simulations of the **detailed cell model**
    - Reproduces parts of **Figure 9** of the paper
    - Creates a maximum of 20 graphs
    - Geometry is set up by loading **cell/tc200.geo**
    - Mechanisms, biophysical properties, functions/procedures, placement of electrodes, other variables, are the same as in **tc3_cc.oc** except that **corrD = 1** (no need for dendritic surface correction)
    - **soma**, **dend3[0]**, **dend4[0]**, **dend6[0]**, **dend7[0]**, **dend8[0]**, **dend9[0]**, **dend10[0]**, **dend11[0]** share the following parameter value (using **localize()** from **loc200.oc**):

Name	Description	Default value	Notes
pcabar_itGHK	Maximum calcium permeability in the GHK equation [cm/s]	**1.7e-5**	

    - **All other dendritic sections** share the following parameter value (using **localize()** from **loc200.oc**):

Name	Description	Default value	Notes
pcabar_itGHK	Maximum calcium permeability in the GHK equation [cm/s]	**8.5e-5*corrD**	

        - This is the value for “low density proximal, high distal, to match volt-clamp of intact cells”
        - The value for “**uniform T-current** with same density as in dissociated cells“ is **1.7e-5*corrD**
    - Creates the following graphs:
        - A voltage vs time plot of **soma.v(0.5)** with y-axis [**-120**,**40**]
        - A voltage vs time plot of **dend10[6].v(0.5)** with y-axis [**-120**,**40**] (this is a **proximal** dendrite)
        - A voltage vs time plot of **dend10[26].v(0.5)** with y-axis [**-120**,**40**] (this is a **distal** dendrite)
        - A shape plot

- **tc200_vc.oc**
    - **Voltage clamp** simulations of the **detailed cell model**
    - Reproduces parts of **Figure 6** of the paper
    - Creates a maximum of 20 graphs
    - Geometry is set up by loading **cell/tc200.geo**

- Mechanisms, biophysical properties, functions/procedures, other variables, are the same as in **tc200_cc.oc** except that:
  - **no HH current** is present in the soma
  - **pcabar_itGHK = 8e-5*corrD** for "**all other dendritic sections"** in the localized high density case
  - **trans = 1000** by default
  - **g_pas = 0** in all sections
- **soma** has an Electrode (**E1**) placed at x = **0.5** and has the following parameter values:

Name	Description	Default value	Notes
vc.dur[0]	Duration of voltage clamp level 1 [ms]	**trans**	
vc.dur[1]	Duration of voltage clamp level 2 [ms]	**1000**	
vc.dur[2]	Duration of voltage clamp level 3 [ms]	**1000**	
vc.amp[0]	Amplitude of voltage clamp level 1 [mV]	**-115**	
vc.amp[1]	Amplitude of voltage clamp level 2 [mV]	**-65**	
vc.amp[2]	Amplitude of voltage clamp level 3 [mV]	**-65**	
vc.rs	Series resistance [MΩ]	**5**	

- Simulation parameters are the same as in **tc200_cc.oc** except:

Name	Description	Default value	Notes
npoints	Number of points plotted	**1000**	
celsius	Temperature of experiment [℃]	**24**	
v_init	Resting membrane potential [mV]	**-70**	

- Creates the following graphs:
  - A current vs time plot of **E1.vc.i** with y-axis [**-10**,**0.001**]
  - A voltage vs time plot of **soma.v(0.5)** with y-axis [**-120**,**40**]
  - A voltage vs time plot of **dend10[26].v(0.5)** with y-axis [**-120**,**40**] (this is a **distal** dendrite)

- **tcD_vc.oc**
  - **Voltage clamp** simulations of the **dissociated cell model**
  - Reproduces parts of **Figure 6** of the paper
  - Creates a maximum of 20 graphs
  - Geometry is set up by loading **cell/tcD.geo**

- Mechanisms, biophysical properties, functions/procedures, other variables, are the same as in **tc200_vc.oc** except that:
    - **pcabar_itGHK = 1.7e-5*corrD** for “all other sections except soma, dend1[0] & dend2[0]” (**uniform** distribution)
- **soma** has an Electrode (**E1**) placed at x = **0.5** and has the same parameter values as in **tc200_vc.oc** except:

Name	Description	Default value	Notes
vc.amp[1]	Amplitude of voltage clamp level 2 [mV]	**-30**	
vc.amp[2]	Amplitude of voltage clamp level 3 [mV]	**-30**	

- Creates the following graphs:
    - A current vs time plot of **E1.vc.i** with y-axis [**-10**,**0.001**]
    - A voltage vs time plot of **soma.v(0.5)** with y-axis [**-120**,**40**]
    - A voltage vs time plot of **dend2[4].v(0.5)** with y-axis [**-120**,**40**] (this is a **distal** dendrite)

Plan for next week

1. Write down notes for Destexhe et al 1998b
2. Finish compiling relevant papers that have cited Destexhe et al 1998a and/or have used the Destexhe et al 1998 model
3. Read and write down notes for Destexhe et al 2001
4. Read and write down notes for Traub et al 2005
5. Compile relevant papers that have cited Destexhe et al 1996 and/or have used the Destexhe et al 1996 model
6. Compile relevant papers that have cited Destexhe et al 1998b and/or Destexhe et al 2001 and/or have used the Destexhe et al 1998, 2001 model
7. Compile relevant papers that have cited Traub et al 2005 and/or have used the Traub et al 2005 model
8. Reproduce curves in Destexhe et al 1998a
9. Examine the codes for the Destexhe et al 1996 model
10. Examine the codes for the Destexhe et al 1998, 2001 model
11. Examine the codes for the Traub et al 2005 model
12. Compile relevant papers about absence epilepsy

Plan for the future

1. Read and write down notes for Chen et al 2014 and Chen et al 2015
2. Read and write down notes for Zhao et al 2015
3. Examine the codes for the Chen et al 2014, 2015 model
4. Examine the codes for the Zhao et al 2015 model
5. Go through the NEURON Hands-On Course
6. "Reproduce CSD graph" exercise
7. Examine Christine's & Mark's codes
8. Finish NEURON Book Appendix A1
9. Figure out how to export NEURON to Matlab
10. Complete the NEURON Tutorial
11. Understand NEURON Ch 7 & Ch 8
12. Resolve all NEURON Book questions
13. Read Abbott et al 2016 ("Building functional networks of spiking model neurons")
14. Read Markram et al 2015 ("Reconstruction and Simulation of Neocortical Microcircuitry")
15. Read Kragel & LaBar 2016 ("Decoding the Nature of Emotion in the Brain")
16. Read Izhikevich: Dynamical Systems in Neuroscience
17. Read Dayan & Abbott: Theoretical Neuroscience
18. Derivation & shape of the **Goldman–Hodgkin–Katz flux equation**

5/1/2016~5/2/2016

Notes from the NEURON Book Ch 11

- It's more convenient to set up and test a small network with the **GUI**, generate a **hoc file**, then expand the code for larger networks.
- Artificial Cell Builder:
    - Artificial cells are built with the **ArtCellGUI** tool
    - After the network is built, adjustments to ArtCellGUI takes effect immediately
- Biophysical Neuron Builder:
    - Biophysical neurons that will be used in networks are built with the **NetReadyCellGUI** tool
    - A separate NetReadyCellGUI instance is needed for each different *type* of biophysical neuron model
    - After the network is built, adjustments to NetReadyCellGUI *do not* take effect immediately. It is necessary to save the session file, exit NEURON, restart and reload the session file.
    -
- NetWork Builder:
    - Networks are built with the **NetWork Builder** tool, which is an instance of the **NetGUI** class.
    - Weights are **0** by default. Delays are **1 ms** by default.
    - Toggling the **Create** button on creates the network
    - **SpikePlot** plots the spike trains of Cell i on the line **y = i+1**
    - After the network is built, only certain adjustments take effect immediately. It is better practice to save the session file, exit NEURON, restart and reload the session file.
- hoc file:
    - Three sections: Network cell templates, Network specification interface & Network instantiation. The former two can be saved as a reusable code for larger networks.
    - Each cell type has its own cell class template, which is named by a concatenation of the cell "type" and the root cell type (e.g., "**IF_IntervalFire**"). These are listed at the top of the file. Biophysical neuron models are listed before artificial cell models
    - Each cell class has an instance of a point process called **pp**.
    - Each cell class has the following functions:

init()	Creates a new instance of the cell class
is_art()	Checks if the cell class is an artificial cell
connect2target()	Creates a NetCon, an **obfunc**
position()	Set the xyz coordinates for each instance of the cell

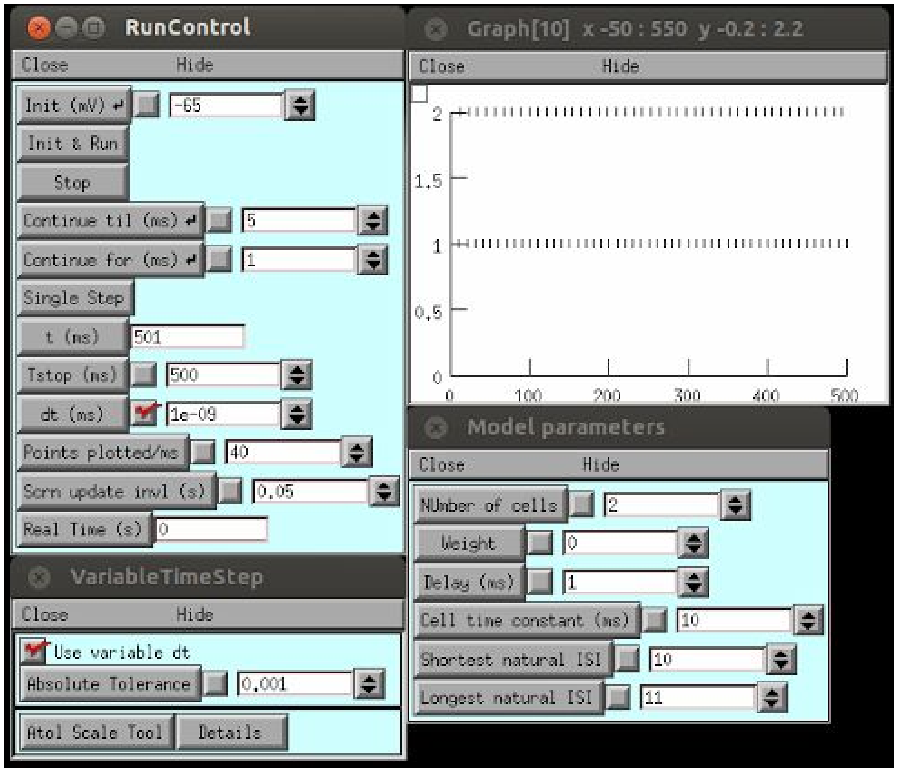
  - Cells are appended to a List call **cells**; connections are appended to a list called **nclist**
  - cell_append(*cell_class*, *x*, *y*, *z*)
  - nc_append(*pre-cell_index*, *post-cell_index*, *synaptic_mechanism*, *weight*, *delay*). *Synaptic_mechanism* is **-1** for an artificial cell.
  - Protections against out of range values can be made with a function such as:

```
func ge() {
        if ($1 < $2) {
                $1 = $2
        }
return $1
}
```

- RunControl:
  - Can be called in hoc by loading "**runctl.ses**"
- Parameter Control Panel:
  - Use **xpanel()** & **xvalue()** to create and setup panel
- Graphs:
  - Record spike times by constructing a NetCon and use the **record()** function to put the spike times in a vector, then append the vector into a list (p. 335).
  - Create a graph from GUI, steal the session file's **save_window_.view()** statement arguments to locate the positions of the graph to use.

Fully connected (all-to-all) network of spontaneously firing neurons with graded natural frequencies:

ncells = 2



RunControl

Init (mV) -65

Init & Run

Stop

Continue til (ms) 5

Continue for (ms) 1

Single Step

t (ms) 501

Tstop (ms) 500

dt (ms) [✓] 1e-09

Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 0

VariableTimeStep

[✓] Use variable dt

Absolute Tolerance 0.001

Atol Scale Tool Details

Graph[10] x -50 : 550 y -0.2 : 2.2

Model parameters

NUmber of cells 2

Weight 0

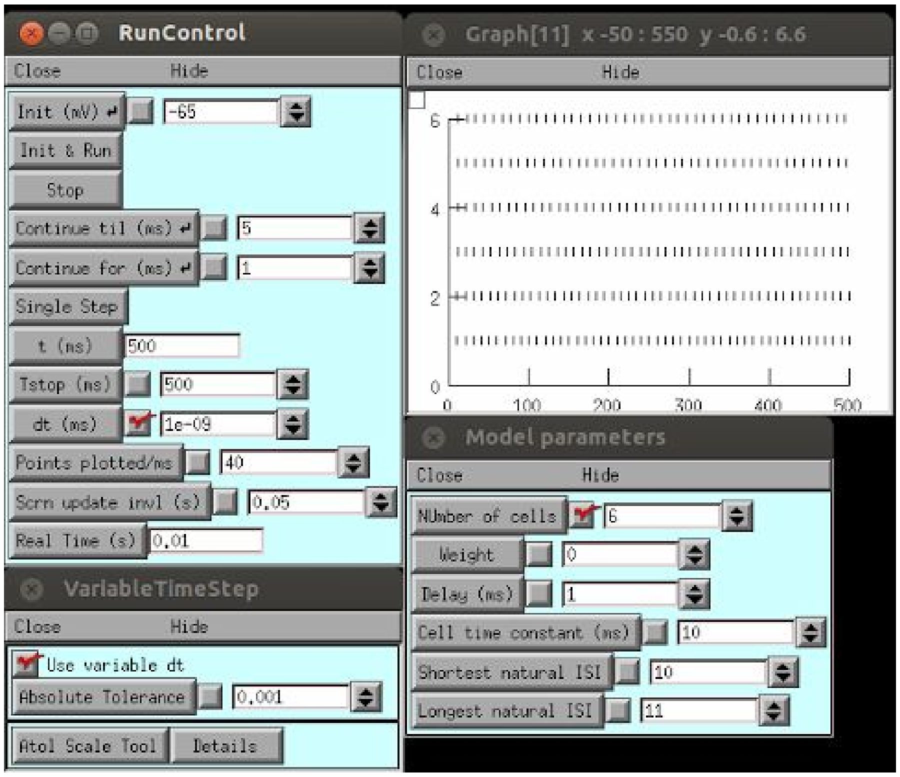
Delay (ms) 1

Cell time constant (ms) 10

Shortest natural ISI 10

Longest natural ISI 11

ncells = 6



RunControl

Init (mV) -65

Init & Run

Stop

Continue til (ms) 5

Continue for (ms) 1

Single Step

t (ms) 500

Tstop (ms) 500

dt (ms) [✓] 1e-09

Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 0.01

VariableTimeStep

[✓] Use variable dt

Absolute Tolerance 0.001

Atol Scale Tool Details

Graph[11] x -50 : 550 y -0.6 : 6.6

Model parameters

NUmber of cells [✓] 6

Weight 0

Delay (ms) 1

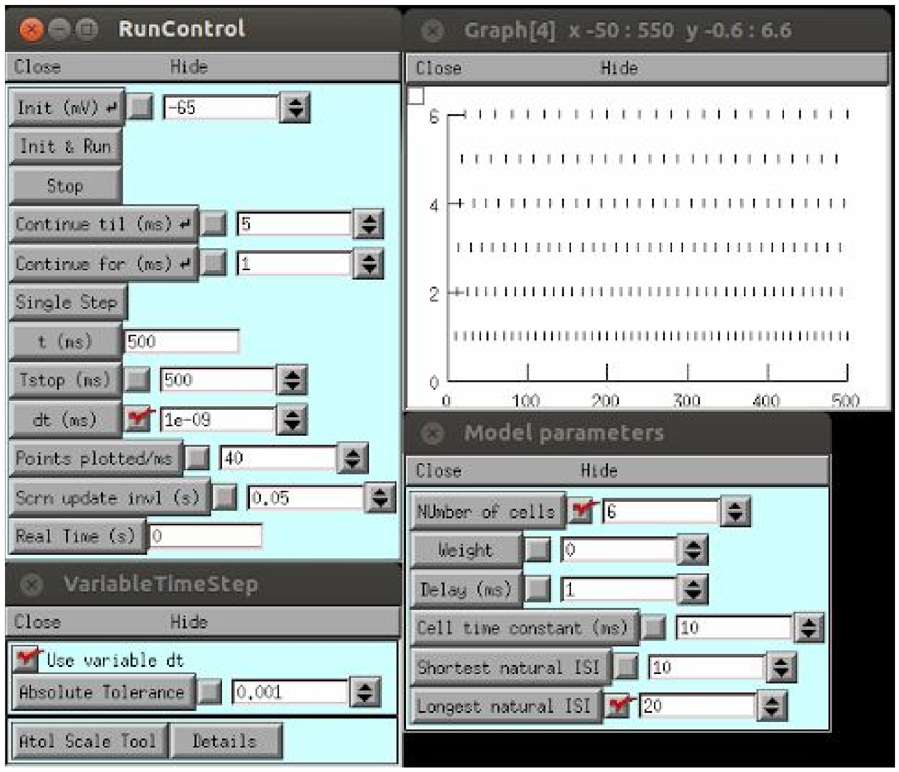
Cell time constant (ms) 10

Shortest natural ISI 10

Longest natural ISI 11

20160508

ncells = 6, larger difference in natural frequency

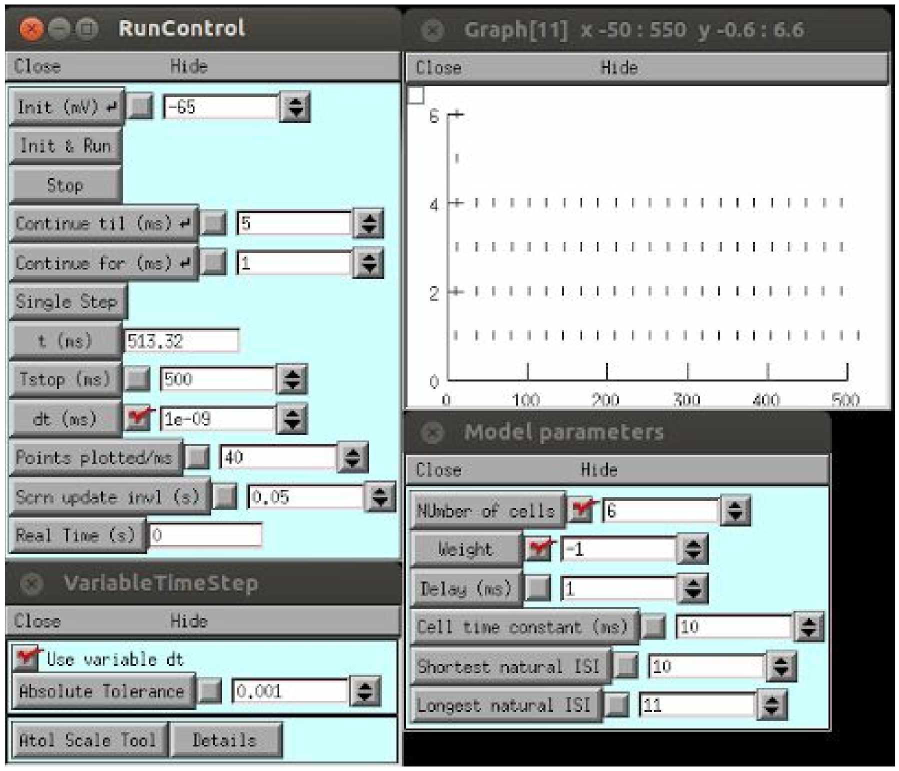


Init (mV) -65
Continue til (ms) 5
Continue for (ms) 1
t (ms) 500
Tstop (ms) 500
dt (ms) ✓ 1e-09
Points plotted/ms 40
Scrn update invl (s) 0.05
Real Time (s) 0

✓ Use variable dt
Absolute Tolerance 0.001

NUmber of cells ✓ 6
Weight 0
Delay (ms) 1
Cell time constant (ms) 10
Shortest natural ISI 10
Longest natural ISI ✓ 20

Mild inhibitory coupling:

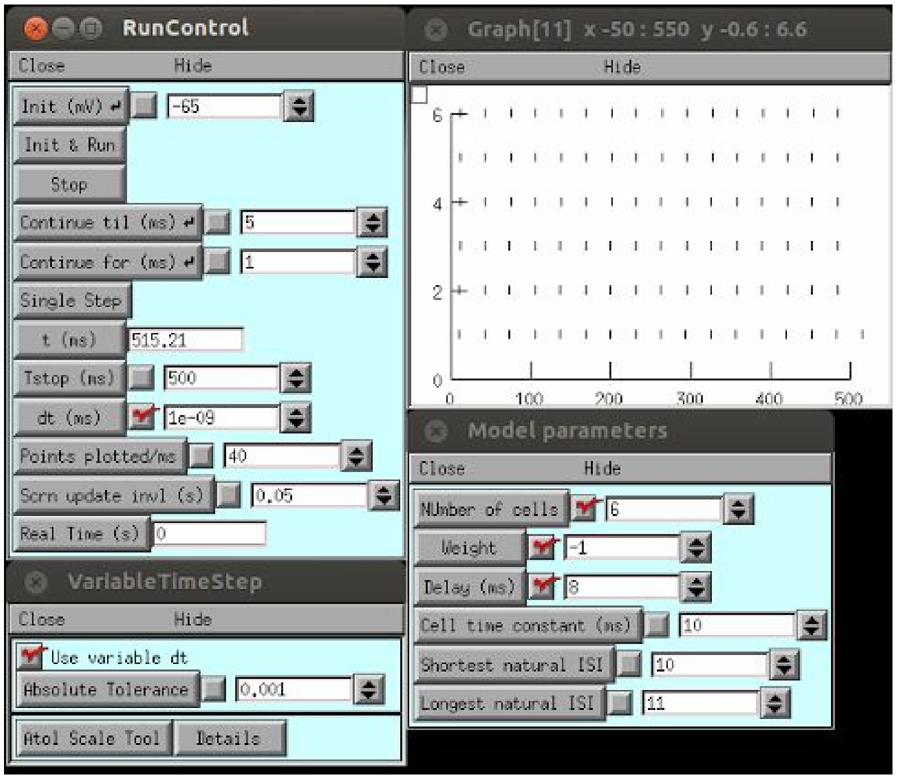


Init (mV) -65
Continue til (ms) 5
Continue for (ms) 1
t (ms) 513.32
Tstop (ms) 500
dt (ms) ✓ 1e-09
Points plotted/ms 40
Scrn update invl (s) 0.05
Real Time (s) 0

✓ Use variable dt
Absolute Tolerance 0.001

NUmber of cells ✓ 6
Weight ✓ -1
Delay (ms) 1
Cell time constant (ms) 10
Shortest natural ISI 10
Longest natural ISI 11

Increasing synaptic delay results in strong correlation



RunControl

Close Hide

Init (mV) -65

Init & Run

Stop

Continue til (ms) 5

Continue for (ms) 1

Single Step

t (ms) 515.21

Tstop (ms) 500

dt (ms) 1e-09

Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 0

VariableTimeStep

Close Hide

Use variable dt

Absolute Tolerance 0.001

Atol Scale Tool Details

Graph[11] x -50 : 550 y -0.6 : 6.6

Close Hide

Model parameters

Close Hide

NUmber of cells 6

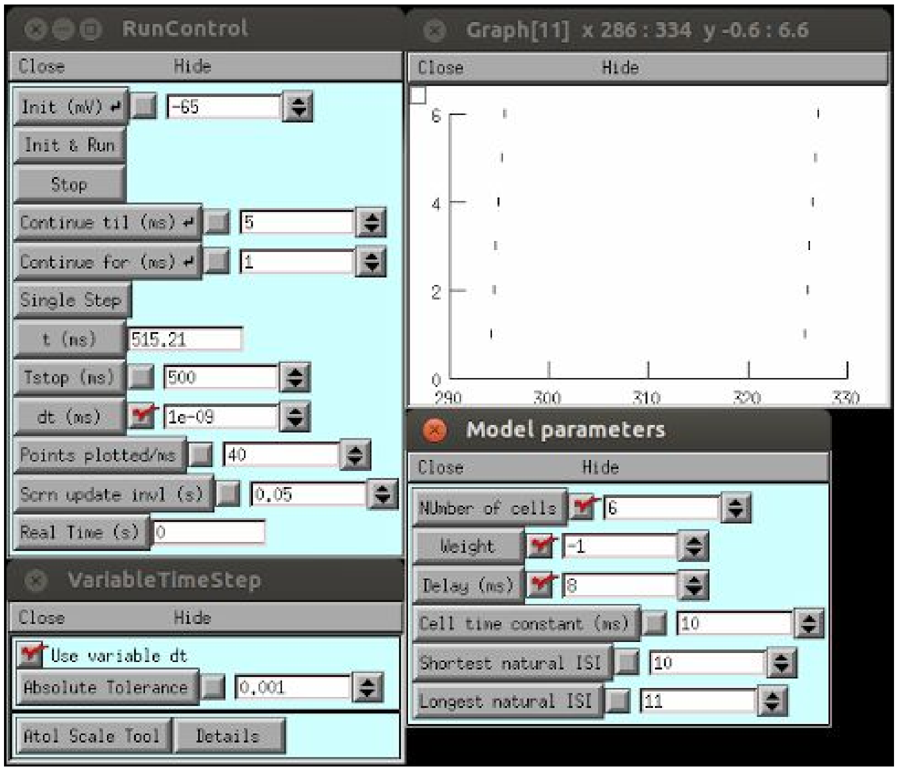
Weight -1

Delay (ms) 8

Cell time constant (ms) 10

Shortest natural ISI 10

Longest natural ISI 11



RunControl

Close Hide

Init (mV) -65

Init & Run

Stop

Continue til (ms) 5

Continue for (ms) 1

Single Step

t (ms) 515.21

Tstop (ms) 500

dt (ms) 1e-09

Points plotted/ms 40

Scrn update invl (s) 0.05

Real Time (s) 0

VariableTimeStep

Close Hide

Use variable dt

Absolute Tolerance 0.001

Atol Scale Tool Details

Graph[11] x 286 : 334 y -0.6 : 6.6

Close Hide

Model parameters

Close Hide

NUmber of cells 6

Weight -1

Delay (ms) 8

Cell time constant (ms) 10

Shortest natural ISI 10

Longest natural ISI 11

5/2/2016

Notes from the NEURON Book Ch 12

- hoc programming language:
  - Based on the **floating point calculator** by the same name that was developed by Kernighan and Pike (1984)
  - hoc has **object-oriented syntax** (supports **information hiding** and **polymorphism**) but **lacks inheritance**
  - The **standard run system** and all **GUI tools** except the Print & File Window Manager (written in C) are written in hoc
  - **Replace** functions and procedures defined in the standard libraries by defining functions or procedures of the same name *after* loading the standard library (**"nrngui.hoc"**).
  - The continuation character "\" can be used to separate statements into multiple lines. However, quoted strings that are constructed with continuation characters have a limit of **256** characters.
- **executables:**

nrniv/nrniv.exe	Main executable Syntax: nrniv [*filenames*] [-] "-" signals that commands are to be taken from standard input until an EOT character (CTRL + D) is encountered
neurondemo	Nrniv + additional mechanisms
nrnivmodl/ mknrndll	Translates NMODL into C by the **nocmodl** translator If no argument: compiles all mod files in the directory

- Error handling:
  - Errors found during parsing are called **parse errors**
  - Errors during interpretation of the **stack machine** are called **run-time errors**
- Cygwin terminal window:
  - Exit hoc by typing **^D** or **quit()**
  - A hoc program can be interrupted by typing one or two **^C** at the terminal.
    One ^C: allows the interpreter to reach a safe place before it halts execution
    Two ^Cs: interrupt the interpreter immediately, even if it is in the middle of updating an internal data structure
- Names:
  - A name is a string that starts with an **alpha character** and contains fewer than **100** alphanumeric characters or the underscore _.
  - Can be: global scalar, local scalar, array, string, function or procedure, template (class or type), object reference
  - Must not conflict with keywords or built-in functions
  - Have **global** scope, except if a **local** declaration is used, or when the name is declared within a template

- Keywords:
  - Listed in src/oc/**hoc_init.c**
  - Those specific to modeling neurons are listed in sec/nrnoc/**neuron.h**
  - Mechanism types and variables are defined in src/nrnoc by **capac.c**, **extcelln.c**, **hh.mod**, and **pas.mod**, etc.
  - Neuron-specific built-in object classes: SectionList, SectionRef, Shape; generic: List, Graph, HBox, File, Random, Vector
  - hoc keywords cannot be redefined (see p. 369)
- Variables:
  - Scalars do not need to be declared. Assignment expressions define the double precision variables.
  - Built-in variables that should be treated as constants:

FARADAY	coulombs/mole	
R	molar gas constant, joules/mole/deg-K	
DEG	180/PI, i.e., degrees per radian	
E	base of natural logarithms	
GAMMA	Euler constant	
PHI	golden ratio	
PI	circular transcendental number	
float_epsilon	resolution for logical comparisons and int()	10^-11

  - Arrays need to be declared:

    double vector[10], array[5][6], cube[first][second][third]

    Array elements are initialized to **0**

    Array indices are truncated to integers and run from **0 to n-1**; if an array name is used without an index, the index is assumed to be **0**.

    Arrays can be **dynamically re-dimensioned** within procedures.
  - String variables need to be declared:

    strdef st1, st2

    No operations (such as addition) are available for strings
  - After a name is defined as a scalar, string or array, it cannot be changed to another type.
  - Names must originally have been declared outside any **func** or **proc** before they can be redeclared in a procedure
- Expressions:
  - Arithmetic operations like in C, except that

    (-1)%5 == 4 instead of -1
- Statements:

    - Statement vs expression:

      a = 4 vs (a = 4)
    - An expression is treated as a statement when it is within a compound statement.
- Comments:
    - ***/* */*** (multiple line) or ***//*** (single line)
- Flow control:
    - These are similar to C:

      if (*expr*) *stmt*
      if (*expr*) *stmt* else *stmt2*
      while (*expr*) *stmt*
      for (*stmt1*; *expr2*; *stmt3*) *stmt*
    - The for loop has the following short form:

      for *var* = *expr1*, *expr2 stmt*

      Here, the increment can only be **1**, and *expr2* must be greater than *expr1*
    - Iteration over a set of items with nontrivial mapping can be performed with this form of the for loop:

      for *iterator_name*(. . .) *stmt*

      An example is the iterator **case** (see pp.356-357), as defined in **stdlib**:

```
x = 1
iterator case() { local i
        for i = 2, numarg() {
                $&1 = $i
                iterator_statement
        }
}
for case(&x, 1, -1, 3, 25, -3) print x
```

    - Operations to alter flow control:

break	Exit from the enclosing while or for loop
continue	Jump to **end** of the enclosing while or for
return	Exit from the enclosing procedure
return *expr*	Exit from the enclosing function
stop	Exit to the top level of the interpreter
quit()	Exit from the interpreter

- Functions and procedures:
    - Arguments:
        - Start with **$**, followed by an optional **&** (the “pointer operator”) to refer to a scalar pointer, followed by an optional **s** or **o** that signifies a string or object reference, followed by an integer.
        - **numarg()** is a built-in function that returns the number of arguments

        - Symbolic positional syntax: the integer could also be a variable **i** ∈ [1, **numarg()**]). Must be **$i** ($j, $k, etc. will not work). i must be declared **local**.
    - Call by reference:
        - If the argument is a scalar, **$1** calls the variable by value and **$&1** calls the variable by reference.
        - If the argument is a one-dimensional array (a double), **$&1** referes to its first element and **$&1[j-1]** refers to its jth element. Warning: there is no array bounds checking
        - A **scalar** or **array** reference may be passed to another procedure with **&$&1**.
        - Arguments of type **strdef** and **objref** are automatically called by reference
- Input and Output:
    - Dealing with multiple files require use of the **File** class
    - Standard input: **read()**; standard output: **print** (comma-separated list of arguments that may be strings or variables), **printf()** (syntax as in C)
    - **sprint()** is useful for building file names
    - **fprintf()** prints to a file opened by **wopen("*filename*")** and closed by **wopen()** or **wopen("")**. If no file is opened, fprintf() prints to standard output
    - **fscan()** reads from a file opened by **ropen("*filename*")** and closed by **ropen()** or **ropen("")**. If no file is opened, fscan() reads from standard input
    - **getstr(*strvar*)** reads the next line from the file opened by ropen() and assigns it to the string variable argument. The trailing newline character is part of the string.
    - **xred("*prompt*", *default*, *min*, *max*)** places a prompt on the standard error device along with the default value. If a newline is typed, the default value is returned; if a number is typed, it is returned only if it is in the range [min, max].
    - **xopen("filename")** or **load_file("filename")** reads in and executes the file
- Editing:
    - The **em** command invokes a public domain editor that is similar to MicroEMACS (see Appendix A2 for details)

5/3/2016~5/4/2016

Notes from the NEURON Book Ch 13 & 14

- Objects and references:
    - Hoc manipulates references to objects (pointers), not the objects themselves. So ob1 = ob2 means that ob1 refers to the same object as ob2. The **reference count** of an object is the number of object references that point to it.
    - Object references are declared by

      objref *name1*, *name2*, *name3*, …

      Initially, these refers to the **NULLobject**
    - And object is created by

      objref g

      g = **new** Graph()
    - If a reference count of an object becomes 0, it is destroyed and the memory that held its data becomes available for any other purpose
    - Public members of an object is accessed by the "dot" notation:

      g.erase()
    - Object names are defined as ***classname*****[index]**, where the "index" is automatically incremented every time a new instance of that class is created. Index numbers are not reused after objects are deleted except when there are no existing objects of that type (in which case it starts over again at 0). To find the object name, use:

      print g

      However, one should not use these names in user-written code, because they are not guaranteed to be the same between different NEURON sessions.
    - When **this** is declared in a template as an object reference, it always refers to the instance of the template that declared it.
- Classes:
    - Classes are defined by enclosing functions, procedures and variables with the keywords **begintemplate** and **endtemplate**.
    - After the hoc interpreter has parsed the code in a template, the class that it defines is fixed for that session. Any changes require restarting NEURON.
    - Syntax:

      begintemplate *classname*

      public *name1*, *name2*, *name3*, . . .

      external *variable1*, *string2*, *function3*, *template4*, . . .

      . . . hoc code . . .

      endtemplate *classname*
    - A function or procedure that is defined in a class is called a **method**.
    - Any user-defined global variables and functions must appear in the **external** statement.
    - To make something visible from the outside, it must be declared **public**.

  - **Direct commands** (such as a = 0) are only executed **once by hoc** (not for each object).
  - All variables start off with a default value of **0**. To initialize variables, the template must contain an **init()** procedure, which is automatically executed every time a new object is created. If init() appear in the public list, it can be executed explicitly as well.
- Arrays:
  - Most efficient but requires a prior knowledge of size. Declared with

    objref *array*[*size*]

    Initially, all elements reference the **NULLobject**
  - Size can only be changed by redeclaring the entire array
  - An array is a **random access object**
  - To destroy the *k*th element of an array, use

    ```
    objref nil          // nil points to NULLobject
    Array[k] = nil   // and now so does Array[k]
    ```
  - An array of strings can be implemented by an array of **String** objects (see p. 373).
- List():
  - A list can store any number of objects at any time
  - Member functions:

append()	Add objects to the list
count()	Returns the number of objects in a list
object(*i*)	Returns the *i*th item of the list
remove(*i*)	Removes the *i*th item from the list

- Graph():
  - Manages a window where x-y plots can be drawn
  - Member functions:

erase_all()	Erase everything on the graph
size(1-4**)**	Returns left, right, top or bottom of first view of the scene
beginline()	The next line() is the first point of the next line to be graphed
line(*x, y*)	Draw a line from the previous point to this point
mark(*x, y, "style", size*)	Make a mark at the indicated position which does not change size when the window is zoomed or resized
flush()	Actually draw what has been placed in the graph scene
view_info()	Return information about the view

menu_tool("*label*", "*procedure_name*")	Add a selectable tool menu item to the Graph popup menu or else, if an xpanel is open, an xradiobutton will be added to the panel having the same action.
exec_menu("*item_name*")	Equivalent to by pressing and releasing one of the items in the Graph menu with the right mouse button

- Vbox()/Hbox():
    - A collection of graphs and command panels with the windows tiled vertically/horizontally
    - Member functions:

ref()	assigns an object to be referenced to the box
intercept(1)	all following window creations go into the box
intercept(0)	end intercept mode
map()	makes the box appear on the screen

    - Use **box.ref(this)** to ensure that the reference count of a tool is decremented when the window is closed
- Encapsulating code:
    - One can encapsulate an entire hoc file with a run statement into a class:
```
begintemplate F1
public run
. . . hoc code
endtemplate F1

objref f1
f1 = new F1()
f1.run()
```
      then all variables and function names used are local, so they won't clash with other files
    - Caveat is that all direct commands must be placed in an **init()** procedure.
- Polymorphism: Functions of the same name under different classes are not confused

<u>Plan for next week</u>

1. Prepare PPT for Kandel and Buzsaki
2. Continue examining the codes for the Destexhe et al 1998 model
3. Examine the codes for the Destexhe et al 1996 model
4. Go through the NEURON Hands-On Course

<u>Plan for the future</u>

1. Continue compiling relevant papers that have used the Destexhe et al 1998 model
2. Compile relevant papers that have used the Destexhe et al 1996 model
3. “Reproduce CSD graph” exercise
4. Examine Christine’s & Mark’s codes
5. Finish NEURON Book Appendix A1
6. Figure out how to export NEURON to Matlab
7. Complete the **NEURON Tutorial**
8. Understand NEURON Ch 7 & Ch 8
9. Resolve all NEURON Book questions
10. Read Abbott et al 2016 (“Building functional networks of spiking model neurons”)
11. Read Markram et al 2015 (“Reconstruction and Simulation of Neocortical Microcircuitry”)
12. Read Kragel & LaBar 2016 (“Decoding the Nature of Emotion in the Brain”)
13. Read Izhikevich: Dynamical Systems in Neuroscience
14. Read Dayan & Abbott: Theoretical Neuroscience

4/22/2016~5/1/2016

Notes from the NEURON Book Ch 9 & 10

- NMODL is a descendant of the MOdel Description Language (MODL) developed at Duke for the Simulation Control Program (SCoP)
- Mechanisms automatically appear with the other distributed mechanisms in GUI tools such as the **Distributed Mechanism Inserter**
- **Variable declaration blocks**: PARAMETER, STATE, ASSIGNED
- **Equation definition blocks**: INITIAL, BREAKPOINT, DERIVATIVE, KINETIC, FUNCTION, PROCEDURE
- Comments
  - Single line: Use "**:**" at the beginning of each line
  - Multiple lines:

    COMMENT

      This is
      a multiple
      line comment

    ENDCOMMENT"
- Embed C code

    VERBATIM

      /* C statements */

    ENDVERBATIM
- Named blocks: *KEYWORD* **{** *statements* **}**
- Variables:
  - User defined variable names can be up to **20 characters** long
  - Variables must be defined before used
  - Variables available to all mechanisms:
    - **v** [millivolts]
    - **celsius** [℃] (default is 6.3℃)
    - **t** [milliseconds]
    - **diam** [μm]
    - **area** [μm^2]
    - **dt** (avoid using when CVODE is used)
- The **NEURON** block:
  - The NEURON block specifications are independent of the simulator. NMODL outputs a C file for NEURON, whereas the GMODL translator outputs a C file for GENESIS
  - Variables declared here (either as RANGE or GLOBAL) can be accessed by the **hoc interpreter**. Variables available to all mechanisms (see above) need not be declared here.
  - Variables can be **arrays**, but NMODL arrays are not dynamic (the array length is set when NMODL is translated into C code)

- To run through indices of arrays, use
  FROM i=0 **TO** end { … }
- **nocmodl** (not nmodl) is the current NMODL translator
- **SUFFIX**:
  - Identifies a **distributed mechanism**
  - Can be incorporated into a NEURON cable section with **insert**
  - Variables & parameters that belong to this mechanism will include the corresponding suffix (e.g. **_leak**)
- **POINT_PROCESS**:
  - Identifies a **point process**
  - Managed in hoc using an **object-oriented syntax**
- **POINTER**:
  - For point processes and distributed mechanisms. A POINTER variable holds a reference to another variable. The specific reference is defined by a hoc statement:
    setpointer *point_process.pointer_var*, *precell.segment.var*(0.5)
- **NONSPECIFIC_CURRENT**:
  - The value will be reckoned in charge balance equations
  - The current will make no direct contribution to mass balance equations
- **ELECTRODE_CURRENT**:
  - Positive values lead to depolarization
  - When the **extracellular** mechanism is present, there will be a change in the extracellular potential **vext**
- **USEION**:
  - A separate USEION statement is needed for each of the ions involved
  - Creates or uses a mechanism *x***_ion** that has range variables **i***x*, *x***i**, *x***o** and **e***x* (units required to be in **[mM]**, p. 251), which represents I_*x*, [*x*]_i, [*x*]_o & E_*x*, respectively. The initial values of *x***i** and *x***o** are set globally to the values of *x*i0_*x***_ion** & *x*o0_*x***_ion**, respectively.
  - **READ e*x***
  - **WRITE i*x*** enables NEURON to keep track of the total outward current, the internal and external concentration and the equilibrium potential for a particular ion *x*
  - **WRITE *x*o** will set the value for the concentration all locations in a section with the mechanism. Thus in any given section, no ionic concentration should be “written” by more than one mechanism.
- **RANGE**:
  - Declares a variable as a **range variable** (a function of position)
  - Each variable mentioned here should also be declared in a PARAMETER or ASSIGNED block
  - Opposite: GLOBAL
- **GLOBAL**:
  - Declares a variable as a **global variable**

- Variable declaration blocks
  - Any user-defined variable must be declared here, even if it had appeared in the NEURON block already
  - To facilitate **units checking**, each variable declaration includes a specification (defined in **nrnunits.lib**, which is based on the UNIX units database) of units in parentheses. If units are not declared, a **units factor** must be applied in equations elsewhere:

    g = *var_no_units* ***1 (umho)**
  - Variables defined by NEURON and currents, concentrations & equilibrium potentials created by the **USEION** statement have their own particular units
- The **UNITS** block:
  - Redefine units:

    (nA) = (nanoamp)
  - Define constants:

    FARADAY = (faraday) (kilocoulombs)

    Here (faraday) takes the value from **nrnunits.lib** & (kilocoulombs) rescales its unit to kilocoulombs/mole
  - **e** is treated as the electronic charge in the UNITS block; elsewhere a single number in parentheses is treated as a units conversion factor e.g., (2e4)
- The **PARAMETER** block:
  - Assign **default values**
  - Angle brackets **<>** specifies the minimum and maximum limits that can be entered into the **field editor** of the GUI
  - PARAMETERs are by default **global variables** that can be seen by hoc and do not need to be declared in the NEURON block
  - If a parameter must take different values in different segments, it has to appear in NEURON->**RANGE**
- The **CONSTANT** block:
  - CONSTANT variables are never changed in the course of a simulation (like PARAMETERs) but cannot be accessed by hoc
- The **ASSIGNED** block:
  - Includes variables that are given values outside the mod file & variables that appear on the left hand side of assignment statements
  - Excludes state variables & dependent variables in differential equations
  - By default, an assigned variable is a **range variable** (not global)
  - Good practice to declare **v** here for units checking
- The **STATE** block:
  - STATE variables are dependent variables or unknowns in **differential equations**, **families of algebraic equations**, or **kinetic reaction schemes**
  - STATE variables are by default **range variables** that can be seen by hoc and do not need to be declared in the NEURON block
  - The **membrane potential** is never a STATE variable because its value is calculated by NEURON and never by NMODL code

  - A STATE variable *state* has an implicitly declared parameter called ***state*0**, whose default value can be specified either in the PARAMETER block:

    *state*0 = 1

    or in the STATE block

    state **START** 1

    If initial value is not declared, the default value is **0**.
  - **Local absolute error tolerance** employed by CVODE can be specified with

    state **(***units***) <***absolute_tolerance***>**

    Default value of *absolute_tolerance* is **10^-3**
  - The CVODE solver tries to use a step size for which the local error ϵ_i of each state_i satisfies etiher

    ϵ_i < *relative_tolerance* * |state_i|

    Or

    ϵ_i < *absolute_tolerance*

    Default value of *relative_tolerance* is 0*.*
- **LOCAL** variables declared outside of equation definition block:
  - Equivalent to a **static** variable in C (visible throughout the mechanism, but not at the hoc level).
  - Initial value is **0**.
- Equation definition blocks:
  - **UNITSOFF … UNITSON** disables unit checking in … .
  - **LOCAL** declares a local variable within a block. Values are not retained between invocations of the block. Takes on units of right hand side.
- The **BREAKPOINT** block:
  - Main computational block
  - The independent variable in NEURON is always time t
  - Neither **t** nor the time step **dt** should be changed in NMODL
  - Put any conversion factor with a unit (that is not 1) in **parentheses**
  - Units can be checked with the **modlunit** command:

    modlunit shunt.mod
  - **at_time(*time*)** guarantees a time step boundary just before *time*.
  - The **SOLVE** statement specifies a block of code that defines the simultaneous equations that govern the STATEs
  - The assignment statements in a BREAKPOINT block are usually called **twice per time step**
  - Computations that must be performed only **once per time step** should be placed in a **PROCEDURE**, which in turn would be invoked by a SOLVE statement
  - **METHOD** specifies the method of integration for the SOLVE statement. If the equation is in the for y' = *f*(v,y) with *f* linear in y, **cnexp** is a good method (second-order accuracy); if *f* is nonlinear in y, **derivimplicit** should be used (first-order accuracy). For variable time step methods, both the **time step** and the **numerical integration formula** (accuracy ranging from first to fifth order) are changed adaptively. Currently, NMODL creates a **diagonal Jacobian**

  approximation (exact if *f* is a polynomial) by **numerical differencing**. The user may also supply an explicit Jacobian.
  - The **sparse** method is generally faster than computing the full Jacobian matrix. It advance STATEs with a fully implicit method (such as derivimplicit, first-order correct).
  - DERIVATIVE blocks can be solved by cnexp, but KINETIC blocks should be solved by sparse.
- The **INITIAL** block:
  - Contains formulas for initialization of **STATE variables**.
  - For a kinetic scheme, STATE variables can be initialized with **STEADYSTATE**, as long as a **CONSERVE** statement is available in the **KINETIC** block to ensure that the equivalent system of ODEs would be linearly independent. This is convenient for mechanisms whose steady state solutions are difficult or impossible to express in analytical form:

    SOLVE *kinetic_scheme* STEADYSTATE *method*
  - Clarity will be served if in the INITIAL block has *state* = *state*0
  - **NET_RECEIVE** blocks may also have its own INITIAL block for nonzero initialization of **NetCon** states.
  - Executed when the standard run system's **finitialize()** is called.
  - Alternative: Initialize states with a preliminary **initialization run**, where t is assigned a large negative value and then advanced over several large time steps.
- The **DERIVATIVE** block:
  - Used to assign values to the derivatives of those STATE variables that are described by differential equations.
  - Form: y' = *expr*
  - Called by the SOLVE statement in the BREAKPOINT block
- The **NET_RECEIVE** block:
  - All **discontinuities** should be handled here.
  - Only called when an event has been delivered.
  - Arguments (a weight vector that is created by a NetCon) are **call by reference**
  - Example:

    ```
    NET_RECEIVE(weight (microsiemens)) {
            a = a + weight*exp(1)
    }
    ```
  - **flag** is an implicit argument of NET_RECEIVE that is call by value and is automatically **0** for an **external event**
  - Use **net_send(Cdur, 1)** to **send a self-event** with flag==1 after Cdur. All the explicit arguments of this self-event will have the values of this particular NetCon.
  - Use **net_move(t + Cdur)** to **move the self-event** to time t+Cdur
- The **FUNCTION** block:
  - Used to assign values to the derivatives of those STATE variables that are described by differential equations.

- Available at the hoc level by adding the **suffix**. If **range variables** are referenced, it is necessary to specify its location on the cell:

  section_name **setdata_***suffix*(x)
- Arguments are **call by value**

- The **PROCEDURE** block:
  - These are like FUNCTIONs but with no return value
  - Arguments are **call by value**
- The **KINETIC** block:
  - The kinetic scheme

    ~ mc **<->** m (a(v), b(v))

    Is equivalent to the differential equations

    mc' = -a(v)*mc + b(v)*m

    m' = a(v)*mc - b(v)*m

    where a(v) & b(v) are the forward and reverse **reaction rates**, respectively
  - **<->** indicates a **reaction**; **->** indicates a **sink reaction**; **<<** indicates **explicit flux**
  - Voltage-sensitive rates are allowed, but to guarantee numerical stability, reaction rates should *not* be functions of STATEs.
  - The forward & reverse **fluxes** are automatically recorded in the variables **f_flux** & **b_flux**, respectively
  - If a reactant is not declared as a STATE variable, it is treated as a constant (no differential equation).
  - With a **CONSERVE** statement, the NMODL translator will replace the ODE for the last STATE on the left side of the equal sign with the conservation equation. This statement is necessary when STATEs are initialized with **STEADYSTATE**
  - The **COMPARTMENT** statement can correct dimensional disparities by specifying volumes for STATE variables:

    COMPARTMENT ***volume*** { *state1 state2* . . . }

    Or in the case where the STATE variables are arrays,

    COMPARTMENT *index***,** ***volume*****[***index***]** { *state1 state2* . . . }
  - The **LONGITUDINAL_DIFFUSION** statement specifies that this mechanism includes “nonlocal” diffusion (longitudinal diffusion along a section and into connecting sections):

    LONGITUDINAL_DIFFUSION ***flux_expr*** { *state1 state2* . . . }

    Or in the case where the STATE variables are arrays,

    COMPARTMENT *index***,** ***flux_expr*****[***index***]** { *state1 state2* . . . }
  - A **LOCAL** statement cannot be defined in a KINETIC block and the variables cannot be used in a COMPARTMENT statement
- The **FUNCTION_TABLEs**:
  - Functions in table form that must be provided by the hoc commands:

    table_*function_mechanism*(*y_vec*, *x_vec*)
  - Prior to developing the Vector database, these functions can be attached a constant value:

    table_*function_mechanism*(*value*)

    - Can be declared with two arguments and attached to **doubly dimensioned hoc arrays** (linear interpolation in both dimensions)
- Usage:
    - For **distributed mechanisms**:

        insert leak
        g_leak = ...
    - For **point processes**:

        objref s
        cable s = new Shunt(0.1)
        s.r = ...
- Synapses - the **NetCon** class:
    - Usage:

        section netcon = new **NetCon**(*&v(x)*, *target*, *threshold*, *delay*, *weight*)

      where *&v(x)* is the source variable. Threshold, delay & weight are optional.
    - Default values:

        netcon.threshold = **10** // mV
        netcon.delay = **1** // ms
        netcon.weight = **0** // uS
    - Multiple targets can be connected from the same source variable, and they share a **single threshold detector**.
    - Multiple NetCons can share a single postsynaptic mechanism
- Artificial spiking cells:
    - **Discrete event simulation** is possible when all the state variables of a model cell can be **computed analytically** from a new set of initial conditions.
    - Implemented in NEURON as point processes that can serve as both a target and a source.
    - **ARTIFICIAL_CELL** is a synonym for POINT_PROCESS, but usually contains a NET_RECEIVE block, lacks a BREAKPOINT block, and is not associated with a section location or numerical integrator (it does not refer to v or any ions or have a POINTER). It is entirely isolated and depends on discrete events from the outside to affect it and affects the outside only by sending discrete events
    - **net_event(t)** notifies all NetCons with this point process as a source that it fired a spike at time t (the argument can be any time at or later than the current time t)
    - The event delivery system only places the **earliest event** to be delivered on the event queue.
    - **IntFire1** (nrn/src/nrnoc/intfire1.mod) has a membrane state variable ***m*** (initial value is 0) that decays toward **0** with time constant $\boldsymbol{\tau}$ (default is **10 ms**):

      $$\tau\frac{dm}{dt} + m = 0$$

      and has an absolute refractory period ***refrac*** (default is **5 ms**). An input event causes *m* to increase by weight *w*. The neuron fires (net_event(t)) when *m* > 1. **M()** is a function that approximates the membrane potential at that point in time.
    - **IntFire2** (nrn/src/nrnoc/intfire2.mod) has a membrane state variable ***m*** (initial value is 0) and a current state variable ***i*** (initial value is 0).

An input event causes *i* to increase by weight *w*. Between events, *i* decays toward a steady state **bias current** *i_b* (default is 0 mA) with time constant ***τ*_s** (default is **20 ms**):

$$\tau_s \frac{di}{dt} + i = i_b$$

and *m* is driven by ***i*** with time constant ***τ*_m** (default is **10 ms**, should be always smaller than *τ*_s):

$$\tau_m \frac{dm}{dt} + m = i$$

The neuron fires (net_event(t)) when *m* > 1. *m* is reset to 0, but not *i*. The firing rate is about **i/*τ*_m**

firetime() returns the first t >=0 for which m(t) = 1, if this will never happen without further external events, it returns **10^9**.

An firing event is called upon initiation for firetime() == 10^9, but will be moved accordingly afterwards.

- **IntFire4** (nrn/src/nrnoc/intfire4.mod) deals with the case where excitation responds faster than inhibition. There are 4 time constants: ***τ*_e < *τ*_i1 < *τ*_i2 < *τ*_m**. See p.301 for equations.
  It exploits the **downward convexity** of the membrane potential trajectory to approximate the next firing time by a **slope approximation**, resulting in an alternating sequence of self-events and single Newton iterations
  A global parameter **eps** is used to guard against missing firings when *m* is asymptotic to 1 (the threshold is **1 - eps**)

Plan for next week

1. Finish NEURON Book Ch 11, Ch 12, Ch 13, Ch 14, Appendix A1, last part of Ch 8
2. Read Kandel and Buzsaki, prepare PPT
3. Continue examining the codes for the Destexhe et al 1998 model
4. Examine the codes for the Destexhe et al 1996 model

Plan for the future

1. Continue compiling relevant papers that have used the Destexhe et al 1998 model
2. Compile relevant papers that have used the Destexhe et al 1996 model
3. Figure out how to export NEURON to Matlab
4. Go through the NEURON Hands-On Course
5. Resolve all NEURON Book questions
6. Read Izhikevich: Dynamical Systems in Neuroscience
7. Read Dayan & Abbott: Theoretical Neuroscience

4/18/2016~4/20/2016 (last modified 5/16/2016)

Relevant papers that have cited Destexhe et al 1998a and/or have used the Destexhe et al 1998 model

- Web of Science showed **201** papers that cited Destexhe et al 1998.
- Pubmed showed **70** papers that cited Destexhe et al 1998.
- The following actually used the NEURON codes:

Author	Year	Title	Description
Connelly et al	2015	The Global Spike: Conserved Dendritic Properties Enable Unique **Ca^{2+} Spike Generation** in Low-Threshold Spiking Neurons	• Parameters for T-type Ca^{2+} channels & fast voltage-gated sodium channels (g_{Na}) were taken from Destexhe et al. (1998) • The gating of the T-type Ca^{2+} channel was shifted by 12 mV in the hyperpolarizing direction to fit the data • Other aspects of the model were taken from other papers • "LTS have remarkably similar amplitudes and occur **synchronously** throughout the dendritic tree...LTS are generated by a **unique whole-cell mechanism** that means they always occur as spatially global spikes"
Forrest, MD	2015	Simulation of alcohol action upon a detailed **Purkinje neuron** model and a simpler surrogate model that runs >400 times faster	• Uses a similar method as Destexhe et al. (1998) to reduce a Purkinje neuron without losing its properties • "Alcohol may modulate Purkinje neuron firing by an inhibition of their **sodium-potassium pumps**."
Park et al	2014	Roles of **GABAA and GABAB** receptors in regulating thalamic activity by the zona incerta: a computational study	• The model of the posterior thalamic nucleus (PO) neuron was based on Destexhe et al. (1998) • GABA inputs to PO come from the **zona incerta (ZI)** • "**spontaneous** PO activity is preferentially regulated by **GABABR**-mediated mechanisms, while **evoked** activity is preferentially regulated by **GABAAR**"
Kent et al	2014	Analysis of **deep brain stimulation electrode**	• Applied **closed-loop deep brain stimulation (DBS)** systems to the neuron model of Destexhe et al. (1998)

		characteristics for neural recording	
Amarillo et al	2014	The interplay of seven subthreshold conductances controls the resting membrane potential and the **oscillatory behavior** of thalamocortical neurons	• The model neuron was based on Destexhe et al. (1998) • "The balance between three amplifying variables (activation of **IT**, activation of **INaP**, and activation of **IKir**) and three recovering variables (inactivation of **IT**, activation of **IA**, and activation of **Ih**) determines the propensity, or lack thereof, of **repetitive burst firing** of TC neurons."
Drion et al	2012	A novel **phase portrait** for neuronal excitability	• **Phase portrait analysis** of the Hodgkin-Huxley Model with the addition of a calcium current • Also did analysis on the TC model of Destexhe et al. (1998)
Keane et al	2012	Improved spatial targeting with directionally segmented **deep brain stimulation** leads for treating essential tremor	• "Multi-compartment neuron models of the thalamocortical, cerebellothalamic and medial lemniscal pathways were first simulated in the context of **patient-specific anatomies**, lead placements and programming parameters from three ET patients who had been implanted with Medtronic 3389 DBS leads." (Neurons not connected?) • The addition of **directionally segmented electrodes (dDBS)** to deep brain stimulation (DBS) @ the ventral intermediate nucleus of thalamus (Vim) should prevent the persistent paresthesias associated with activating the ventral caudal (Vc) nucleus
Birdno et al	2012	Stimulus features underlying reduced tremor suppression with temporally patterned **deep brain stimulation**	• Uses a biophysical model of the thalamic **network** that include elements of Destexhe et al. (1998)

Rabang and Bartlett	2011	A computational model of cellular mechanisms of temporal coding in the **medial geniculate body (MGB)**	• The model neuron was adapted from Destexhe et al. (1998)
Wei et al	2011	Thalamic burst firing propensity: a comparison of the **dorsal lateral geniculate** and **pulvinar** nuclei in the tree shrew	• The model neuron was adapted from Destexhe et al. (1998)
Tscherter et al	2011	Minimal alterations in T-type calcium channel gating markedly modify physiological firing dynamics	• The T currents used for dynamic clamp studies were adapted from Destexhe et al. (1998)
Steuber et al	2010	Determinants of synaptic integration and heterogeneity in rebound firing explored with data-driven models of **deep cerebellar nucleus** cells	• A model of a deep cerebellar nucleus neuron • The T currents used was adapted from Destexhe et al. (1998)
Zomorrodi et al	2008	Modeling thalamocortical cell: impact of **ca channel distribution** and **cell geometry** on firing pattern	• The model neuron was adapted from Destexhe et al. (1998)
Chemin et al	2002	Specific contribution of human T-type calcium channel isotypes (alpha(1G), alpha(1H) and alpha(1I)) to neuronal excitability	• The model neuron was adapted from Destexhe et al. (1998) • “Using simulations of reticular and relay thalamic neuron activities, we show that **alpha(1I)** currents contributed to sustained electrical activities, while **alpha(1G)** and **alpha(1H)** currents generated short burst firing.”

- The following didn't use the NEURON codes but explored the model or related equations further

Author	Year	Title	Description
David et al	2016	Dynamic Analysis of the Conditional Oscillator Underlying **Slow Waves** in Thalamocortical Neurons	• Bifurcation analysis • "Although stable delta oscillations can be evoked with minimal T conductance, the full range of slow oscillation patterns, including **groups of delta oscillations separated by Up states** ("grouped-delta slow waves") requires a **high density** of T channels."
Amarillo et al	2015	Analysis of the role of the **low threshold currents IT and Ih** in intrinsic **delta oscillations** of thalamocortical neurons	• Bifurcation analysis and phase plane portrait analysis performed using XPPAUT • "The interplay between the amplifying variable mT and the recovering variable hT of the calcium channel IT is sufficient to generate **low threshold oscillations** in the delta band"
Birdno et al	2014	Response of Human Thalamic Neurons to **High-Frequency Stimulation**	• 24 **3-D reconstructed TC cell models** were used to calculate the combined effects of the intrinsic synaptic inputs and microstimulation on TC neuron activity • "During DBS the **axons** of thalamocortical neurons are activated while the **cell bodies** are inhibited thus blocking the transmission of pathological signals through the network and replacing them with high frequency regular firing"
Franci et al	2013	A Balance Equation Determines a **Switch** in Neuronal Excitability	• Mathematical equation for switch proposed • Bifurcation analysis and phase plane portrait analysis verifies the switch
Lajeunesse et al	2013	Regulation of **AMPA and NMDA** receptor-mediated EPSPs in dendritic trees of thalamocortical cells	• A multicompartment model based on fully reconstructed TC neurons from the ventroposterolateral nucleus of the cat • "**AMPAR**-mediated responses, when synapses were located at **proximal dendrites**, induced a larger depolarization at the level of soma, whereas **NMDAR**-mediated responses were more efficient for synapses located

			at **distal dendrites**”
Marasco et al	2012	Fast and accurate **low-dimensional reduction** of biophysically detailed neuron models	• Another way to reduce neurons for network modeling
Agarwal & Sarma	2012	**Performance limitations** of relay neurons	• Construct analytic bounds for a biophysically-based model of a relay cell
So et al	2012	Relative contributions of local cell and passing fiber activation and silencing to changes in thalamic fidelity during **deep brain stimulation** and lesioning: a computational modeling study	• Uses a **basal ganglia-thalamic network**
Halnes et al	2011	A multi-compartment model for **interneurons** in the **dorsal lateral geniculate nucleus**	• A biophysically based interneuron model
Crandall et al	2010	Low-Threshold Ca^{2+} Current Amplifies Distal Dendritic Signaling in Thalamic **Reticular** Neurons	• “A single somatic burst discharge evokes large-magnitude calcium responses, via I(T), in distal TRN dendrites” • “Direct stimulation of distal TRN dendrites, via focal glutamate application and synaptic activation, can locally activate distal I(T), producing a large distal calcium response independent of the soma/proximal dendrites” • “distally located I(T) may function to amplify afferent inputs”
Errington et al	2010	State-dependent firing determines intrinsic dendritic Ca2+ signaling in	• “T-type Ca(2+) channels are expressed throughout the entire dendritic tree of rat thalamocortical neurons and that they mediate regenerative propagation of low threshold spikes, typical of, but not

		thalamocortical neurons	exclusive to, sleep states, resulting in global dendritic Ca(2+) influx.”
Ernst et al	2009	Genetic enhancement of thalamocortical network activity by elevating alpha 1g-mediated low-voltage-activated calcium current induces pure absence epilepsy	● Mouse models of absence epilepsy: Two BAC transgenic mouse lines overexpressing the **Cacna1g** gene for **alpha1G T-type calcium channels**
Rhodes & Llinás	2005	A model of thalamocortical relay cells	● A new model of a TC neuron that can produce regular spiking relay and low threshold rebound bursts, as well as **fast oscillations** occurring at near-threshold somatic potentials ● “The model produces the low threshold spike behaviour of the relay cell *without* requiring high T-current density in the distal dendritic segments”

4/21/2016 (last modified 5/15/2016)

Details of Destexhe et al 1998a model (continued)

- Overview of all files

File name	Called by	Procedures included	Notes
mosinit.hoc	None		

rundemo.hoc	mosinit.hoc	destroy_elec()
		restart()	

tc1_cc.oc	rundemo.hoc	add_graph()
		add_shape()
			Burst behavior in single-compartment model

tc3_cc.oc	rundemo.hoc	add_graph()
		add_shape()
			Burst behavior in 3-compartment model

tc200_cc.oc	rundemo.hoc	add_graph()
		add_shape()
			Burst behavior in detailed cell model

tc200_vc.oc	rundemo.hoc	add_graph()
		add_shape()
			Voltage-clamp in detailed cell model

tcD_vc.oc	rundemo.hoc	add_graph()
		add_shape()
			Voltage-clamp in dissociated cell model

cells/**tc1.geo**	tc1_cc.oc		Geometry of single-compartment model
cells/**tc3.geo**	tc3_cc.oc		Geometry of 3-compartment model
cells/**tc200.geo**	tc200_cc.oc tc200_vc.oc		Geometry of detailed cell model
cells/**tcD.geo**	tcD_vc.oc		Geometry of dissociated cell model
el.oc	tc1_cc.oc	**Electrode** makeelectrode()	The class Electrode
loc3.oc	tc3_cc.oc	localize()	localize T-current differentially in soma and dendrites
loc200.oc	tc200_cc.oc tc200_vc.oc	localize()	localize T-current differentially in soma and dendrites
locD.oc	tcD_vc.oc	localize()	localize T-current differentially in soma and dendrites

cadecay.mod	tc1_cc.oc tc3_cc.oc tc200_cc.oc tc200_vc.oc tcD_vc.oc	**cad**	Fast mechanism for submembranal Ca++ concentration
hh2.mod	tc1_cc.oc tc3_cc.oc tc200_cc.oc	**hh2**	Fast Na+ and K+ currents responsible for action potentials
ITGHK.mod	tc1_cc.oc tc3_cc.oc tc200_cc.oc tc200_vc.oc tcD_vc.oc	**itGHK**	Ca++ current responsible for low threshold spikes (LTS)
VClamp.mod	el.oc	**SEVClamp**	Single electrode voltage clamp with three levels

- **nrnivmodl**:
  - Generates a folder called x86_64 that contains all the mod files (.mod), the C files (.c) and linker files (.lo) for all user-defined mechanisms
  - Also generates the files **libnrnmech.la**, **special**, **mod_func.c**, **mod_func.lo**

4/4/2016~4/14/2016

Notes from the NEURON Book Ch 3

- The derivation of the cable equation (p. 54) yields

  Time constant τ_m = R_m*C_m

  Space constant λ = (½) sqrt(d*R_m/R_a)

  where **R_m** = membrane resistance

  C_m = specific membrane capacitance

  R_a = cytoplasmic resistivity\

Notes from the NEURON Book Ch 4

- Spatial discretization (p. 59):
  - For a cable of length *L*, there are *m* intervals of

    length Δx = ***L/m*** with center at *x_i* = **(*i* + 0.5)*L/m***
  - Highest spatial frequency that could be represented: **(*m*-1)/2L**
  - This produces less error than the ordinary method (*m* points total with *m-1* intervals) at higher frequencies.
  - To represent the system better at high frequencies (high *n*), we need:

    Δx << L/n
- Temporal discretization (p. 62):
  - For the forward Euler method (sample @ T(t)), to avoid numerical instability, we need

    Δt/Δx² < R_a*c/a

    where *c* = specific membrane capacitance, *a* = radius
  - For both the backward Euler method (sample @ T(t + Δt), NEURON's default) & Crank-Nicholson method (sample @ T(t + (1/2)Δt), set global parameter "**secondorder** = 2"), numerical stability is achieved for all Δt.
  - Oscillations can be minimized by using a small Δt while the solution contains a large amplitude component that is changing rapidly and increasing Δt after the slower components dominate. Or it could be prevented by satisfying

    (*Δ*t/*τ*_m) / (*Δ*x/*λ*) ≤ ½
- Handling of nonlinearity (p.70):
  - Nonlinear equations such as Hodgkin-Huxley can be solved with a **staggered method** (updating gating variables at a separate time step than the voltage, offset by **0.5*Δ*t**), which "turns a system of differential equations with nonlinear coupling into a linear system of decoupled equations."
  - Unstaggered - first order accuracy
  - Staggered - second order accuracy
- Adaptive integration (p.72):
  - CVODE employs **Backward Differentiation Formula (BDF)** methods for stiff problems.
  - CVODE employs the **iterative Krylov method** to approximate the Jacobian.

    - CVODE was implemented using **encapsulated data structures** and placed in an **object-oriented class wrapper**. It can “efficiently retreat to any time within the previous integration interval.”
    - However, models that contain **linear circuits** and **extracellular fields** are not easily expressed in the ODE form that can be solved by CVODE. In these situations, **DASPK** is used.
    - Default error setting for CVODE (users can set specific error criteria for individual states):
      Membrane potential: **10 μV**
      Internal free calcium concentration: **0.1 nM**
- Local variable time step method for networks (**NetCon**, p. 80):
    - Cells are driven by **discrete input events**.
    - NEURON uses a **separate CVODE solver instance** for each cell
    - At each iteration, the **least time cell or event** is found. A cell is integrated and moved to a location on the cell list appropriate to its new time; an event is delivered to the proper cell, which becomes the least time cell.
    - This method works well for **sparse activity**. In periods of synchronous activity (e.g., if there are gap junctions), fixed time step integration is more suitable.
- Errors (p.83):
    - Only worry about **simulation errors** (from discretization) if they lead to trajectories significantly different from those defined by **parameter errors**

<u>Notes from the NEURON Book Ch 5</u>

- Properties that apply to sections as a whole (**section variables**, p. 94):

			Default value
L	section length	μM	
Ra	cytoplasmic resistivity	Ω·cm	35.4
cm	specific membrane capacitance	μF/cm²	1
nseg	discretization parameter (# of internal nodes), preferably an *odd number* so that there is an internal node at midpoint	1	

- Continuous functions of position within a section (**range variables**, p. 94):

diam	diameter	μm
area	surface area	μm²
ri	segment axial resistance	megΩ
v	membrane potential	mV

ina	sodium current	mA/cm²
nai	internal sodium concentration	mM
n_hh	HH potassium conductance gating variable	1

 - ***sectionname.rangevar*(x)** returns the value at the center of the segment that contains x, *not* the linear interpolation of adjacent segments.
 - x is a normalized distance between 0 and 1 (**range** or **arc length**).
   Default: x = **0.5**
 - **forall** or **forsec** runs through all sections.
- Test for **spatial accuracy**:
  (1) Run a simulation
  (2) **forall nseg*=3** (preserves previous nodes & reduces spatial error by 9)
  (3) Run simulation again, see if a significant qualitative or quantitative change has occurred
- Three ways to access a section, by order of precedence:
  - ***sectionname.variablename***
  - ***sectionname* { *stmt* }**
  - **access *sectionname*** (defines a default section)
- The topology of a model cell must be a **tree** (any two points are connected by a unique path).
  - **create *sectionname***
  - **connect *child*(0 or 1), *parent* (x)**
  - ***parent* connect *child*(0 or 1), x**
  - **disconnect()**
    Loops can exist if at least one element is a membrane mechanism (e.g. gap junction). However, it's better to implement with the **LinearMechanism** class.
  - The **root section** does not have to be the same as the default section.
- Useful functions that can have no arguments:

topology()	prints tree structure (p. 102)	
Shape()	creates Shape plot (p. 103)	
psection()	prints section properties (p. 103)	
n3d()	number of (x, y, z, *diam*) points used to specify the geometry of a section	

- Geometry specification (p. 103) -- two methods:
  - **Stylized specification**:
    axon { L=1000 diam=1 }
  - **3-D specification**:
    dend {

```
pt3dadd(10,0,0,5) // x, y, z, diam
pt3dadd(16,10,0,3)
pt3dadd(25,14,-3,2)
}
```

        - Here, a section is treated as a sequence of **frusta** (truncated cones)
        - **arc3d(i)** is the anatomical distance of the ith 3-D point from the 0 end of the secion
        - **area()** and **ri()** are computed by **trapezoidal integration** along the **centroid**
        - diam(x) as x ∈ [0,1] has same sense as **diam3d(i)** as i ∈ [0,n3d()-1] only if the *0 end of child is attached to the parent*
        - **diam(x)** is the diameter of a right cylinder that would have the same length and area as the segment that contains x
        - To avoid artifacts, diameters of adjacent segments at connecting points should match
    - If **define_shape()** is called (if a Shape object is created or if any GUI tool that show the shape of the model such as a Shape plot or the PointProcessManager is used), a stylized specification is automatically reinterpreted as a 3-D specification. As a consequence, **diam(x), area (x), ri(x)** might be altered (see p. 108-110)
- Biophysical properties (p. 111):
    - **Distributed mechanisms** are usually specified with **density units**, and are assigned by:

        soma **insert** hh
        dend **insert** pas

			Default
gnabar_hh			0.12
gkbar_hh			0.036
gl_hh			0.0003
el_hh			-54.3
g_pas			
e_pas			
ena			50
ek			-77
gna_hh	conductance density of HH Na channels	S/cm²	
ina	net Na current density	mA/cm²	

- Point processes are usually specified with **absolute units**, and are assigned by:

  objref stim
  soma stim = **new** IClamp(0.5)
  stim.amp = 0.1 // amplitude 0.1 nA
  stim.del = 1 // delay 1 ms
  stim.dur = 0.1 // duration 0.1 ms

amp	amplitude	
del	delay	ms
dur	duration	ms
rs	series resistance	10^6 Ω
gmax	peak conductance	µS
i	total current delivered	nA
loc(x)	moves location of point process to x of current section	

  - Range variables (p.114):
    - Iterate over nodes

      for (*var*) *stmt*
    - Linear over an interval

      rangevar*(*xmin*:*xmax*) = *e1*:*e2
    - If you change **nseg** and range variables are not constant, the hoc expressions used to set the range variables need to be **re-executed**
- Use the **d_lambda rule** to choosing a spatial grid (p. 122):
  - Most cells of interest has τ_m ≥ **8 ms**
  - Ionic and capacitive transmembrane currents are equal at the frequency **f_m = 1/(2$\pi\tau$_m)** (~**20 Hz**)
  - Specific membrane resistance R_m "has little effect on the propagation of signals ≥ **5*f_m**", therefore, **λ_100** should be "high enough for signal propagation to be insensitive to shunting by ionic conductances"
  - **λ_f = sqrt(diam/(4*PI*f_m*Ra*cm))**, implemented in stdlib.hoc
  - In GUI, can set **d_lambda** (maximum allowable distance between adjacent nodes) in Cellbuilder (default is **0.1**)
  - Alternatively, you can specify **nseg** (the actual # of grid points) or **d_X** (maximum anatomical distance between grid points in µm)
  - In the case of d_lambda & d_X, nseg is automatically set to be an odd number

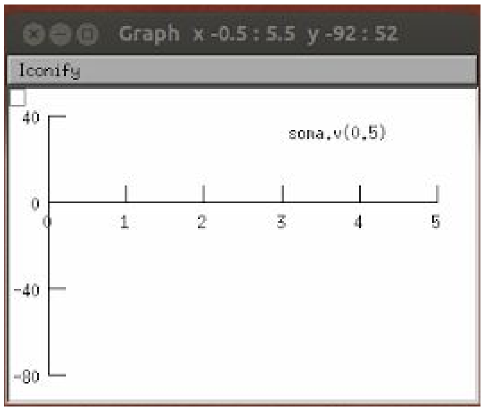
<u>Notes from the NEURON Book Ch 6</u>

- GUI vs hoc (p.129):

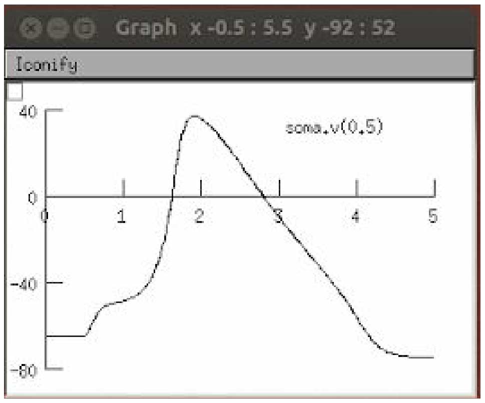
- **GUI** has useful **optimization** and **electrotonic analysis** tools difficult to implement in hoc
- **hoc** is more appropriate for noninteractive simulations (those that generate large amounts of data)

- GUI codes (p.129):
    - All except **Print & File Window Manager** (written in **C**) are written in **hoc**
- Running a hoc file (p. 133):
    - Use **nrngui** example.hoc if GUI toolbar wanted
    - Use **nrniv** example.hoc if GUI toolbar not wanted
- Implementing a model with a hoc file
    - See “Ch6.hoc” (Same specifications as Chapter 1)
    - Test with the following:

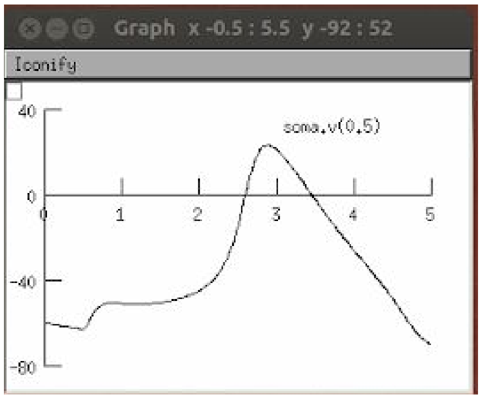
```
nrniv
load_file(“Ch6.hoc”)
```



```
go()
```



```
v_init = -60
go()
```



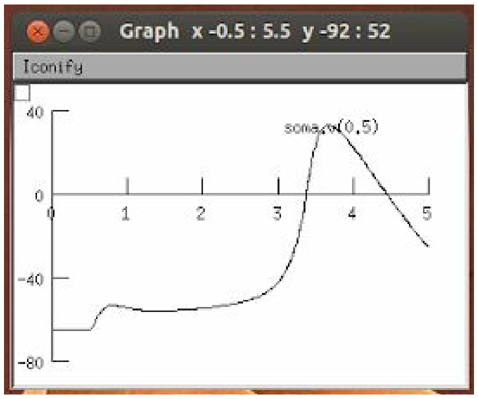
quit()

- Combining hoc and the GUI (p. 141):
    - NEURON Main Menu toolbar:

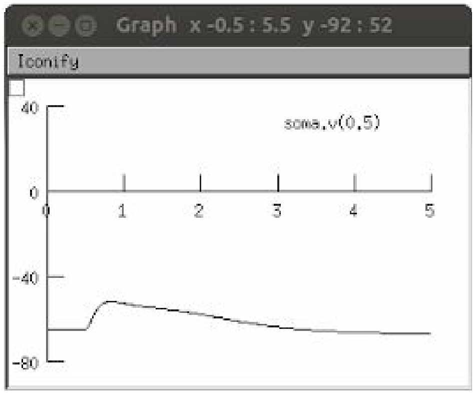
      load_file("nrngui.hoc")
    - Default section:

      access soma
    - Add inhibitory synapse (p. 144):

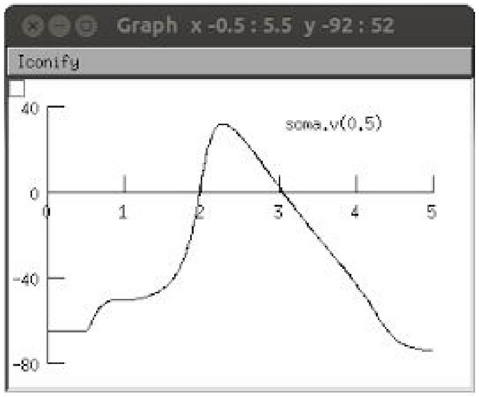
```
objref isyn
soma isyn = new AlphaSynapse(0.5)
isyn.onset = 0.5
isyn.tau = 0.3
isyn.gmax = 0.04
isyn.e = -70
```



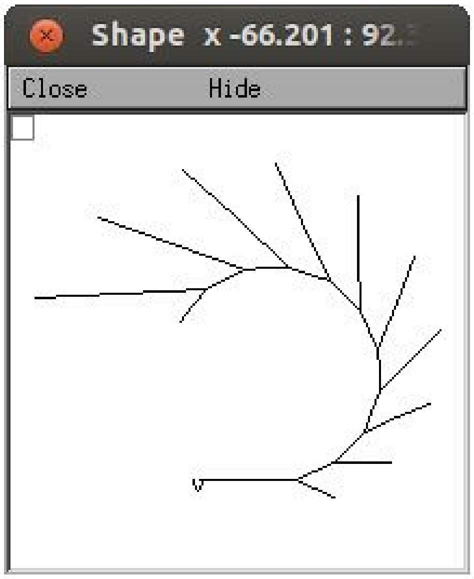
isyn.tau = 1

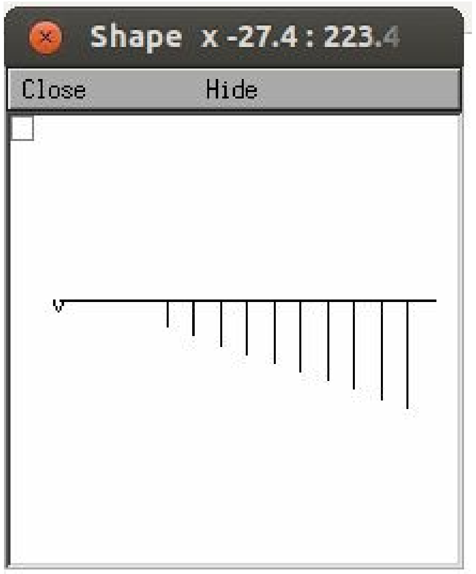


isyn.tau = 3



- Geometry specification (p. 145), using L, diam (left fig) or **pt3dadd()** (right fig)





Using absolute coordinates

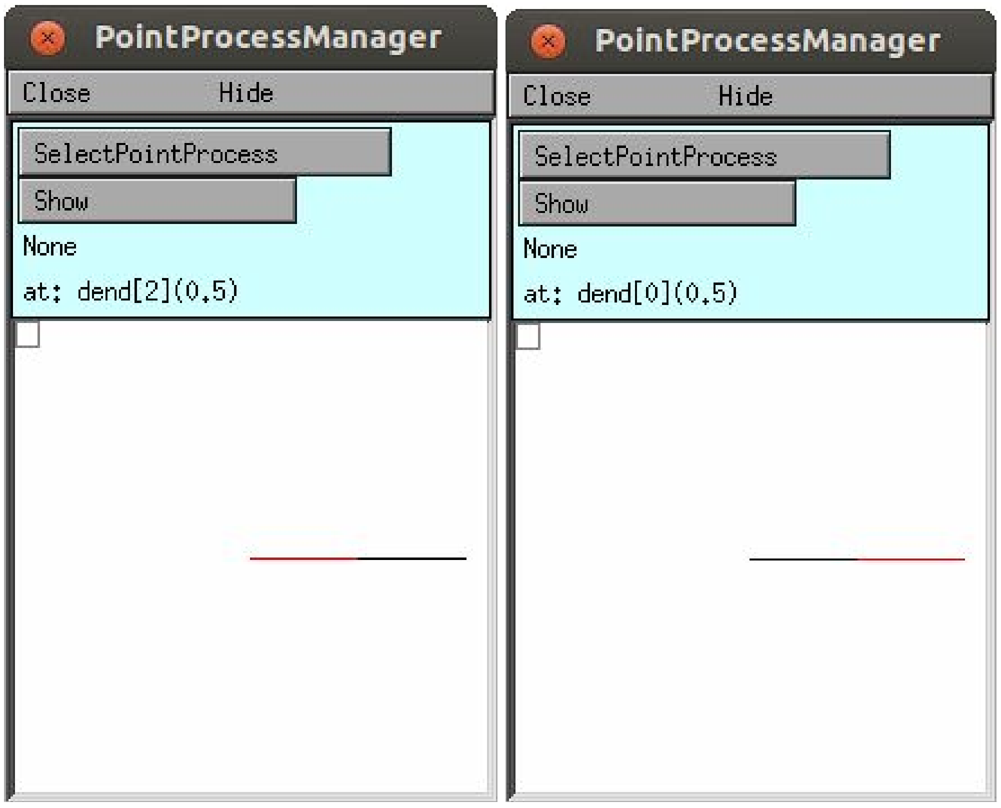
```
pt3dadd(30, 0, 0, 5)
pt3dadd(60, 0, 0, 5)
```

Or relative coordinates

```
pt3dadd(0, 0, 0, 5)
pt3dadd(30, 0, 0, 5)
```

gives the same result

- Disappearing sections (p.148):
  - If the 1 end of a child is attached to a parent, confusion may occur:



Here dend[1] overlaps with dend[2] so it's impossible to select

- Graph from hoc code doesn't update under GUI:

  addplot(g, 0)

  (adds g to a list of graphs that the **standard run system** automatically updates during the course of the simulation)
- Session files only contain the state of the GUI tools; any updates from the hoc code would not be saved
- Changes made at the hoc level are not propagated to the GUI tools
- Variables cannot be declared with a new **type** during the same session
- Changes to a **template** (for a class) require exiting NEURON and restarting

<u>**Notes from the NEURON Book Ch 7**</u>

- RunControl panel:
  - A **single step** is 1/(Points plotted/ms) ms and consists of 1/(dt*Points plotted/ms) calls to **fadvance()**
  - **Quiet**: when checked, turns off graph updates during a simulation
- Standard Run System:
  - nrn/share/lib/hoc/**stdrun.hoc**
  - **fadvance()**

        - Integrates all equations of the system from t to t+dt
    - **advance()**
        - With the (default) **fixed step method**, states & parameters can be changed
        - With the **variable step methods**, states & parameters cannot be changed unless **cvode.re_init()** is executed after the change
        - "The only way that time-varying parameters may be simulated with variable step methods is in the context of a **model description** or by using the interpolated form of **Vector.play()**"
    - **step()**
        - local i

          ```
          if (using_cvode_) {
                  advance()
          } else for i=1,nstep_steprun {
                  advance()
          }
          Plot()
          ```
        - **set_dt()** reduces **dt** if necessary to ensure that Dt steps (interval between plots) lie on a dt boundary
    - **steprun()**

        -
          ```
          proc steprun() {
                  step()
                  flushPlot()     // ensures that any deferred graph updates
                                  // are performed
          }
          ```
        - Same as **Single Step** button on RunControl
    - **continuerun()**
        - **Continue til** button on RunControl: **continuerun(runStopAt)**
        - **Continue for** button on RunControl: **continuerun(t+runStopIn)**
        - **Stop** button on RunControl: **stoprun** becomes nonzero
          stoprun is a **global variable** in C
        - Uses a stopwatch to count the seconds in a variable called **realtime** (displayed in **Real-Time** on RunControl)
        - **doEvents()** is called at every step for the first two seconds and less often after that
        - Plots are flushed at intermediate times only if **stdrun_quiet**==0 (toggled by **Quiet** on RunControl)
    - **run()**

        -
          ```
          proc run() {
                  stdinit()
                  continuerun(tstop)
          }
          ```
        - Implements the run part of **Init & Run** button on RunControl

        - **tstop** is shown in **Tstop** on RunControl
    - **finitialize()**
        - See Chapter 8
    - **init()**
        - proc init() {

```
    finitialize(v_init)
    // User-specified customizations go here.
    // If this invalidates the initialization of variable time step
    // integration and vector recording, uncomment the following code:
    /*
    If (cvode.active()) {
        cvode.re_init()
    } else {
        fcurrent()
    }
    frecord_init()
    */
    fcurrent()
}
```

    - **stdinit()**
        - proc stdinit() {

```
    realtime=0    // "run time" in seconds
    startsw()     // initialize run time stopwatch
    setdt()       // ensures that the points plotted fall on time step
                  // boundaries (1/(steps_per_ms*dt) is an integer))
    init()
    initPlot()    // begins each plotted line at t = 0 with the proper y
                  // value
}
```

        - **v_init** is set in **Init** on RunControl, which calls **stdinit()**
- fadvance()
    - Implemented in nrn.../src/nrnoc/fadvance.c
    - Order of additions:
      (1) **CVODE** (variable order, variable time step integrator)
      (2) **NetCon** (event delivery system)
      (3) **LinearMechanism** (overlay of algebraic equations onto the Jacobian)
      (4) **DASPK** (differential algebraic solver)
    - Turning variable time step integration on or off: **VariableStepControl** panel
    - [A lot of stuff I can't understand]
- CVODE
    - Initialize
    - Advance
    - Interpolate

    - [A lot of stuff I can't understand]

<u>**Notes from the NEURON Book Ch 8 (last modified 05/05/2016)**</u>

- Variables in NMODL:
    - **PARAMETERS** - can be set as constants throughout the simulation
    - **STATE** - any variable that is an unknown quantity in a set of equations
        - The number of STATEs in a model description is equal to the number of equations
        - E.g., nodes in a resistive network (although this technically doesn't require initialization)
    - **ASSIGNED** - any variable that is not a PARAMETER or a STATE
        - E.g., the membrane potential **v**
- STATE variables can be initialized to:
    - An unchanging **steady state**
    - Parameters that **meet certain conditions**
    - Random values that need to be **saved** in order for the simulation to be reproduced
- finitialize()
    - Implemented in nrn.../src/nrnoc/**fadvance.c**
    - [A lot of stuff I can't understand]
- Custom initialization:
    - Default syntax:

```
proc init() {
        finitialize(v_init)
        // User-specified customizations go here.
        if (cvode.active()) {
                cvode.re_init()
        } else {
                fcurrent()
        }
        frecord_init()
}
```

    - Load user-defined version *after* loading stdrun.hoc
- INITIAL blocks
    - **SOLVE** scheme **STEADYSTATE** sparse
    - [A lot of stuff I can't understand]
- NEURON blocks
    - **GLOBAL** m0 specifies that every m will be set to the single global m0 value
    - **RANGE** h0 specifies that h will be set to the possibly spatially varying h0 values
- Ion mechanisms
    - **USEION** ca READ ica WRITE cai, cao
    - [A lot of stuff I can't understand]
- Initializing concentrations in hoc

  - The default concentrations for ion names created by the user are **1 mM**
  - If one or more sections of the model are supposed to have different initial concentrations, use the **ion_style()** function (see NEURON documentation)
- Initializing to a particular resting potential (p. 195)
  - Adjust the leak current so that the total membrane current at steady state is 0:

    ```
    0 = ina + ik + gl_hh*(v - el_hh)
    => el_hh = (ina + ik + gl_hh*v)/gl_hh
    ```
  - Adjust a constant current such that

    ```
    0 = ina + ik + i_hh + i_constant
    => ic_constant = - (ina + ik + il_hh)
    ```
- Initializing to steady state (p. 197)
  - Use a fixed, large time step to reach steady state:

    ```
    t = -1e10
    dtsav = dt
    dt = 1e9
    // if cvode is on, turn it off to do large fixed step
    temp = cvode.active()
    if (temp != 0) { cvode.active(0) }
    while (t<-1e9) {
        fadvance()
    }
    // restore cvode if necessary
    if (temp != 0) { cvode.active(1) }
    dt = dtsav
    t = 0
    ```
- Initializing to a desired state (p. 198)
  - Use the class **SaveState()**:

    ```
    objref svstate, f
    svstate = new SaveState()
    svstate.save()
    f = new File("states.dat")
    svstate.fwrite(f)
    ```
  - Future sessions can read the file into the SaveState object by

    ```
    objref svstate, f
    svstate = new SaveState()
    f = new File("states.dat")
    svstate.fread(f)
    ```
  - Now initiate by:

    ```
    svstate.restore()
    t = 0 // t is one of the "states"
    ```
- Initializing by changing model parameters
  - [A lot of stuff I can't understand]

4/15/2016~4/17/2016

Details of Destexhe et al 1998a model

- **Geometry**
  - Detailed cell model (**tc200.geo**):
    - soma: diameter = 21.7456 µm length = 42.6901 µm area = 2916.41 µm²
      - 13 3-D points; 14 outline points numbered 347-360
      - Outline diameter = 24.4073 µm
    - 11 primary neurites (**primary_branches_cell**)
    - 108 branches (**branches_cell**) totaling 7094.47 µm in length (**max_dx_cell**), 33581 µm² in area
    - 1224 tree points (**points_cell**) translated to 206 segments (**nseg_cell**, 1 requested)
    - Neurites divided into segments of equal distance between adjacent digitized branch points.
    - Segment length constrained to be < 7094.47 µm.
    - No. points   1238
    - No. trees   12

Section	# of branches	Diameter (µm)	Surface area (µm²)
soma	1		
dend1	1		
dend2	3		
dend3	13		
dend4	21		
dend5	1		
dend6	31		
dend7	9		
dend8	33		
dend9	47		
dend10	27		
dend11	19		

    -

  - Dissociated cell model (**tcD.geo**):

9 compartments, 2 dendrites:
(dend2[4]) 1--0
(dend2[3],[2]) 1--0 1--0
(dend2[1],[0]) 1--0 1--0 (connected to soma @ 0.5)
(dend1[2]) 1--0
(dend1[1],[0]) 1--0 1--0 (connected to soma @ 0.5)
(soma) 1--0

	3-D points (x, y, z, diam)	Branch of
soma	-23.25 -7.35 -34.2 **0** -21.9 -6.18 -31.05 14.959 -21 -5.8875 -30.4875 21.439 -20.1 -5.03862 -28.6974 24.974 -16.95 -4.48638 -27.5526 25.363 -16.05 -4.26426 -26.5824 26.719 **-13.8 -3.05889 -24.0111 28.865** -9.75 -1.51125 -21.6675 28.311 -4.8 1.0875 -17.5662 25.297 -3.45 0.795 -15.435 23.776 -0.75 1.875 -11.4192 15.383 0.15 1.56 -10.53 10.826 1.5 3 -9 **0**	
dend1[0]	7 -3 -5 4.8 7.5 -4 2.5 3.9 9 -6 7.5 2.5 11 -8 10.5 **2.5**	soma(0.5)
dend1[1]	11 -8 10.5 **2.5** 14 -8 11 2 17 -7.5 12.5 2 20.5 -6.5 14 2 23.5 -4.5 13.5 2 27 -2.5 13 3	dend1[0]
dend1[2]	11 -8 10.5 **2.5** 12 -10 10.5 2 15 -11 14 2 17.5 -12 19.5 2 18.5 -14 23.5 2 18 -16.5 26.5 2 17 -19 30.5 2	dend1[0]
dend2[0]	-13.5 0 -38.5 2.2 -22.5 0.5 -33.5 **2.2**	soma(0.5)
dend2[1]	-22.5 0.5 -33.5 **2.2**	dend2[0]

	-30	-1.5	-33.5	1.2	
	-36.5	-2	-33.5	1.2	
	-41	-2	-33.5	1.2	
	-47	-2	-28	1.2	
	-53	-1	-21	1.2	
	-60.5	1.5	-14.5	1.5	
	-66.5	2.5	-13	1.5	
	-72.5	1	-13.5	1.2	
dend2[2]	-22.5	0.5	-33.5	**2.2**	dend2[0]
	-26	2.5	-36	**1.6**	
dend2[3]	-26	2.5	-36	**1.6**	dend2[2]
	-31.5	2	-35.5	1.2	
	-38.5	2.5	-34.5	1.2	
	-45.5	4	-29	1.2	
	-53.5	5	-24.5	1.2	
	-61.5	5.5	-20	1.2	
dend2[4]	-26	2.5	-36	**1.6**	dend2[2]
	-32	8.5	-29	1.2	
	-38	15	-24.5	1.2	
	-45.5	22	-22.5	1.2	
	-52	27.5	-20	1.2	
	-58.5	33.5	-17	1.2	
	-63	40	-15	1.2	
	-68.5	47.5	-13.5	1.2	

	Length (μm)	Diameter (μm)	Surface area (μm²)
Soma			
dend1[0]			
dend1[1]			
dend1[2]			
dend2[0]			
dend2[1]			
dend2[2]			
dend2[3]			
dend2[4]			

- How it was obtained:
  - The full cell has an input capacitance of 113 pF, whereas dissociated cells have 16.7 pF on average.
  - If one assumes that the **input capacitance is proportional to the area of the cell**, then a typical dissociated cell must have an area of: 23980.547 µm² * 16.7 / 113 = 3544 µm²
  - This area was obtained from the full geometry by keeping the **thickest and most proximal dendrites** (i.e. dend3[0,1,8] and dend7[0,1,4,5,8]) in order to match this area. The aspect of the dissociated cell was consistent with the shape of dissociated TC cells
- **rescale_diameters()**:
  - Uses a sigmoid transformation function:
    newdiam = oldiam * 1/(1 + exp(-(diam-diam_hlf)/diam_stp))
    diam_min = **0.8** // minimal diameter
    diam_hlf = **1.5** // half-value of the sigmoid
    diam_stp = **1.5** // steepness of sigmoid
  - The total membrane surface area is of 3427.67 µm² (PI*diam*L) and 3974.32 µm² (using area function)

- 3-compartment model (**tc3.geo**):

```
1--0 1--0     1--0
 (dend1)   (soma)
```

	Length (µm)	Diameter (µm)	Surface area (µm²)
Soma	38.42	26	2624.6
dend1[0]	12.49	10.28	403.37
dend1[1]	84.67	8.5	2260.99

  - Total surface area = 5288.96 µm² (2664.36 for dendrites)
  - (Reconstructed cell was 23980.547 µm² (21355.8 for dendrites))
  - **Average reduction factor** for dendrites is CorrD = 8.02
  - SimFit of experimental voltage-clamp trace gives CorrD = 7.954

- Single compartment model (**tc1.geo**):

	Length (µm)	Diameter (µm)
Soma	100	76.58

  - Total surface area = 24058 µm²
  - (Reconstructed cell was 23980.547 µm²)

3/28/2016

Implemented NEURON Book Ch 1 simple model

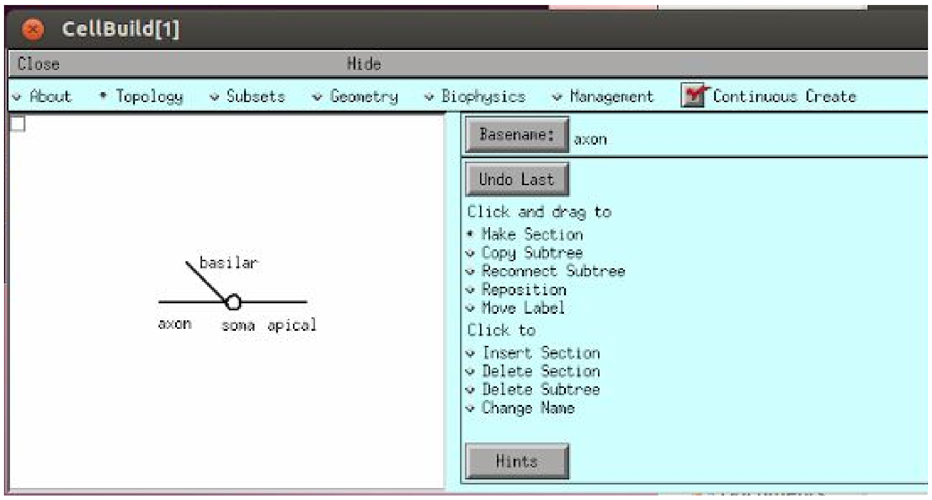
Session file:
adamX/NEURON_sessions/NEURON_Book_Ch1.ses

HH = Hodgkin-Huxley equations for action potentials, for equations see:
https://en.wikipedia.org/wiki/Hodgkin%E2%80%93Huxley_model

Neuron(s):
1 neuron with 4 compartments:

	Length (μm)	Diameter (μm)	Biophysics
Soma	30	30	HH with g_Na_max = 0.12 S/cm², g_K_max = 0.036 S/cm², g_leak = 0.003 S/cm², E_leak = -54.3 mV
Apical dendrite	600	1	Passive with R_m = 5000 Ω cm², E_pas = -65 mV
Basilar dendrite	200	2	Passive with R_m = 5000 Ω cm², E_pas = -65 mV
Axon	1000	1	HH with g_Na_max = 0.12 S/cm², g_K_max = 0.036 S/cm², g_leak = 0.003 S/cm², E_leak = -54.3 mV

C_m = 1 μF/cm²
cytoplasmic resistivity (R_a) = 100 Ω cm
Temperature = 6.3 °C


Basename: axon

Subsets:
has_HH = soma + axon
no_HH = apical + basilar

Discretization:
d_λ = 0.1 (10% of the AC length constant)

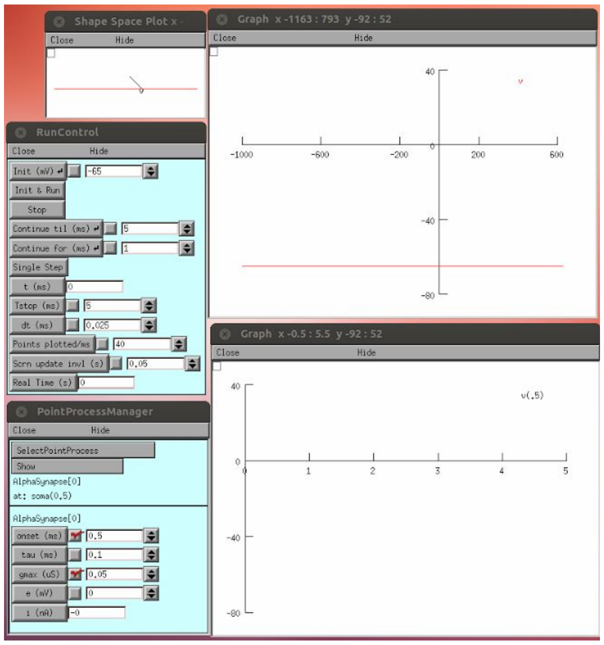
Execution:
Continuous Create

Synapse(s):
1 synapse with time course described by:
g_s(t) = 0 for t < t_act
g_s(t) = g_max * [(t-t_act)/tau_s] * e^[-(t-t_act)/tau_s] for t ≥ t_act

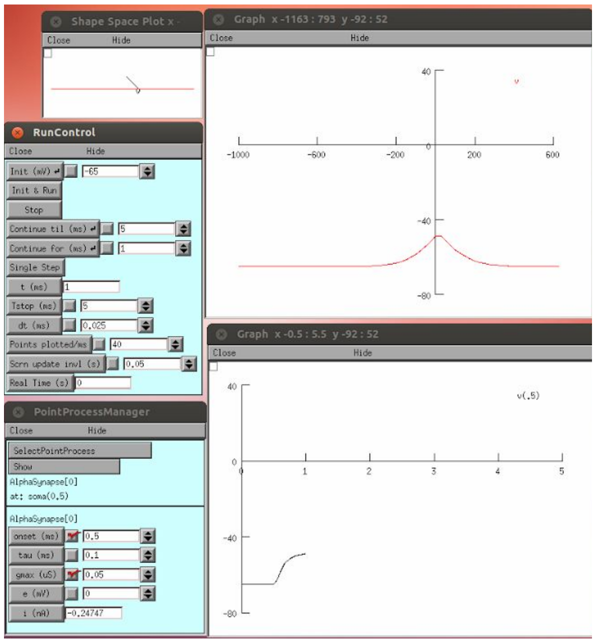
where

t_act = 0.5 ms
g_max = 0.05 μS
tau_s = 0.1 ms
E_s = 0 mV

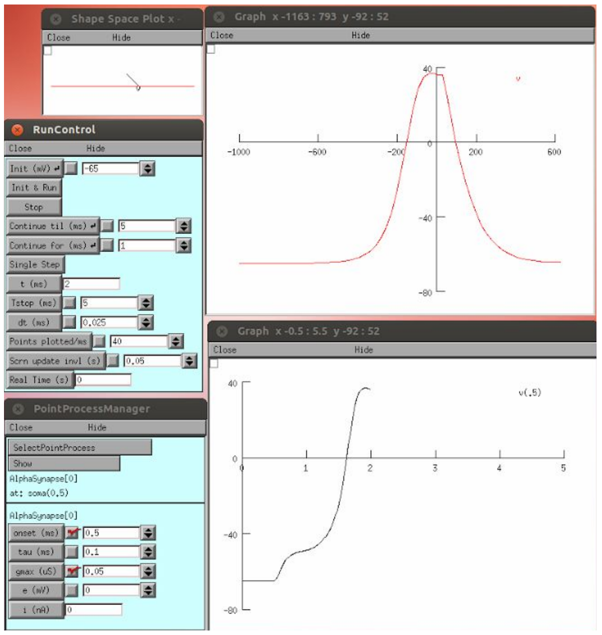
Synapse location @ soma(0.5):



```
RunControl
Init (mV)            -65
Init & Run
Stop
Continue til (ms)    5
Continue for (ms)    1
Single Step
t (ms)               0
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0

PointProcessManager
SelectPointProcess
Show
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        -0
```



```
RunControl
Init (mV)            -65
Init & Run
Stop
Continue til (ms)    5
Continue for (ms)    1
Single Step
t (ms)               1
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0

PointProcessManager
SelectPointProcess
Show
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        -0.24747
```


Shape Space Plot x

Graph x -1163 : 793 y -92 : 52

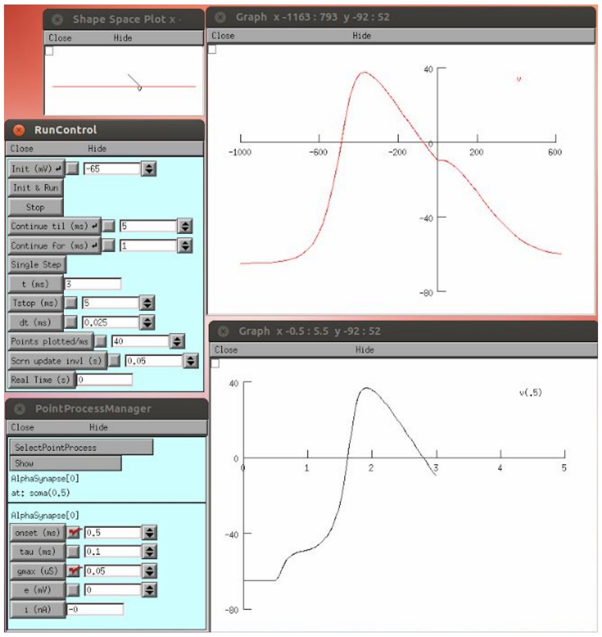
RunControl

```
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               2
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
```

Graph x -0.5 : 5.5 y -92 : 52

PointProcessManager

```
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        0
```


Shape Space Plot x

Graph x -1163 : 793 y -92 : 52

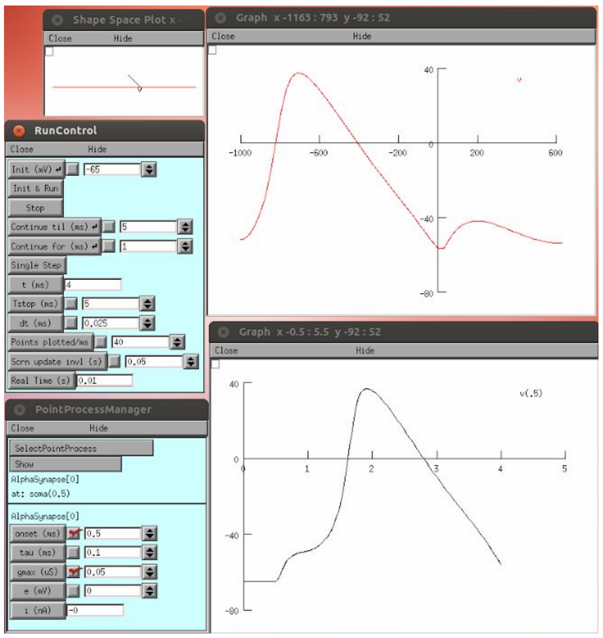
RunControl

```
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               3
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
```

Graph x -0.5 : 5.5 y -92 : 52

PointProcessManager

```
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        -0
```


Shape Space Plot x

Graph x -1163 : 793 y -92 : 52

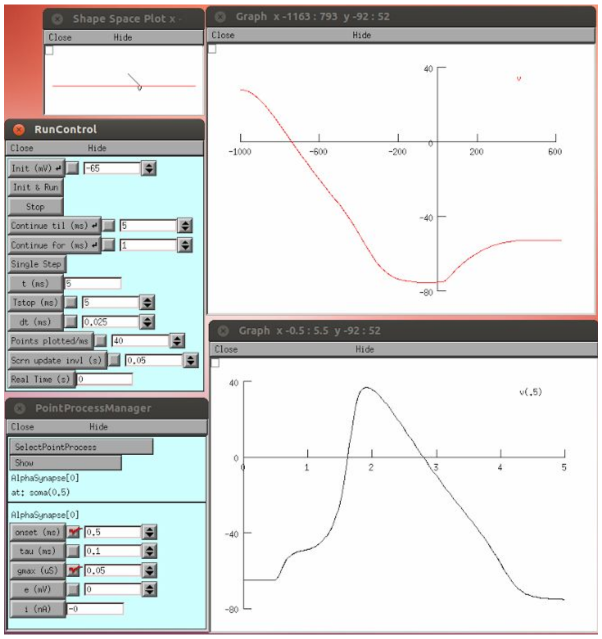
RunControl

```
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               4
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0.01
```

Graph x -0.5 : 5.5 y -92 : 52

PointProcessManager

```
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        -0
```


Shape Space Plot x

Graph x -1163 : 793 y -92 : 52

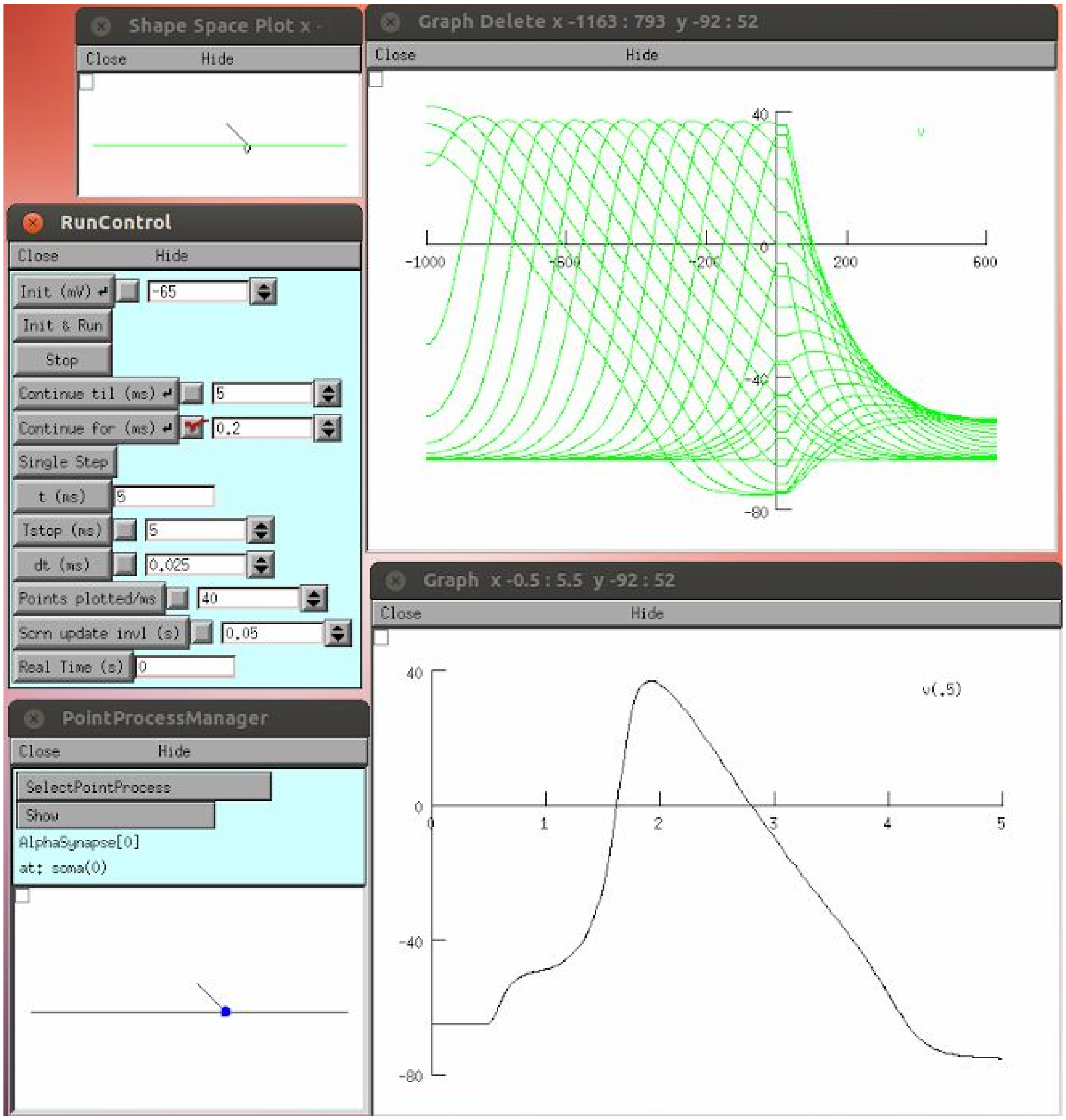
RunControl

```
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               5
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
```

Graph x -0.5 : 5.5 y -92 : 52

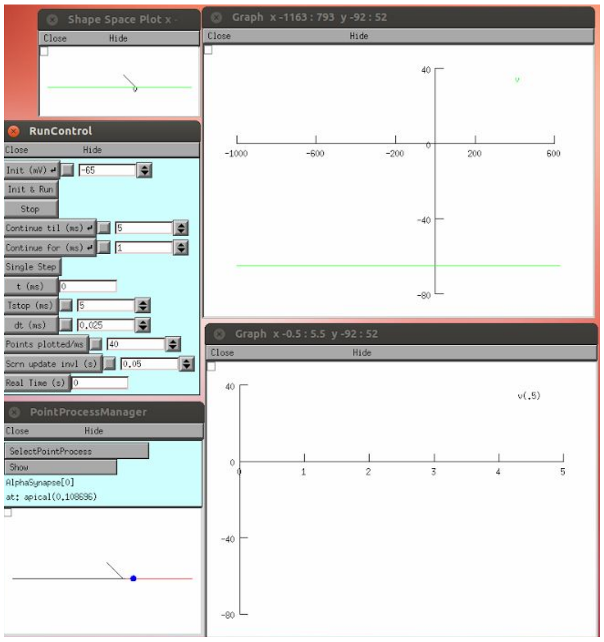
PointProcessManager

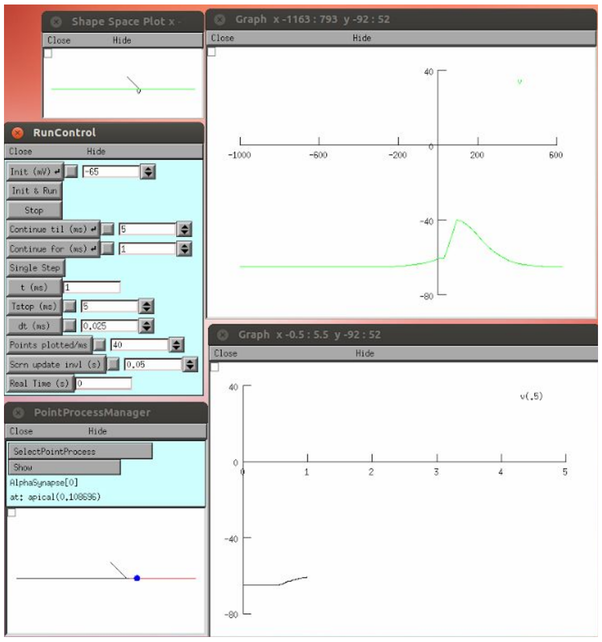
```
AlphaSynapse[0]
at: soma(0.5)

AlphaSynapse[0]
onset (ms)  ✓ 0.5
tau (ms)      0.1
gmax (uS)   ✓ 0.05
e (mV)        0
i (nA)        -0
```

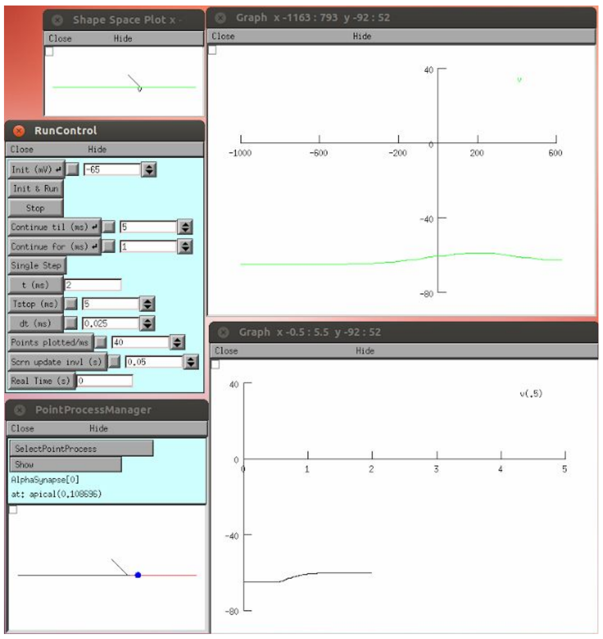


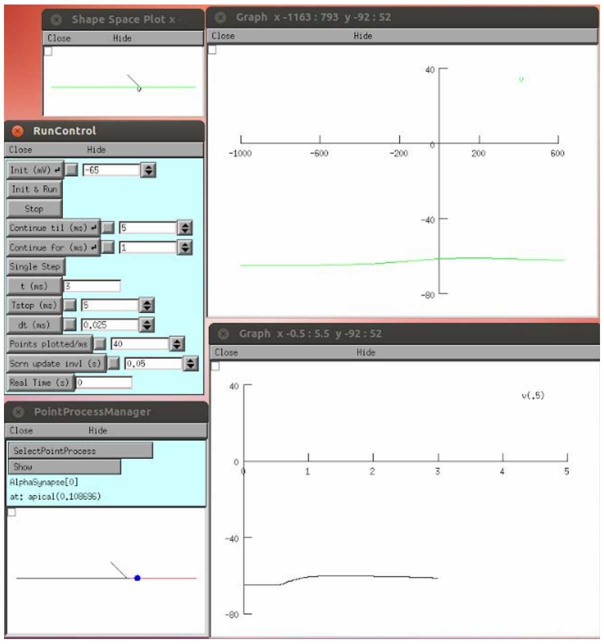
```
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    0.2
t (ms)               5
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0

AlphaSynapse[0]
at: soma(0)
```

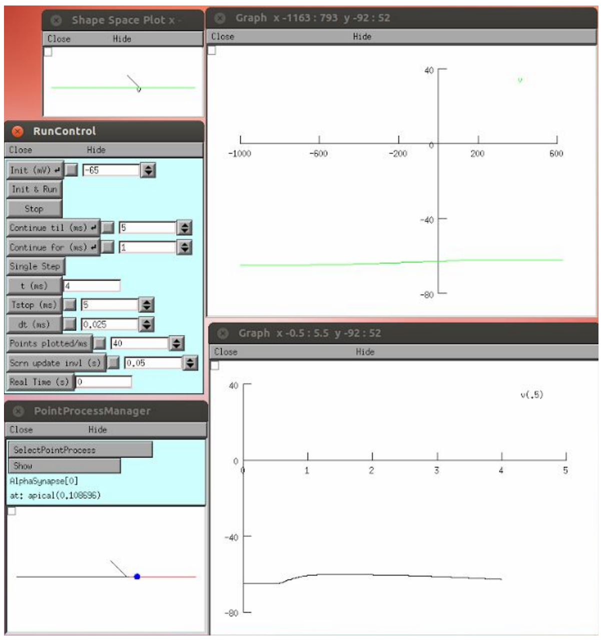
Synapse location @ apical(0.108696):


```
Shape Space Plot x
Graph x -1163 : 793 y -92 : 52
RunControl
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               0
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
Graph x -0.5 : 5.5 y -92 : 52
PointProcessManager
AlphaSynapse[0]
at: apical(0.108696)
```


```
Shape Space Plot x
Graph x -1163 : 793 y -92 : 52
RunControl
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               1
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
Graph x -0.5 : 5.5 y -92 : 52
PointProcessManager
AlphaSynapse[0]
at: apical(0.108696)
```

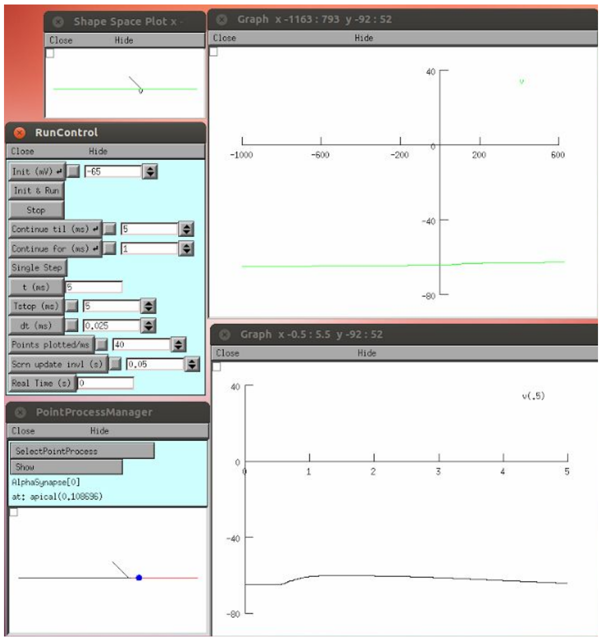

```
Shape Space Plot x
Graph x -1163 : 793 y -92 : 52
RunControl
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               2
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
Graph x -0.5 : 5.5 y -92 : 52
PointProcessManager
AlphaSynapse[0]
at: apical(0.108696)
```


```
Shape Space Plot x
Graph x -1163 : 793 y -92 : 52
RunControl
Init (mV)            -65
Continue til (ms)    5
Continue for (ms)    1
t (ms)               3
Tstop (ms)           5
dt (ms)              0.025
Points plotted/ms    40
Scrn update invl (s) 0.05
Real Time (s)        0
Graph x -0.5 : 5.5 y -92 : 52
PointProcessManager
AlphaSynapse[0]
at: apical(0.108696)
```



```
Init (mV)              -65
Init & Run
Stop
Continue til (ms)      5
Continue for (ms)      1
Single Step
t (ms)                 4
Tstop (ms)             5
dt (ms)                0.025
Points plotted/ms      40
Scrn update invl (s)   0.05
Real Time (s)          0
```

```
AlphaSynapse[0]
at: apical(0.108696)
```



```
Init (mV)              -65
Init & Run
Stop
Continue til (ms)      5
Continue for (ms)      1
Single Step
t (ms)                 5
Tstop (ms)             5
dt (ms)                0.025
Points plotted/ms      40
Scrn update invl (s)   0.05
Real Time (s)          0
```

```
AlphaSynapse[0]
at: apical(0.108696)
```

3/29/2016~3/31/2016

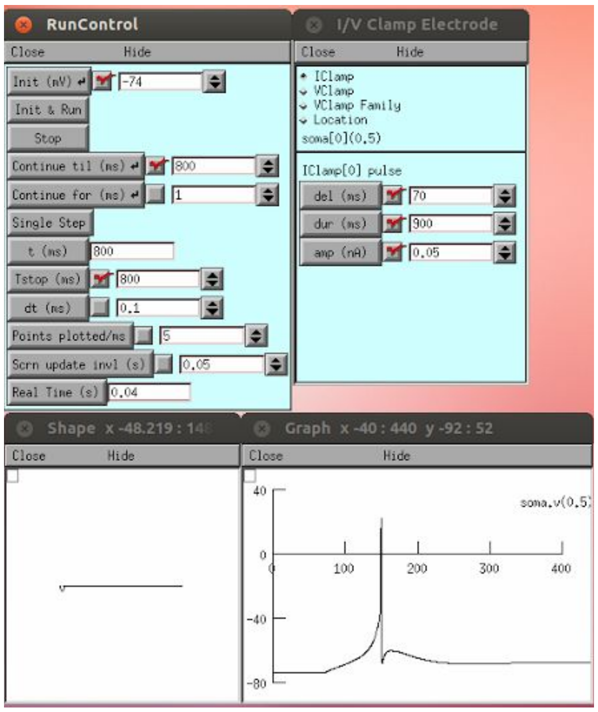
<u>Read Destexhe et al 1998 ("Dendritic low-threshold calcium currents in thalamic relay cells")</u>

- Single neuron models of **thalamocortical relay neurons** (TC cells) in the **ventrobasal nucleus** (VB) of rats
- 4 models: **Intact-cell**, **dissociated-cell** (these are reconstructed from a computerized tracing system), **3-compartment** & **single compartment**.
- Simplification: dendritic arbor reduced to 2 proximal dendrites, then together with soma reduced to single compartment, based on **conservation of axial resistance**:
  r = sqrt(Sum_i r_i^2)
  l = Sum_i l_i*r_i / Sum_i r_i
  Total area is not conserved, but a **dendritic correction factor** is introduced to preserve the **input resistance** and **time constant**.
- Upon passive fitting (injecting leak currents @ all compartments), the electrotonic length of the longest dendrite was determined to be **0.34 space constants**, showing that TC neurons are **relatively compact electrotonically** (contrary to reticular thalamic neurons).
- In both the intact-cell & 3-compartment models, T-current densities must be **increased** in the distal dendrites **many fold** to reproduce voltage clamp **I-V curves** & current clamp low-threshold spikes (**LTS**) from experiments.
- **Tail currents** showed that the intact-cell model had **poor voltage clamp control**, accounting for the shift in I-V curve
- With the total number of T-channels held constant, localizing T-channels in the dendrites **decreased the excitability** of the cell with respect to LTS generation
- Upon bombarding TC cells with mixed excitatory and inhibitory inputs (**dendritic shunt conductances**), there is a more graded bursting behavior in the LTS response curve. This curve is also shifted to the right (more current needed to elicit same LTS response) more efficiently when the T-channels are localized to the dendrites.

<u>Implemented Destexhe et al 1998 model</u>

Troubleshoot:
Files couldn't compile per instructions (using "nrnivmodl"), with error "cannot find -lncurses"
Turns out that our server is 64-bit but the original model was written in a 32-bit environment.
Fixed after installing a 32-bit package "lib32ncurses5-dev" (using "sudo apt-get install lib32ncurses5-dev")
Reference:
http://stackoverflow.com/questions/14416487/gcc-usr-bin-ld-error-cannot-find-lncurses

Burst behavior in single compartment model:



RunControl

Close Hide

Init (mV) -74

Init & Run

Stop

Continue til (ms) 800

Continue for (ms) 1

Single Step

t (ms) 800

Tstop (ms) 800

dt (ms) 0.1

Points plotted/ms 5

Scrn update invl (s) 0.05

Real Time (s) 0.04

I/V Clamp Electrode

Close Hide

- IClamp
- VClamp
- VClamp Family
- Location

soma[0](0.5)

IClamp[0] pulse

del (ms) 70

dur (ms) 900

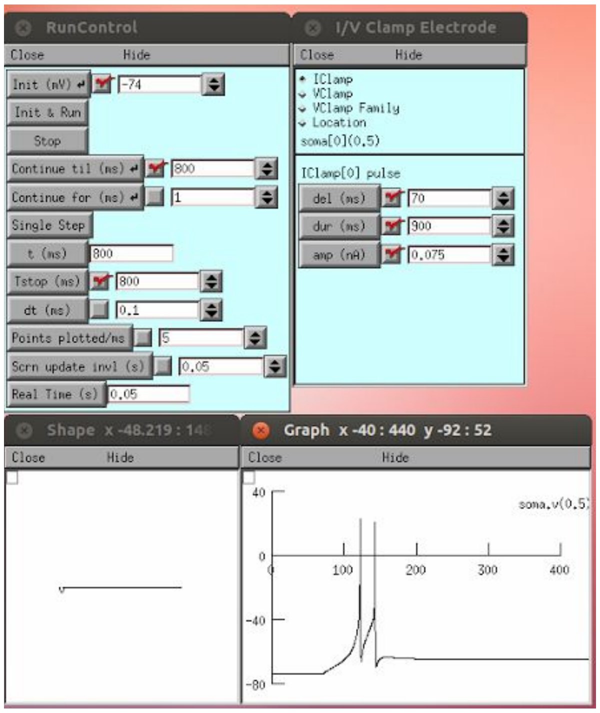
amp (nA) 0.05

Shape x -48.219 : 14

Close Hide

Graph x -40 : 440 y -92 : 52

Close Hide



RunControl

Close Hide

Init (mV) -74

Init & Run

Stop

Continue til (ms) 800

Continue for (ms) 1

Single Step

t (ms) 800

Tstop (ms) 800

dt (ms) 0.1

Points plotted/ms 5

Scrn update invl (s) 0.05

Real Time (s) 0.05

I/V Clamp Electrode

Close Hide

- IClamp
- VClamp
- VClamp Family
- Location

soma[0](0.5)

IClamp[0] pulse

del (ms) 70

dur (ms) 900

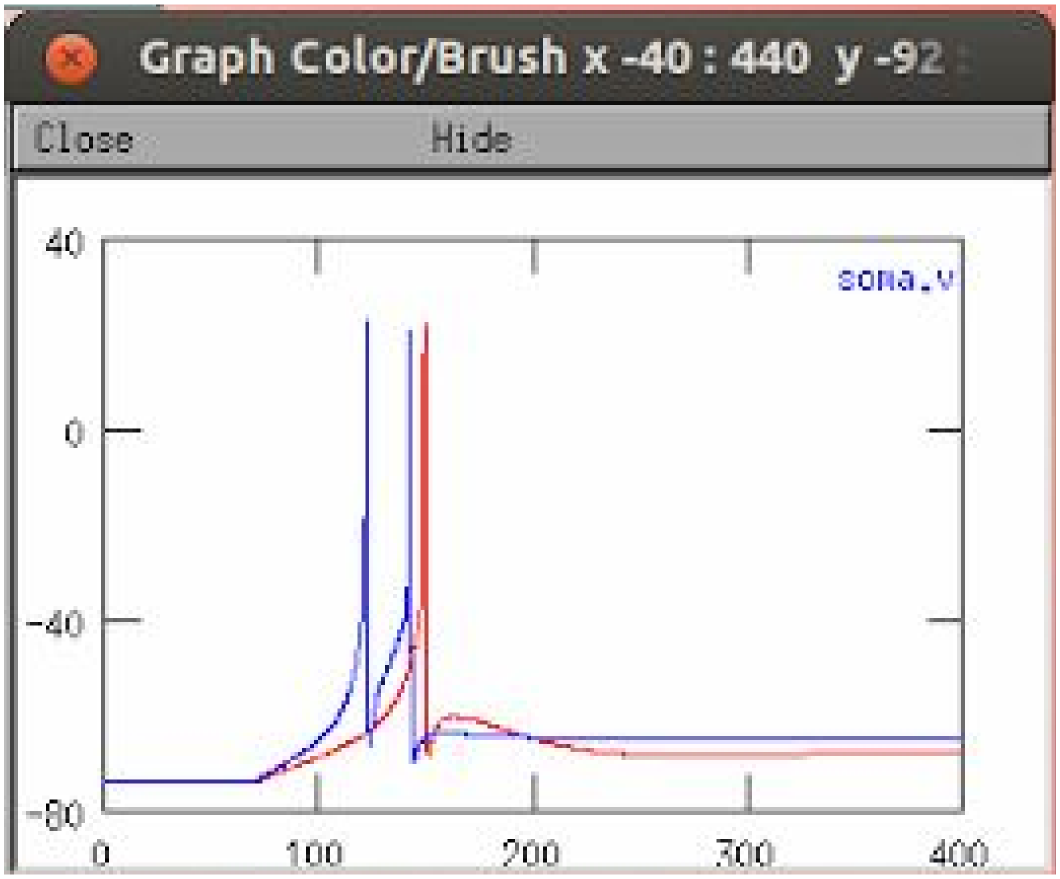
amp (nA) 0.075

Shape x -48.219 : 14

Close Hide

Graph x -40 : 440 y -92 : 52

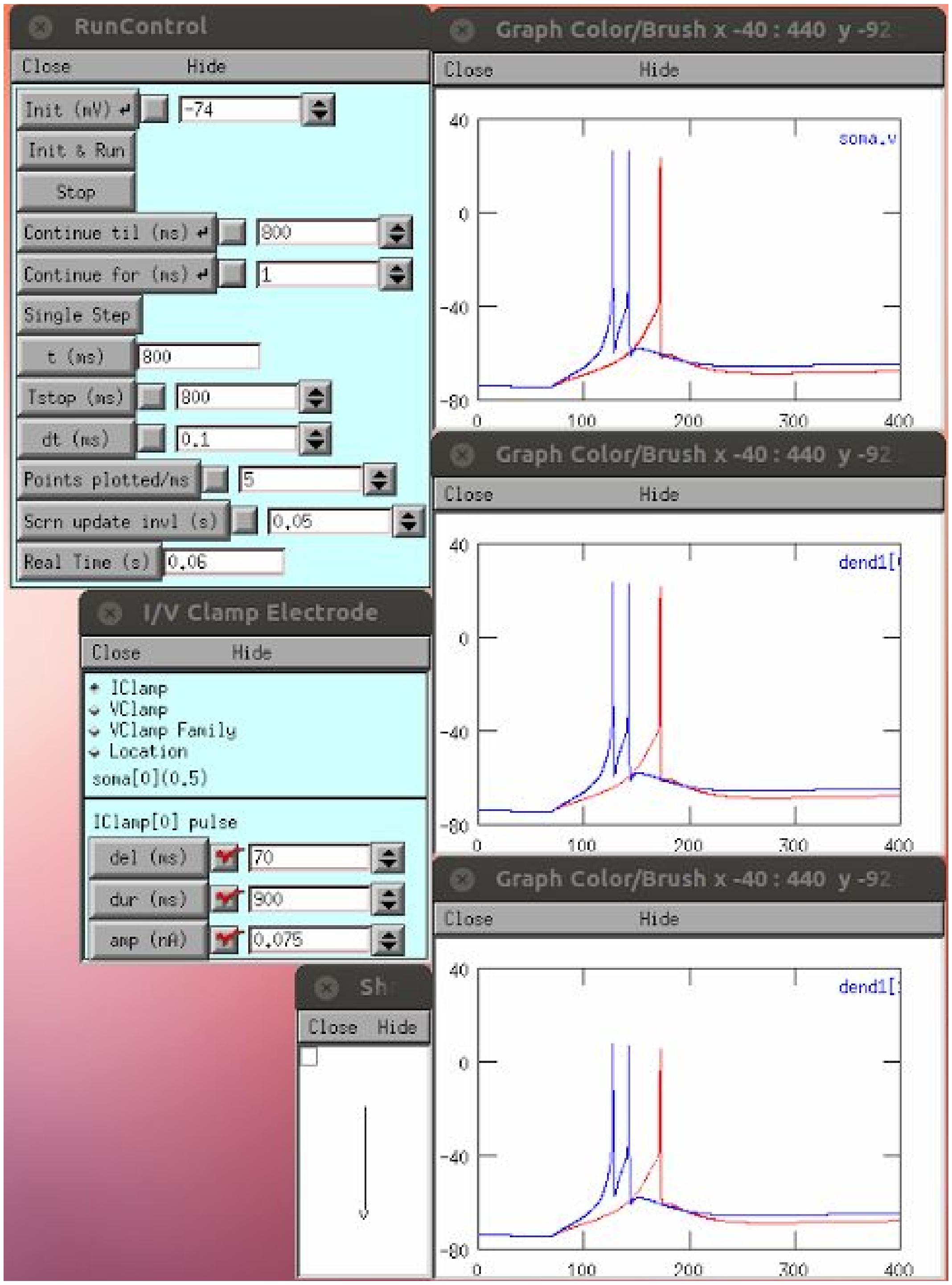
Close Hide



Graph Color/Brush x -40 : 440 y -92

Close Hide

Burst behavior in 3-compartment model:



RunControl

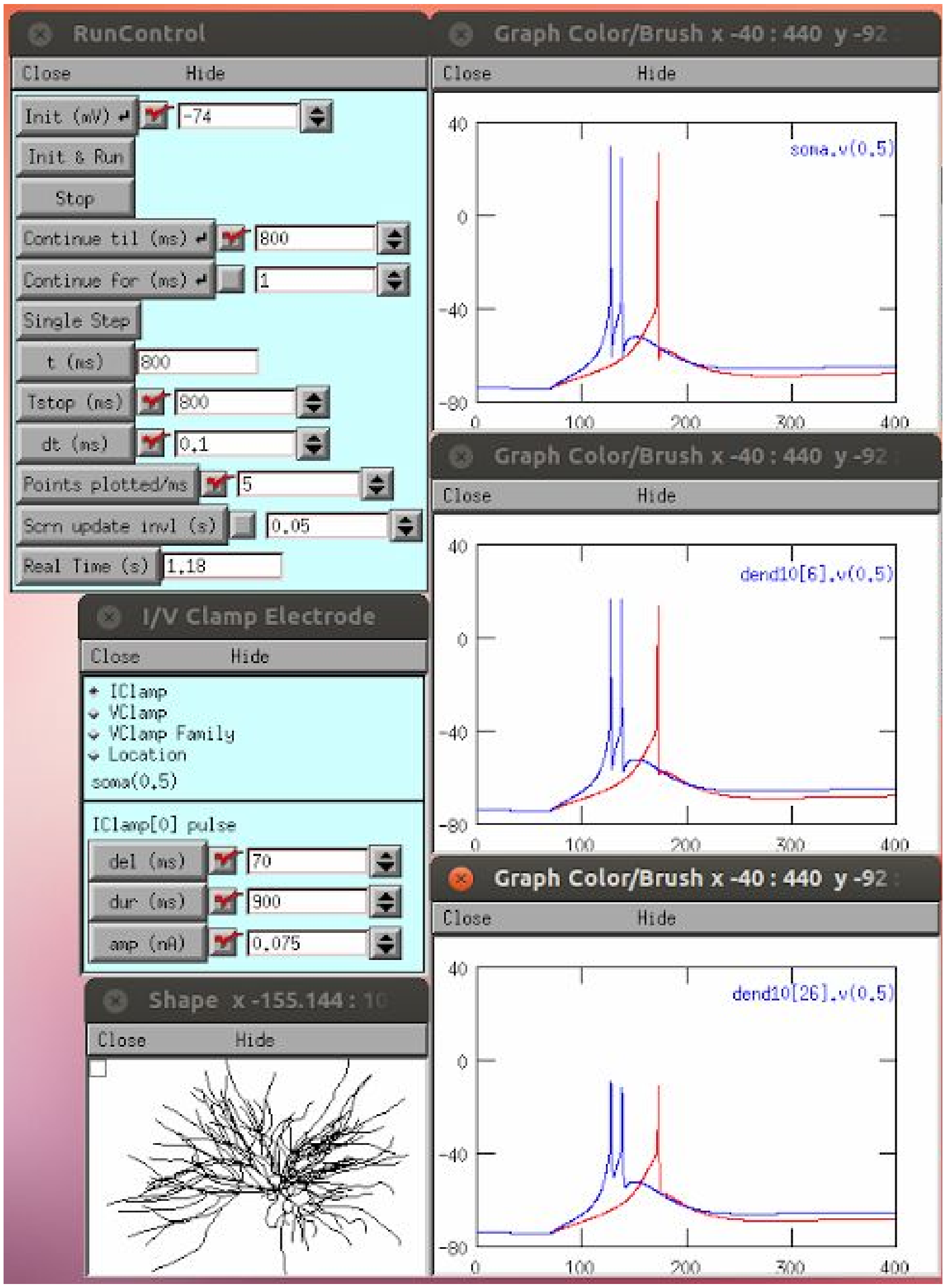
Init (mV) -74
Init & Run
Stop
Continue til (ms) 800
Continue for (ms) 1
Single Step
t (ms) 800
Tstop (ms) 800
dt (ms) 0.1
Points plotted/ms 5
Scrn update invl (s) 0.05
Real Time (s) 0.06

I/V Clamp Electrode

● IClamp
◇ VClamp
◇ VClamp Family
◇ Location
soma[0](0.5)

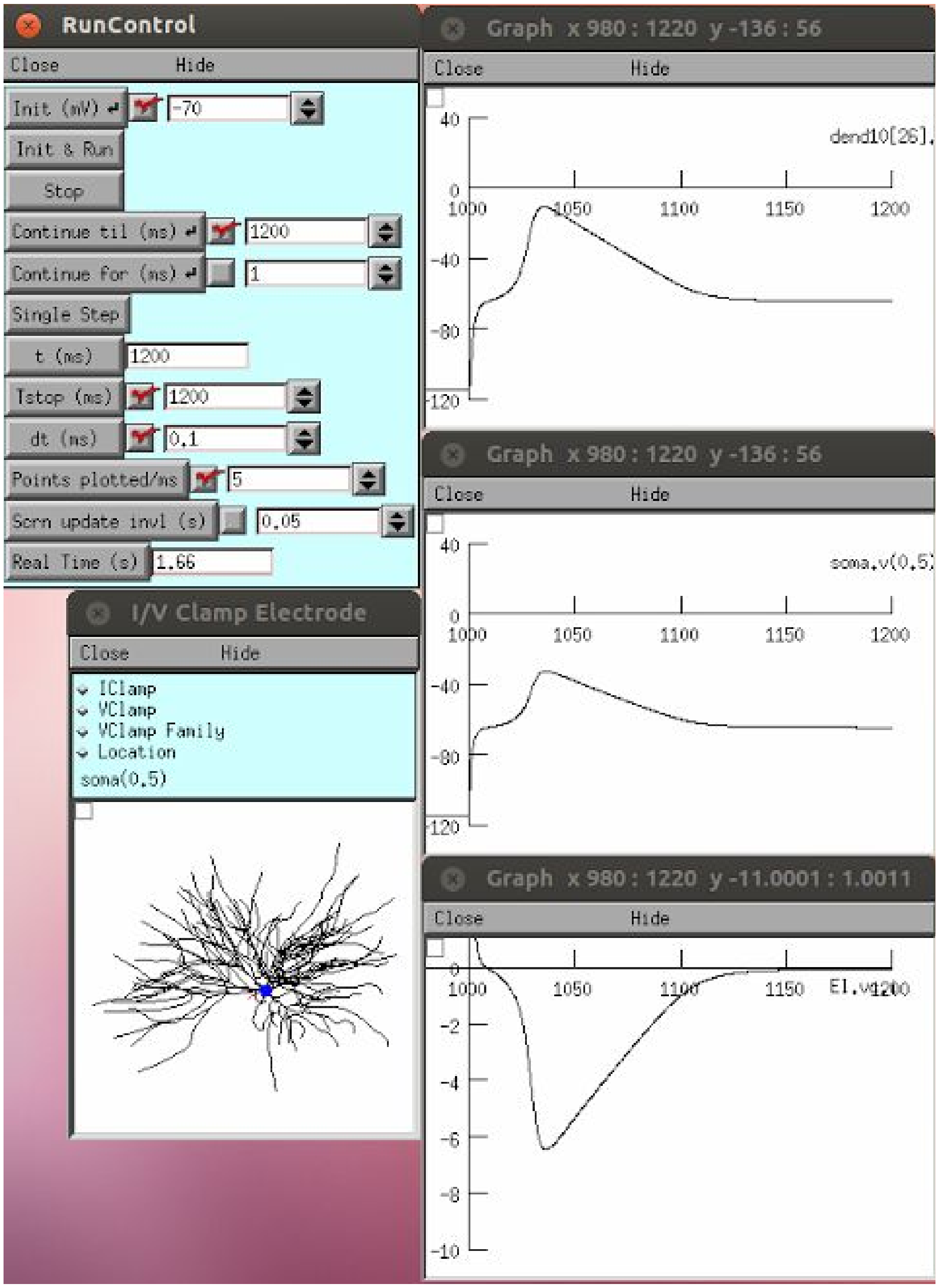
IClamp[0] pulse
del (ms) ✓ 70
dur (ms) ✓ 900
amp (nA) ✓ 0.075

Burst behavior in detailed cell model:


RunControl
Close Hide
Init (mV) -74
Init & Run
Stop
Continue til (ms) 800
Continue for (ms) 1
Single Step
t (ms) 800
Tstop (ms) 800
dt (ms) 0.1
Points plotted/ms 5
Scrn update invl (s) 0.05
Real Time (s) 1.18

I/V Clamp Electrode
Close Hide
IClamp
VClamp
VClamp Family
Location
soma(0.5)
IClamp[0] pulse
del (ms) 70
dur (ms) 900
amp (nA) 0.075

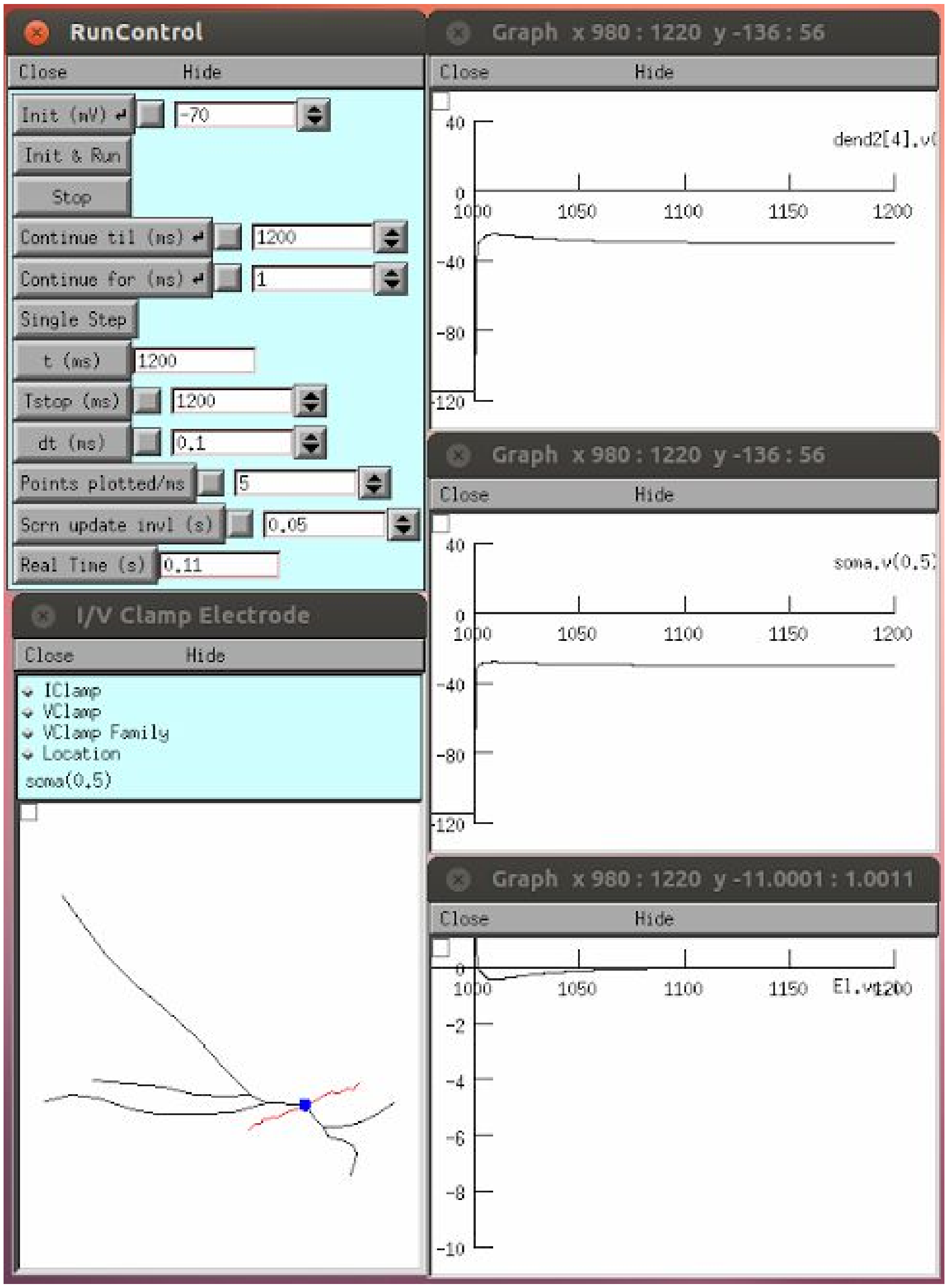
Voltage-clamp in detailed cell model:



Init (mV) -70
Continue til (ms) 1200
Continue for (ms) 1
t (ms) 1200
Tstop (ms) 1200
dt (ms) 0.1
Points plotted/ms 5
Scrn update invl (s) 0.05
Real Time (s) 1.66

soma(0.5)

Voltage-clamp in dissociated cell model:


RunControl

Init (mV) -70

Init & Run

Stop

Continue til (ms) 1200

Continue for (ms) 1

Single Step

t (ms) 1200

Tstop (ms) 1200

dt (ms) 0.1

Points plotted/ms 5

Scrn update invl (s) 0.05

Real Time (s) 0.11

I/V Clamp Electrode

IClamp

VClamp

VClamp Family

Location

soma(0.5)

Graph x 980 : 1220 y -136 : 56

Graph x 980 : 1220 y -136 : 56

Graph x 980 : 1220 y -11.0001 : 1.0011

4/1/2016~4/3/2016

<u>Read Christine's thesis Ch 3 ("GABA Transporters at the Reticulothalamic Synapse Shape Thalamic Oscillations")</u>

- Drugs:

bicuculline	GABA_A antagonist	10 μM
NO-711	GAT1 blocker	4 μM
SNAP-5114	GAT3 blocker	100 μM
tetradotoxin citrate (TTX)	Na channel blocker	1 μM
TTA-P2	T-type Ca channel (I_T) blocker	1 μM
ZD7288	I_h blocker	50 μM

- TC single cell model:

	length (μm)	diameter (μm)
soma	38.4	26
proximal dendrite	12.5	10.3
middle dendrite	26.59	8.5
distal dendrite	58.08	8.5

  C_m = 0.789 μF/cm²
  cytoplasmic (axial) resistivity (R_a) = 173 Ω cm
  temperature = 33 °C
  Correction factor C_d = 7.954
  Currents: passive leak current (**I_leak**), low-threshold calcium current (**I_T**), hyperpolarization-activated cationic current (**I_h**), fast transient A-type potassium current (**I_A**), and persistent sodium current (**I_NaP**)
- Network model:
  Two 1-dimensional layers: One TC layer of 100 cells and one RT layer of 100 cells, each on a straight line
  TC cells: Modeled as above with the addition of Hodgkin-Huxley type fast Na and K currents
  RT cells: Single-compartment model developed in Destexhe et al 1996 (total membrane area of 14,260 μm², contains a passive leak current (**I_leak**), low-threshold calcium current (**I_T**), Ca-dependent potassium current (**I_KCa**), and Hodgkin-Huxley type fast Na and K currents
  Synapses: Excitatory (AMPA) from TC to RT; Inhibitory (GABA_B) from RT to TC (2

types: diffuse “tickler” & strong “cluster”); connections were sparse with a distance-dependent distribution

- Oscillation spike detection:
  spikes: slope deflections greater than 3 times the threshold, which was defined as the root-mean-square of background noise of baseline sweeps.
  bursts: clusters of 5 spikes with <10 ms inter-spike interval
  oscillations: clusters of 3 bursts with <1 s inter-burst interval
- *In vitro* current clamp recordings of ventrobasal TC cells in slices of a P11 Sprague-Dawley rat. 3 Hz bursts evoked in aCSF containing bicuculline:
  (1) GAT1 blockade increased oscillation **duration** only
  (2) GAT3 blockade increased oscillation **duration & period**
  (3) GAT1 & GAT3 dual blockade **suppressed** oscillation
- Dynamic clamp experiments:
  Template conductance waveforms of GAT-modulated GABA_B IPSCs (input from RT neurons) were applied to ventrobasal TC cells and resulted in post-inhibitory rebound bursting that was generally stereotyped across experiments.
  (1) GAT1 blockade & GAT3 blockade **increased** burst **probability**
  (2) GAT1 & GAT3 dual blockade **decreased** burst **probability** (likely due to the very slow GABA_B IPSC decay resulting in the closure of T-current inactivation gate by the time the cell depolarized sufficiently to open the activation gate, thus preventing LTS bursts from occurring)
  (3) GAT3 blockade & GAT1 & GAT3 dual blockade increased burst **onset time**
- For some GAT3 blockade trials, **double bursting** occurred (“likely due to sufficient continued hyperpolarization caused by the slow GABAB IPSC decay causing sufficient de-inactivation of the T-channels (i.e. open probability of hT gate) to induce a second burst.”
- Model optimization:
  Current clamp recordings (steps of current injections) are made for ventrobasal TC cells sequentially applied with regents blocking:
  (1) action potentials
  (2) action potentials + I_T
  (3) action potentials + I_T + I_h
  Parameters of the TC single cell model was optimized in the following sequence:
  data of (3): I_leak variables, distal dendritic length, C_m
  data of (2): I_h variables
  data of (1): I_T variables
- The model TC cell recapitulated **low-threshold spike (LTS) bursting behavior** observed in dynamic clamp experiments when the following conditions were applied:
  (i) high density of T-channels in dendritic compartments for all-or-nothing
  (ii) a hyperpolarizing shift in the activation and inactivation curve for I_T as well as a steeper slope of the activation curve
  (iii) a depolarizing shift in the activation curve for I_h
- The network model “showed similar characteristics to the experimental data:”

(1) the GAT1 blockade & GAT3 blockade had **longer oscillation durations**
(2) GAT1 & GAT3 dual blockade had **truncated oscillations**

<u>PLAN FOR NEXT WEEK:</u>

- Finish the NEURON Book
- Understand Destexhe et al 1998 model
- Understand Destexhe et al 1996 model