I_{Kr} protocol comparison

Michael Clerx[a]

[a]*Computational Biology, Department of Computer Science, University of Oxford, Oxford, UK*

[b]*Research IT Services, University College London, London, UK*

[c]*Centre for Mathematical Medicine & Biology, School of Mathematical Sciences, University of Nottingham, Nottingham, UK*

Abstract

Comparing fits to Sine wave protocol with whole trace fits to traditional protocols and summary-statistic fits to traditional protocols.

1. Matching my simulations to Kylie's

Reproducing, not replicating, Kylie's work (Beattie et al., 2017) in Myokit (Clerx et al., 2016) and Pints [1]. Both use CVODE for simulations (Hindmarsh et al., 2005).

1.1. Loading sine wave data

Using data from **Cell 5** (16713110) as found on GitHub [2]. Data was already Dofetilide-subtracted and leak-subtracted by Kylie. All unnormalised data, so given in nA rather than A/F. The distance between data points was 0.1ms. Loaded data and converted to CSV. No capacitance artefact filtering was applied at this stage (except to create the figure below).

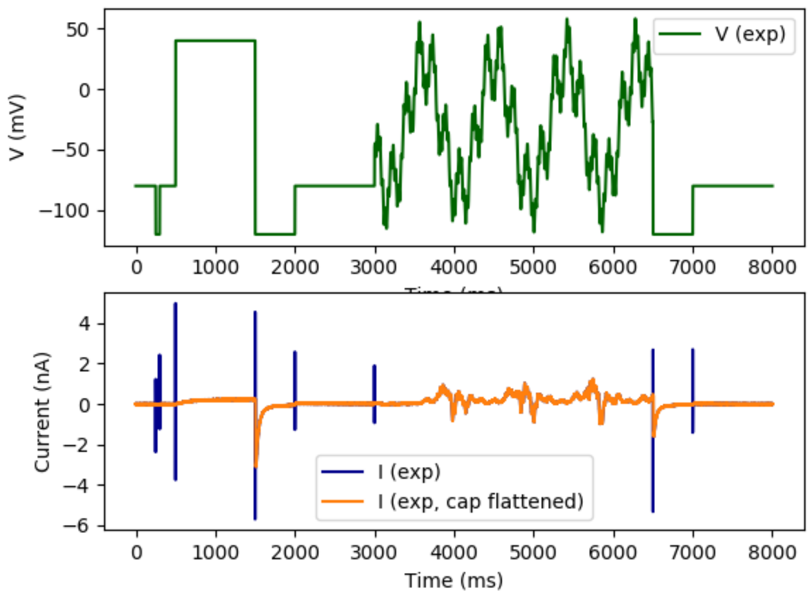

Figure 1: Sine wave data shown uncorrected (blue, passed to modelling stage) and corrected (orange).

1.2. Simulating with Kylie's model

Implemented Kylie's model in Myokit in two versions: one in Hodgkin-Huxley (HH) style (with two gating variables) and one in a Markov-model style (with four states y_1, y_2, y_3, y_4 where $y_{1,2,3}$ are described by ODEs while $y_4 = 1 - y_1 - y - 2 - y_3$. Used the **Markov** version throughout (to match Kylie's Matlab work).

```
IKr = p9 * y3 * (V - nernst.EK)
y4 = 1 - y1 - y2 - y3
k12 = p1 * exp( p2 * V)
k21 = p3 * exp(-p4 * V)
k41 = p5 * exp( p6 * V)
k14 = p7 * exp(-p8 * V)
dot(y1) = -k12 * y1 + k21 * y2 + k41 * y4 - k14 * y1
dot(y2) = -k14 * y2 + k41 * y3 + k12 * y1 - k21 * y2
dot(y3) = -k21 * y3 + k12 * y4 + k14 * y2 - k41 * y3
```

[1]https://github.com/pints-team/pints
[2]https://github.com/mirams/sine-wave

```
p1 = 2.26e-4 [1/ms]
p2 = 0.06990 [1/mV]
p3 = 3.45e-5 [1/ms]
p4 = 0.05462 [1/mV]
p5 = 0.08730 [1/ms]
p6 = 8.91e-3 [1/mV]
p7 = 5.15e-3 [1/ms]
p8 = 0.03158 [1/mV]
p9 = 0.15240 [mS/uF]
```

Updated the Matlab code from GitHub to extract the parameters with more precision. The results are given in Table 1.

Table 1: Kylie's parameters

Name	**Value**
p_1	2.26026076650526008e-04
p_2	6.99168845608636041e-02
p_3	3.44809941106439982e-05
p_4	5.46144197845310972e-02
p_5	8.73240559379589998e-02
p_6	8.91302005497139962e-03
p_7	5.15112582976275015e-03
p_8	3.15833911359110001e-02
p_9	1.52395993652347989e-01

The protocol was implemented in Myokit in a two-step process.. First, voltage steps were applied using Myokit's step protcol functionality, using the steps given in Table 2.

Table 2: Voltage steps in sine wave protocol. Note the 0.1ms offset.

Start (ms)	**Duration (ms)**	**Potential (mV)**
0	250.1	-80
250.1	50	-120
300.1	200	-80
500.1	1000	40
1500.1	500	-120
2000.1	1000	-80
3000.1	3500	-30
6500.1	500	-120
7000.1	1000	-80

Secondly, the model was amended to switch between the step protocol (represented by the variable `engine.pace`) and a sine wave:

```
model.get('membrane.V').set_rhs(
    'if(engine.time >= 3000.1 and engine.time < 6500.1,'
    + ' - 30'
    + ' + 54 * sin(0.007 * (engine.time - 2500.1))'
    + ' + 26 * sin(0.037 * (engine.time - 2500.1))'
    + ' + 10 * sin(0.190 * (engine.time - 2500.1))'
    + ', engine.pace)')
```

At this point, capacitance filtering was also applied, using a similar strategy to Kylie's code, in which data points from the first 5ms after each voltage transition were removed from the data set used to perform fitting. For example, given a time series $(t_1, x_1), (t_2, x_2), ..., (t_n, x_n)$ such a filter could omit all data from t_{11} to t_{15}, resulting in the filtered series $(t_1, x_1), (t_2, x_2), ..., (t_{10}, 10_x), (t_{16}, x_{16}), ..., (t_n, x_n)$. Alternative strategies, such as setting $x_{11}, x_{12}, ..., x_{15}$ to some estimated value (e.g. zero) all require some assumptions to be made, so simply ignoring these data points is the simplest and safest option.

1.2.1. Solver settings

Solver tolerances were set to 10^{-8}, 10^{-8}, to match Kylie's simulations. Note that the the tolerances are checked w.r.t. the states, not the output variable I_{Kr}. Since the states $y_i \leq 1$, the relative tolerance is usually the more stringent of the two. Since I_{Kr} is between 0 and 120 times g (≈ 0.1) times y_3, it seems the error in I_{Kr} will be between 0 and 12 times that in y_3 so $O(10^{-7})$. Finally, this is a per-step tolerance, not a global tolerance. A a rule of thumb, The CVODE manual suggests setting the tolerance to 0.01 times the desired global tolerance, suggesting an error of $O(10^{-5})$ in I_{Kr} might be accepted.

Both Myokit and Kylie's code use the `CV_BDF` and `CV_NEWTON` methods. A difference between the Myokit simulation and the one used by Kylie is that Myokit resets the solver at each voltage step transition (but not during the sine wave protocol). The advantage of this method is that you don't need to set a maximum step size (Kylie's simulation uses a max step size of 0.1ms).

The Matlab code makes repeated calls (one for each 0.1ms interval) to CVODE, using the `CV_NORMAL` calling mode. This asks the solver to step *to or just over* the next point in time, and returns the value at the requested time via interpolation. Note that this is independent of the maximum step size, so in the Matlab code the solver might move from $t = 0$ms to $t = 1$ms via the points 0ms, 0.7ms, 1.7ms with an interpolation to get the result at $t = 1$ms. The Myokit code makes calls using the `CV_ONE_STEP` mode, which makes steps of an arbitrary length. As soon as the next logging point is passed, Myokit then asks CVODE to return the values at the desired time via interpolation. Because no maximum step size is given, the example above could become 0ms, 0.7ms, 2.4ms with an interpolation to get the result at $t = 1$ms. So in effect both methods step to arbitrary points beyond the next requested time and then deliver an answer using interpolation.

The major difference between the two methods are that (1) Myokit regularly resets the solver, causing small steps to be made just after each transition and (2) the Matlab code enforces a maximum step size of 1ms.

1.2.2. How well do the Matlab/Myokit protocol implementations match?

To check how well the Myokit protocol matched the Matlab one, I compared against Kylie's protocol files. Because the data supplied on GitHub was used to generate figures and didn't show all digits used internally, I generated new files with higher precision output. Fig. 2 shows the difference in the membrane potential between the resulting Matlab protocol file and the membrane potential simulated in Myokit. Some error still occurs during the sine-wave, but this is on the order of numerical error (typically given as $O(10^{15})$ for double precision).

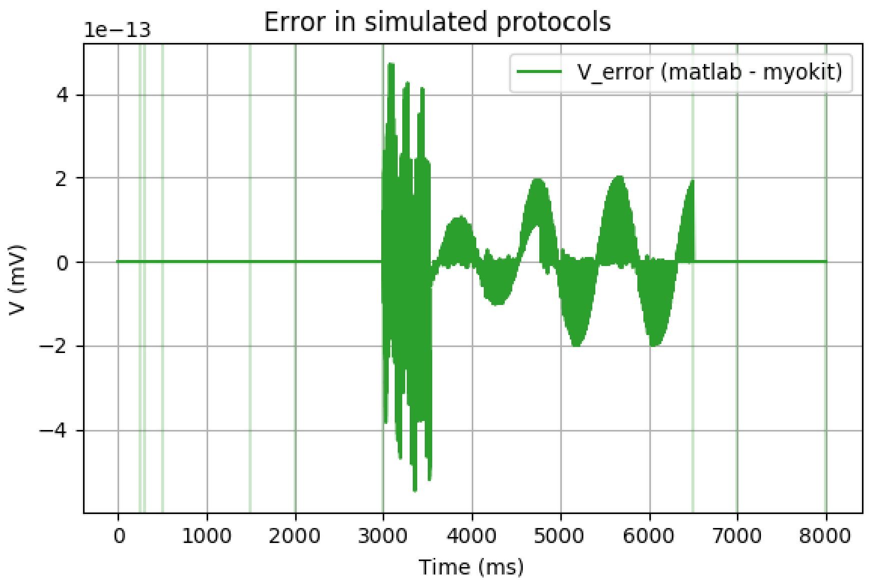

Figure 2: Difference between the Matlab and Myokit implementations of the voltage protocol. The maximum difference at any moment in time is 6e-13 mV.

1.2.3. And what about the simulations?

To check simulation results, I extracted new high-precision data files from the Matlab implementation and plotted them in Fig. 3. At first glance, the difference between the implementations seems to be relatively large.

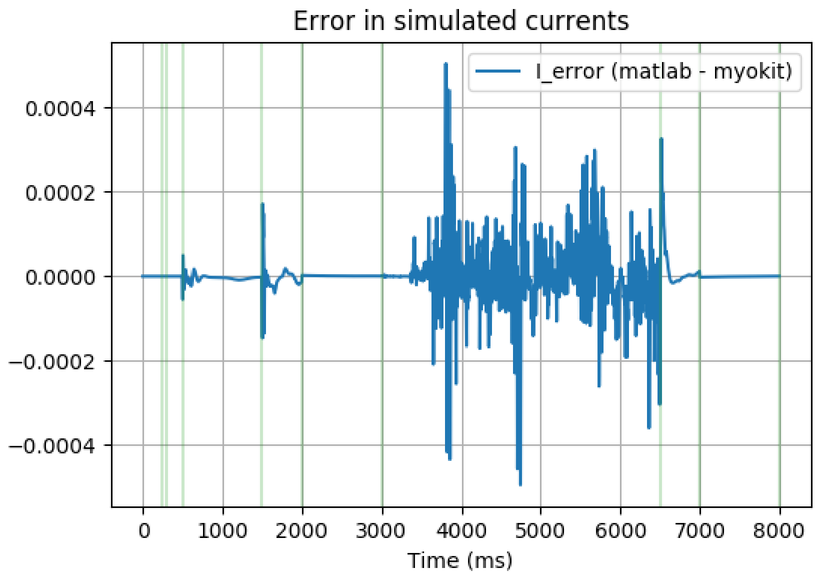

Figure 3: Difference between the Matlab and Myokit simulations of current during the voltage protocol, both with tolerances of 10^-8. The same capacitance filtering was applied to both traces. The maximum difference at any moment in time is 5e-4 nA.

To check if this was due to a difference in the model, or simply an accumulation of small numerical differences, I re-ran *both simulations* with higher tolerances (10^{-10}, 10^{-10}). The result is shown in Fig. 4.

This 10^4 increase in precision after a 10^2 increase in tolerance suggests that the errors in the signals are not systematic.

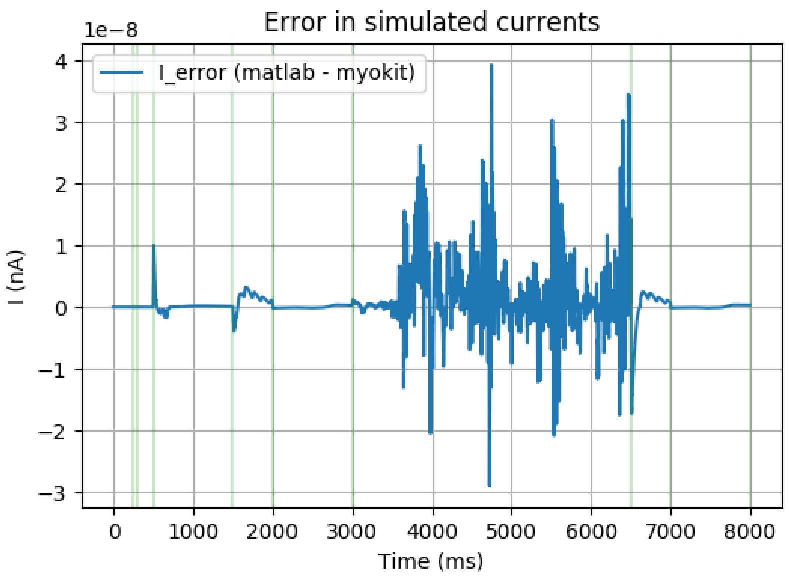

Figure 4: Difference between the Matlab and Myokit simulations of current during the voltage protocol, both with tolerances of $10^{-}10$. The same capacitance filtering was applied to both traces. The maximum difference at any moment in time is 4e-8 nA.

Adding a maximum step size of 0.1 reduces the difference further, as seen in Fig. 5.

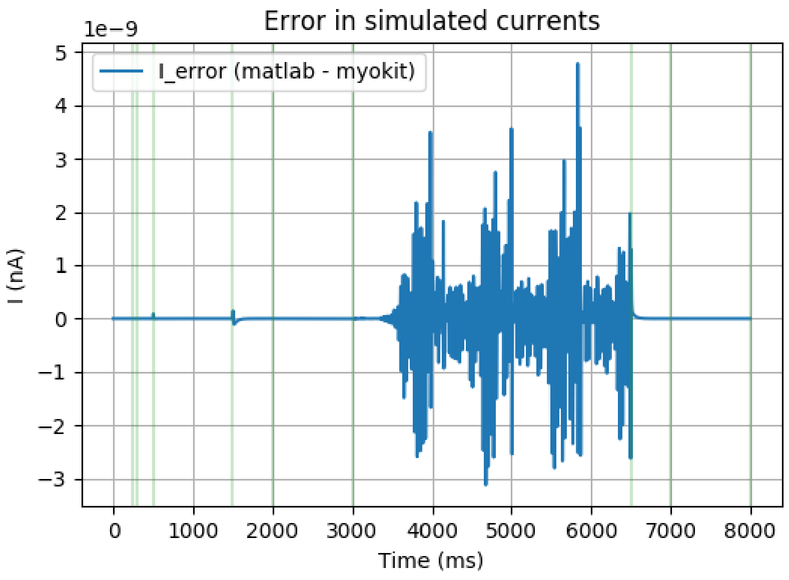

Figure 5: Difference between the Matlab and Myokit simulations of current during the voltage protocol, both with tolerances of $10^{-}10$ and a maximum step size of 1.0ms. The same capacitance filtering was applied to both traces. The maximum difference at any moment in time is 5e-9 nA.

It seems that the difference at this point is largely due to the difference in the input protocol.

1.2.4. Log-likelihood calculation

I next calculated the log-likelihood of the data, given Kylie's parameters and stochastic gaussian noise. The standard deviation of the noise was determined as the standard deviation of the first 200ms (2000 samples) of data, and is given in Table 3.

Table 3: Noise in the first 200ms.

σ_{noise} (nA)
0.00462852386082

The log-likelihood was then calculated using Pints (and Myokit for simulations) and compared with a value calculated with a simplified version of Kylie's Matlab code (on another machine). Both results are shown in Table 4.

Table 4: Log-likelihood of Cell 5 data given Kylie's cell 5 parameters. Pints/Myokit vs. Matlab. All calculated with $10^{-}10$ solver tolerances (but on separate machines).

Method	Log-likelihood
Pints/Myokit	-1.510 4 1629778709938e+06
Matlab	-1.510 6 37413819120 e+06

I found a couple of reasons for this difference. (1) Matlab and NumPy have different defaults for the `std()` function. In particular, Matlab uses the unbiased $1/(n-1)$ estimation by default, which Python does not. (2) Kylie's capacitance filtering different very slightly from mine, starting 2 samples earlier and finishing 1 sample earlier. In addition, Kylie doesn't filter at the start of the sine wave. (3) Kylie's code performed a log10 transform, and then undid it again, leading to slightly different parameters. (4) As seen in the graphs above, the simulation data differs slightly between the Matlab and Myokit implementations.

Table 7 shows the *cumulative* effect of compensating for these differences. The top row shows the original results from Table 4. In the next row, Python is made to use a similar sample standard deviation calculation is Matlab. This appears to make things worse. However, the next row shows that if the capacitance filtering is also changed, the results grow much closer. Similarly, applying *only* the capacitance filtering change led to a worse result. In the 4th row, the Python code is made to perform a log10 transform and detransform. After this operation, the results agree to within 10 digits.

The final row shows the effects of omitting the Myokit simulations altogether. Interestingly, this still only gives 10 digits precision. Factors that might explain this final difference are: (1) Different implementations of numerical operators between Python/Matlab. (2) Differences in data loading between Matlab and Python (3) Differences between the machines the tests were run on.

Table 5: An exploration of the differences in Table 4.

Method	Log-likelihood
Pints/Myokit	-1.510 4 1629778709938e+06
Matlab	-1.510 6 37413819120 e+06
+fix std()	-1.5 0 950362278065225e+06
Matlab	-1.5 1 0637413819120 e+06
+fix cap filter	-1.51063741 5 35030073e+06
Matlab	-1.51063741 3 819120 e+06
+log10 transform	-1.510637413 2 5626755e+06
Matlab	-1.510637413 8 19120 e+06
+kylie's data	-1.510637413 3 1823613e+06
Matlab	-1.510637413 8 19120 e+06

Following up on the observation that the forward and backward log 10 transform makes a difference, I ran one more simulation with the transformed/untransformed parameters, shown in Fig. 6.

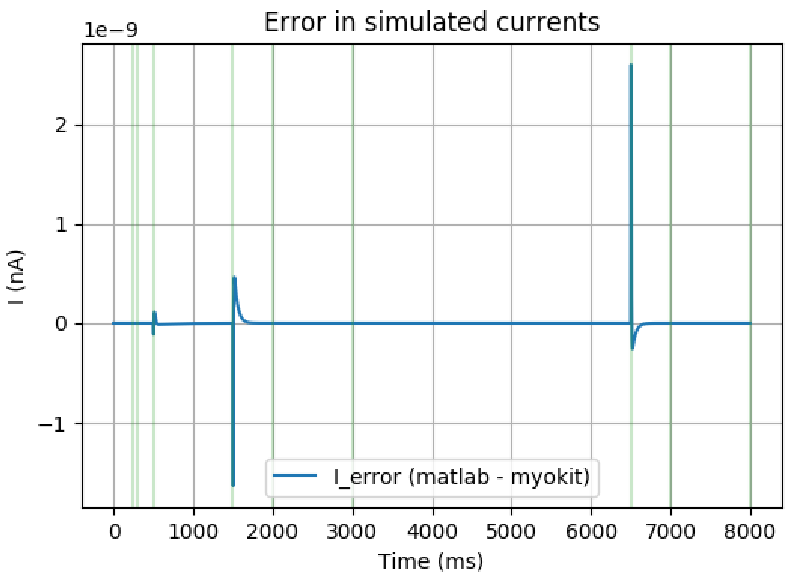

Figure 6: Difference between the Matlab and Myokit simulations of current during the voltage protocol, both with tolerances of 10^{-10}, a maximum step size of 1.0ms, and the parameters transformed and untransformed. The same capacitance filtering was applied to both traces. The maximum difference at any moment in time is 3e-9 nA.

As seen, this completely removes the error during the sine-wave part, leading only to differences at the voltage transitions (indicated with vertical green lines).

1.3. Conclusions

1. Simulating in Matlab and Myokit gives similar log likelihoods.
2. Differences in capacitance filtering and the `std()` method can lead to $O(10^{-4})$ differences in the calculated log-likelihood.
3. Because $O(10^{-4})$ isn't all that big, we should be able to continue using the methods used to create Table 4 (so Python `std()` and slightly different capacitance filtering from Kylie)

2. Finding the optimal parameters

Set up CMAES fitting in Pints, using the same model implementation as in the previous section. But: this time with tolerances 10^-8 and without the minimum step size.

2.1. Prior / boundaries / constraints

Kylie's model uses three types of constraint.

The first is a manually determined lower and upper bound on the conductance parameter (p_9).

The remaining eight parameters are divided into *as* (p_1, p_3, p_5, p_7) and *bs* (p_2, p_4, p_6, p_8). Each rate constant then has the form $r_i = a_j \exp(b_k V)$. The second constraint is:

$$10^{-7} < a < 10^{3}$$
$$10^{-7} < b < 0.4$$

The third constraint is on the rate parameters and on the the applied voltage V.

$$
\begin{aligned}
r1 &= p_1 \exp(p_2 V) \\
r2 &= p_3 \exp(-p_4 V) \\
r3 &= p_5 \exp(p_6 V) \\
r4 &= p_7 \exp(-p_8 V)
\end{aligned}
$$

This constraint says that *the maximum rate during the protocol* must be within certain physiological bounds. The minimum rate is unbounded (but ≥ 0 due to the previous constraints on the as and bs).

$$
\begin{aligned}
r_{\text{min}} &< \max_{V \in \{V_{\text{min}}, V_{\text{max}}\}} r(V) < r_{\text{max}} \\
r_{\text{min}} &= 1.67 \cdot 10^{-5} \text{ms}^{-1} \\
r_{\text{max}} &= 1000 \text{ms}^{-1} \\
V_{\text{min}} &= -120 \text{mV} \\
V_{\text{max}} &= 58.25 \text{mV}
\end{aligned}
$$

For the forward rates, r_1 and r_3 we find

$$
\begin{aligned}
\max\{ae^{bV_{\text{min}}}, ae^{bV_{\text{max}}}\} &> r_{\text{min}} && (1) \\
ae^{bV_{\text{max}}} &> r_{\text{min}} && (2) \\
b &> \frac{\log r_{\text{min}} - \log a}{V_{\text{max}}}, \qquad V_{\text{max}} > 0 && (3)
\end{aligned}
$$

and

$$
\begin{aligned}
\max\{ae^{bV_{\text{min}}}, ae^{bV_{\text{max}}}\} &< r_{\text{max}} && (4) \\
ae^{bV_{\text{max}}} &< r_{\text{max}} && (5) \\
b &< \frac{\log r_{\text{max}} - \log a}{V_{\text{max}}}, \qquad V_{\text{max}} > 0 && (6)
\end{aligned}
$$

For the backward rates, r_2 and r_4 we find

$$
\begin{aligned}
\max\{ae^{-bV_{\text{min}}}, ae^{-bV_{\text{max}}}\} &> r_{\text{min}} && (7) \\
ae^{-bV_{\text{min}}} &> r_{\text{min}} && (8) \\
b &> \frac{\log r_{\text{min}} - \log a}{V_{\text{min}}}, \qquad V_{\text{min}} < 0 && (9)
\end{aligned}
$$

and

$$
\begin{aligned}
\max\{ae^{-bV_{\text{min}}}, ae^{-bV_{\text{max}}}\} &< r_{\text{max}} && (10) \\
ae^{-bV_{\text{min}}} &< r_{\text{max}} && (11) \\
b &< \frac{\log r_{\text{max}} - \log a}{V_{\text{min}}}, \qquad V_{\text{min}} < 0 && (12)
\end{aligned}
$$

So the forward rates are bound by V_{max} and the backward rates are bound by V_{min}.

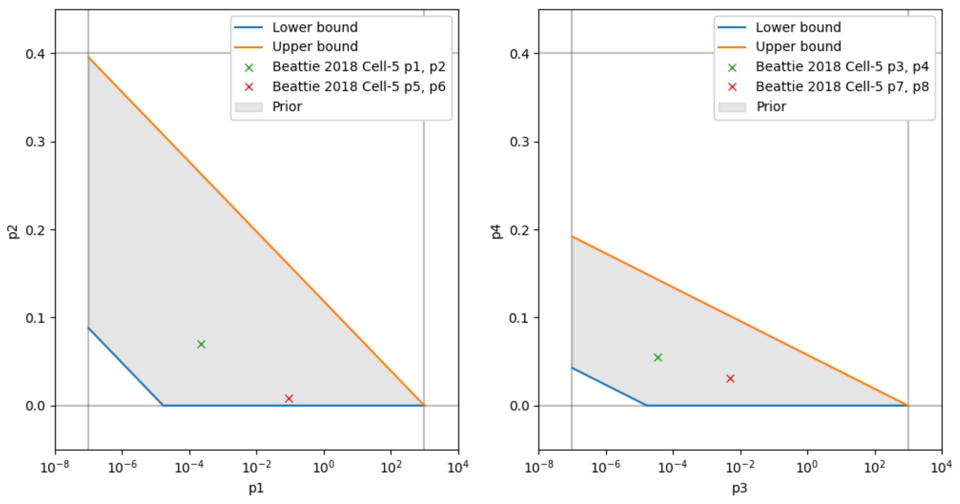

Figure 7: Prior on forward rates (left) and backward rates (right).

The full prior is then created by combining the constants on as, bs, and rs for each rate, and finally adding a lower and upper bound on the conductance. Values for the conductance bounds are taken from Kylie's code without modification.

Note: At the time of writing it seems that the Matlab code in the sine wave project only checks the rate constant constraints for the fastest rate in the model. This means that the upper bounds in Fig. 7 are enforced, but the lower bounds (from the rate constants, the diagonal bit in the bottom left corner) are only enforced for the fastest rate. In other words, some values are allowed end up in the little white triangle in the bottom left corner. This will be fixed soon, but shouldn't change the results as the current values are within the grey shaded area.

2.2. Selecting starting points

To run an optimisation, we need to select some starting points within the prior. Sampling within the boundaries on the parameters themselves is easy: just collect all lower and upper bounds and sample a 9d parameter vector uniformly within those bounds. However, this doesn't guarantee it meets the rate constant constraints.

The simplest strategy to deal with this is to do rejection-sampling (accept-reject): Sample points within the parameter boundaries, then check if they violate the rate constant constraints, and if so, reject them and sample again. If we sample a 9d point uniformly within the outer bounds, the chances of getting a point that meets the rate constant requirements are not good. Fig. 8 shows the prior in an untransformed space. The area's of the grey areas are approximately 0.043 (forward) and 0.021 (backward) times the area of the parameter bounds. This means that testing a single sample with two forward and two backward rates the chances of acceptance are approximately $0.043^2 \cdot 0.021^2 \approx 8.2 \cdot 10^-7$, so that an average of 1 in 1.2 million samples would be accepted.

Kylie's Matlab code gets around this by sampling uniformly in a much smaller space, with boundaries close to the minimum and maximum values found in models in the literature.

However, if we log transform the a parameters, our chances get much better. From Fig. 7 we can estimate the forward area is ≈ 0.5 times the total area, while the backward area is approx 0.25 times the total area. To further improve our chances we can split the problem into 5 subproblems: Twice sampling the parameters for a forward rate, twice sampling the parameters for a backward rate, and finally sampling a conductance value. This should raise our sampling success rate, leading to an expected number of $2 + 2 + 4 + 4 = 12$ samples needed to get a single accepted 9d point.

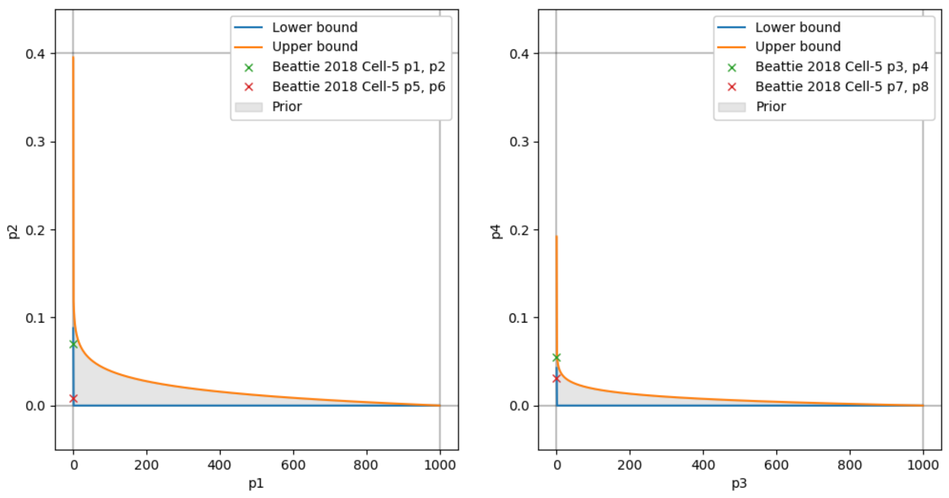

Figure 8: Prior on forward rates (left) and backward rates (right), shown without a logarithmic axes for p_1, p_5, p_3 and p_7.

2.3. Running an optimisation

Using Pints, which wraps around CMA-ES (as well as providing other optimisation algorithms and MCMC). Created a `KnownNoiseLogLikelihood` using a σ estimated from the first 2000 data points (200ms, when the signal is flat). To estimate it, I used the sample standard deviation with ddof=1 (the $n-1$ version). Then added a custom `LogPrior` object that implemented the parameter and rate constraints (as well as providing the sampling method described above). Combined these into a `LogPosterior` and passed that to the `Optimisation` class (which maximises it). Because the `LogPrior` evaluates to $-\infty$ when the constraints are violated, the problem is automatically bounded and so no `Boundaries` object was passed to the optimiser. Evaluation of the `LogPosterior` (which involves a simulation) was set to happen in parallel. CMAES was chosen as the optimisation method.

As stopping criteria we used the occurrence of 200 subsequent iterations without a significant change in the score function (threshold 1e-11). No other stopping criteria were applied.

The fits were then run on scrambler, which provides 24 real and 24 fake cores (hyperthreading), allowing 48 simulations to be run in parallel(ish, hyperthreading). A population size of 48 was used, leading to a core use of approx 80% each (perhaps we should double/triple the population size?). With these settings, scrambler runs around 4 iterations per second. 25 repeats were run (sequentially), all from different points sampled from the prior.

2.4. Results

Results from CMAES were highly similar to those found by Kylie.

Table 6: Results of parameter fitting, compared with published results by Kylie. The log-likelihoods shown were calculated using Myokit/Pints, in exactly the same manner but with a different parameter set.

Value	**Kylie**	**Myokit/Pints**
p_1	2.26 026076650526008e-04	2.26 137846758451367e-04
p_2	6.991 68845608636041e-02	6.991 30764903859586e-02
p_3	3.44 809941106439982e-05	3.44 958226788683007e-05
p_4	5.461 44197845310972e-02	5.461 21826895913237e-02
p_5	8.7324 0559379589998e-02	8.7324 2841545401188e-02
p_6	8.9 1302005497139962e-03	8.9 3017989292029316e-03
p_7	5.1 5112582976275015e-03	5.1 4883265963462632e-03
p_8	3.15 833911359110001e-02	3.15 620982414665449e-02
p_9	1.52 395993652347989e-01	1.52 436968881218465e-01
log-likelihood	-1.509 50112826760067e+06	-1.509 48474675446958e+06

3. Inferring a distribution around the optimal parameters

Started MCMC using adaptive covariance MCMC. 3 chains in parallel using Pints. All starting from position found by CMAES. Ran for 250 000 iterations, discarded first 50 000 as warm-up. No thinning was applied.

Inspected results of 3 chains: All kept to highly similar mean, found highly similar distribution. Continued analysis with only 1st chain.

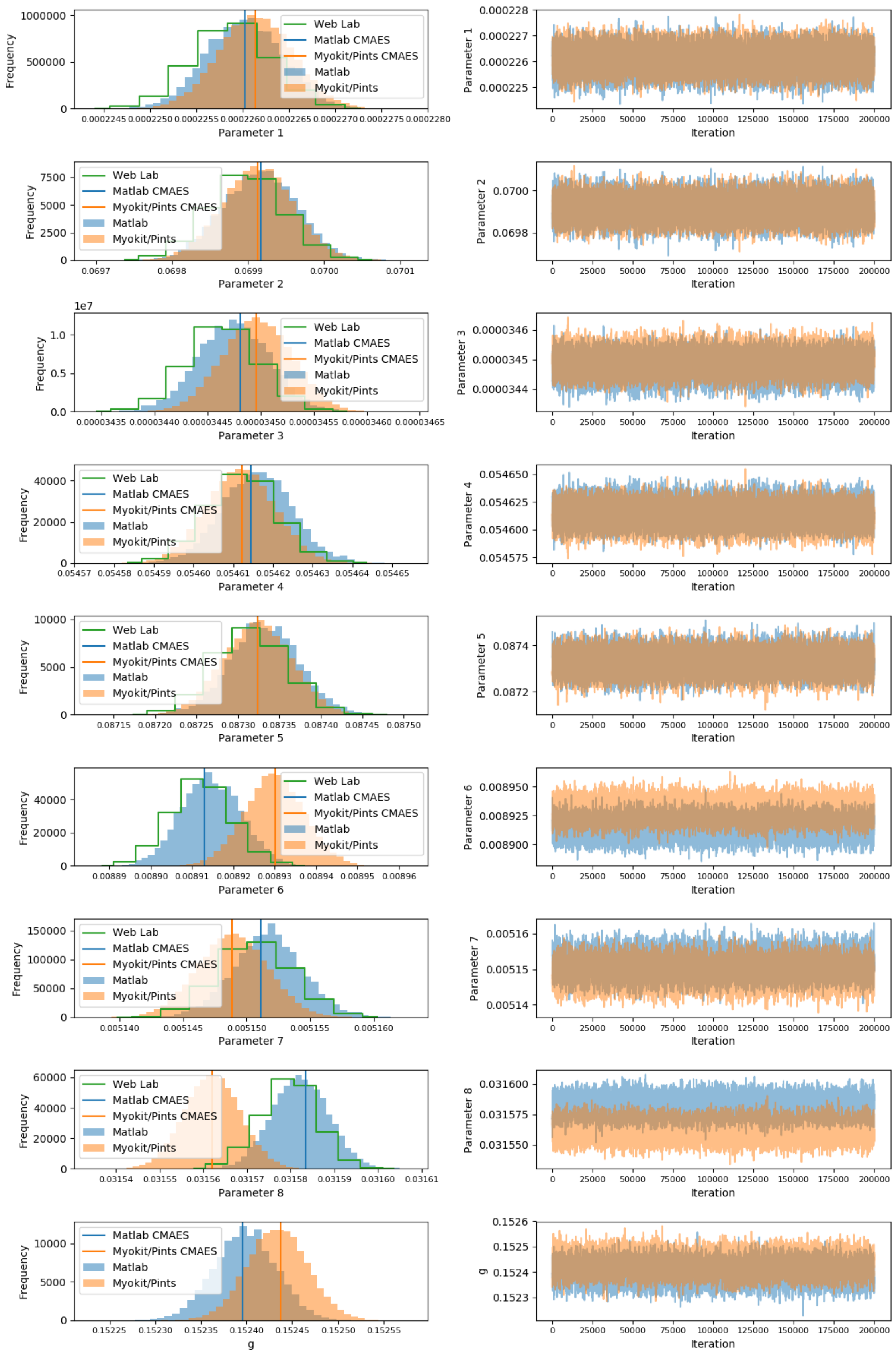

Figure 9: MCMC results comparison, Matlab, Web Lab, Pints/Myokit. The resulting distributions all have similar widths, but there are very slight (3d decimal) shifts in the means for some parameters. Plotting the CMAES results (vertical lines) reveals these small shifts are due to the same factors identified for the CMAES results.

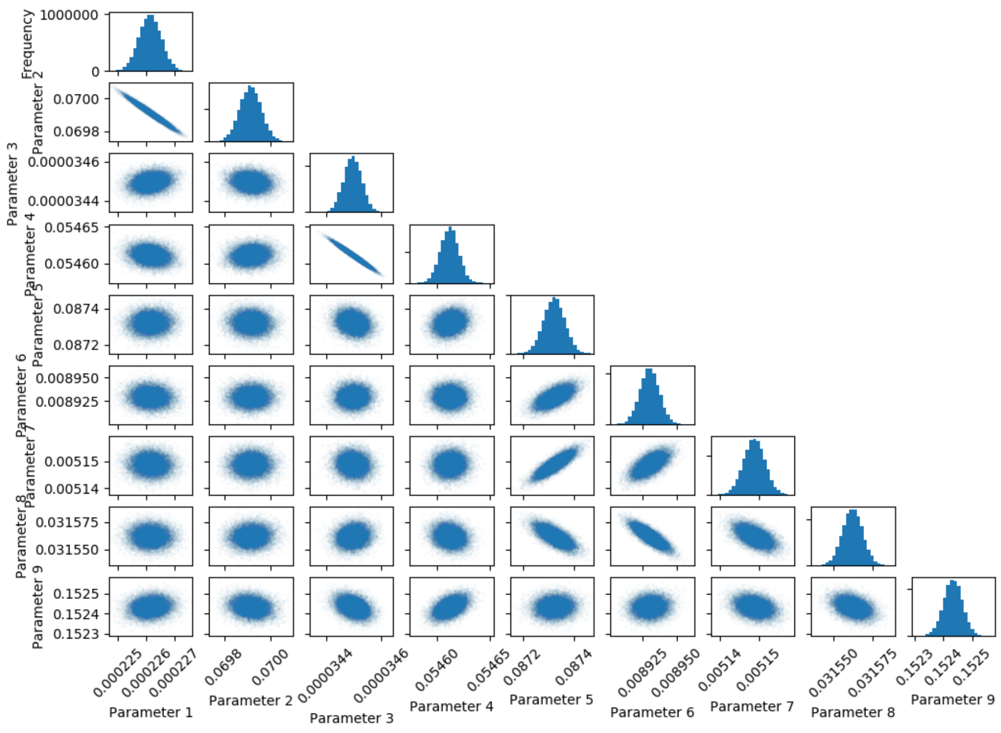
Figure 10: Pairwise plots of correlations between parameter values, as found by MCMC.

4. Fitting to multiple traces / Traditional protocols

Loaded the first activation protocol ("steady activation"). Again, the first step has one extra time point: adjusted the simulation protocol to match this. Results of importing the data shown in Fig. 11. Ran simulations with the parameters Kylie inferred from the sine wave experiments. The error between the simulated and measured current is considerable.

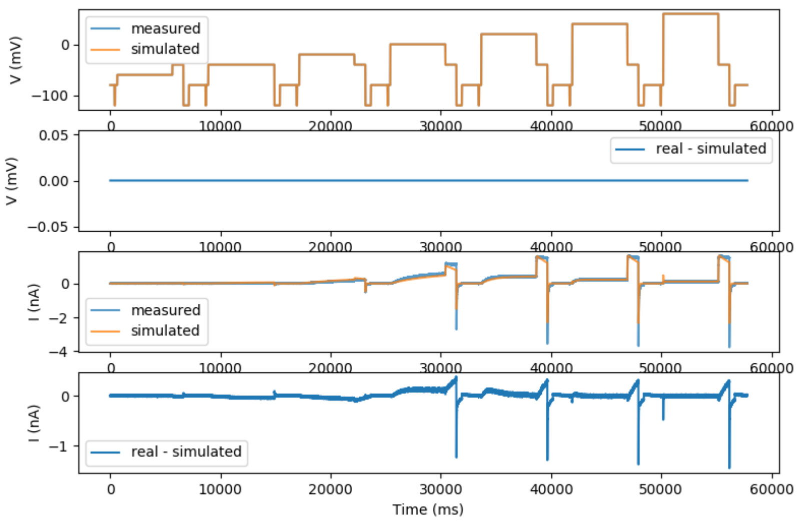

Figure 11: The first activation protocol (steady activation): data and simulation with Kylie's parameters from the sine-wave protocol.

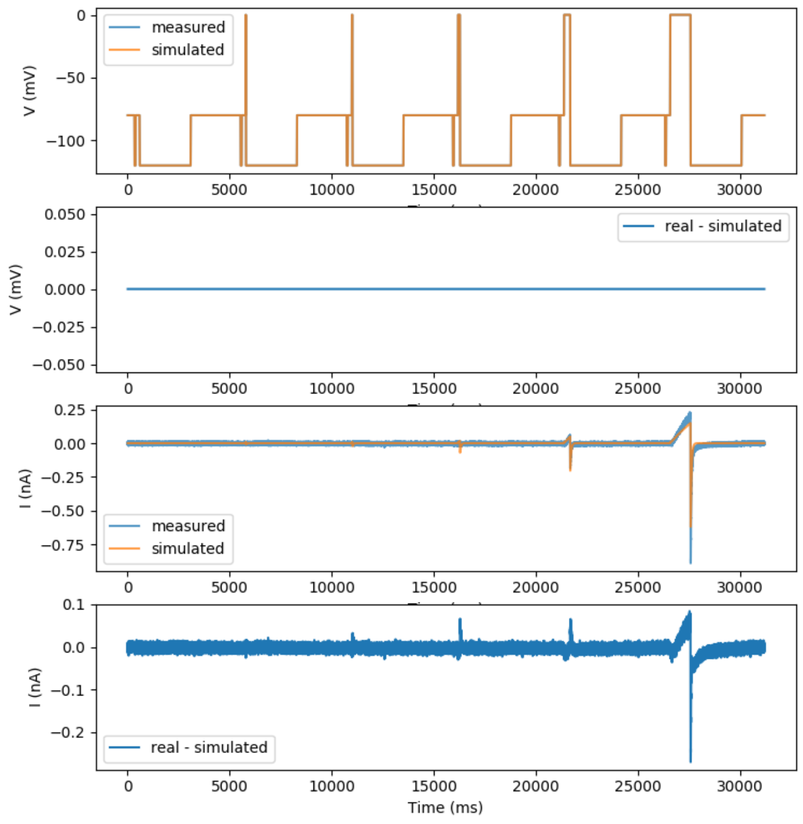

Figure 12: The second activation protocol (activation kinetics 1): data and simulation with Kylie's parameters from the sine-wave protocol.

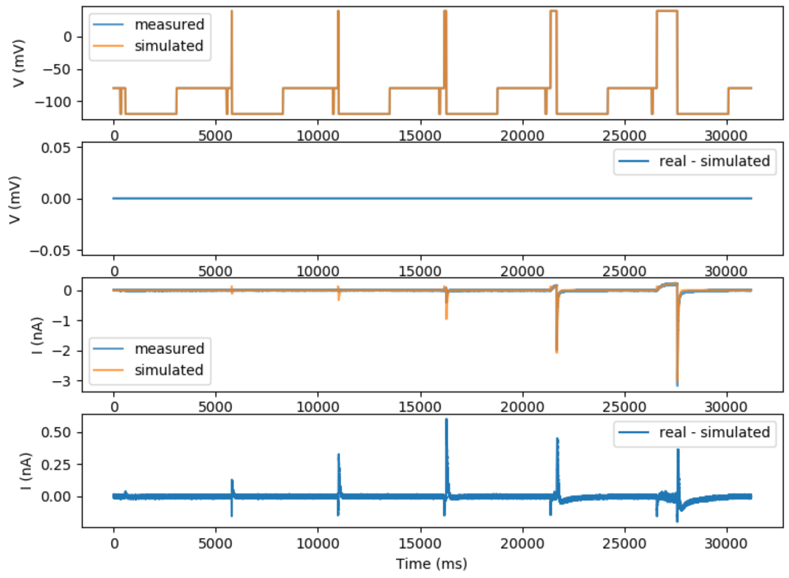

Figure 13: The third activation protocol (activation kinetics 2): data and simulation with Kylie's parameters from the sine-wave protocol.

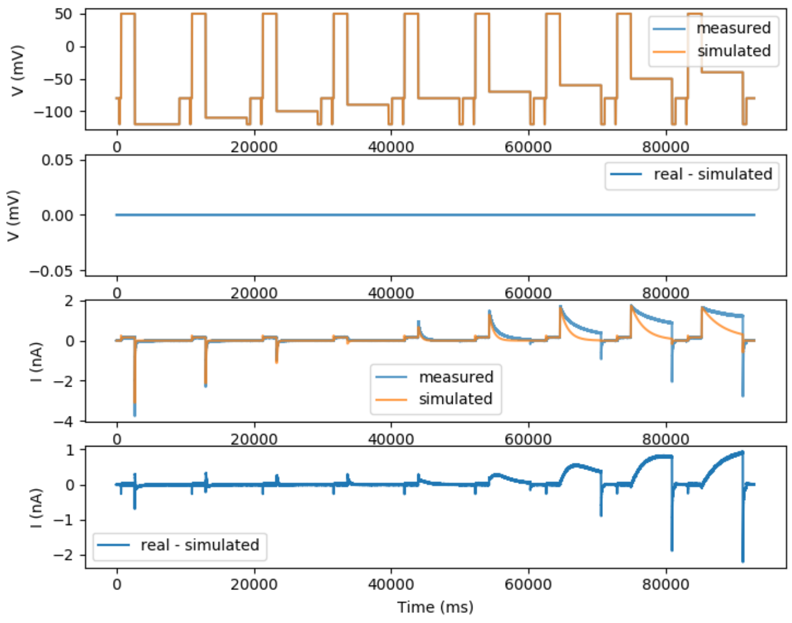

Figure 14: The deactivation protocol: data and simulation with Kylie's parameters from the sine-wave protocol.

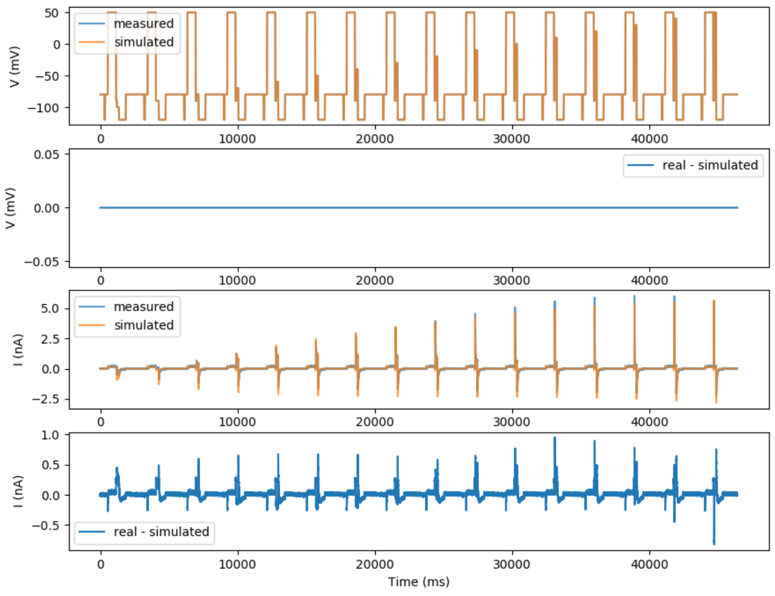

Figure 15: The inactivation protocol: data and simulation with Kylie's parameters from the sine-wave protocol.

4.1. Fitting to trad. prot.

Table 7: Results of parameter fitting, comparing sine wave results with results from fitting to whole traces using traditional protocols.

Value	**Sine wave**	**Traditional traces**
p_1	2.261e-4	2.364e-4
p_2	6.991e-2	**7**.563e-2
p_3	3.450e-5	3.313e-**6**
p_4	5.461e-2	**7**.391e-2
p_5	8.732e-2	**9**.255e-2
p_6	8.930e-3	8.508e-3
p_7	5.149e-3	**6**.648e-3
p_8	3.156e-2	3.098e-2
p_9	1.524e-1	1.292e-1

4.2. MCMC on trad. prot.

MCMC with same settings as previously. Runs into some issues without an initial covariance set. Had to use $I \cdot x_0 \cdot 10^{-35}$ to get any acceptance in the early stages. Seems very very tight, presumably due to massive number of samples.

5. Fitting to summary statistics

Analysing 5 protocols.

Versions used so far match *exactly* with the recorded signals, but have a few annoying issues with e.g. an extra sample added to the first step, or very slight variations in step size. Figures below show simulation-only results with 'tidied up' versions of same protocols. Simulations are performed with fitting results to combined whole traces for these protocols.

5.1. Pr1 and Pr2: Activation kinetics

Pr1 and Pr2 differ only in size of one step.

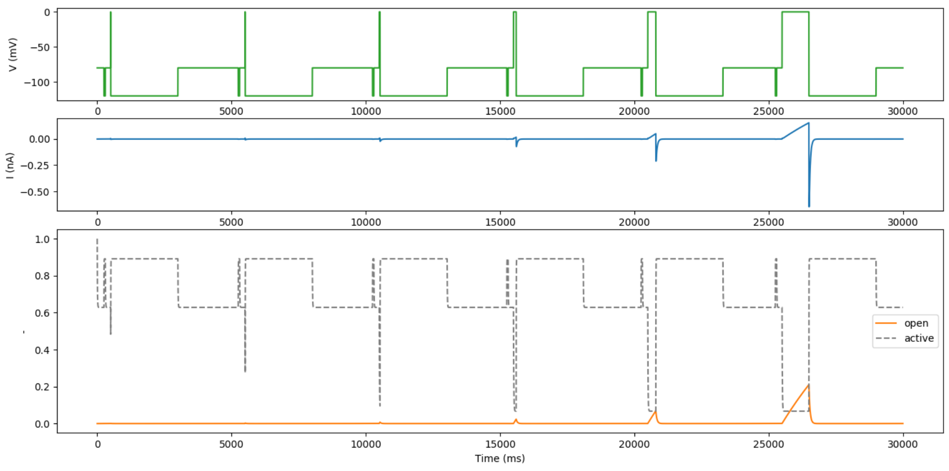

Figure 16: Pr1 (tidied)

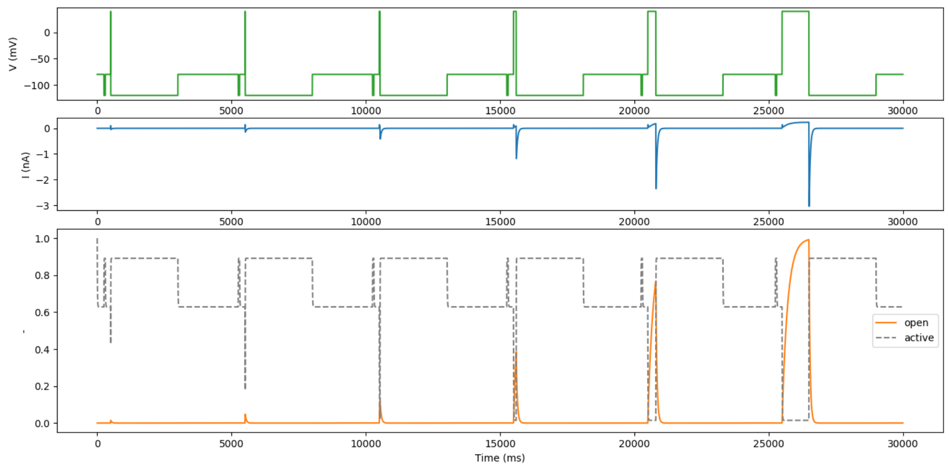

Figure 17: Pr2 (tidied)

Protocols are called “activation kinetics 1 & 2”, but what do they tell us about activation kinetics? Overlaying the different steps shows the (voltage-dependent) activation takes the same course at every (same-voltage) step (but note this is a model prediction, and more complicated behaviour might occur if a more complicated model was used).

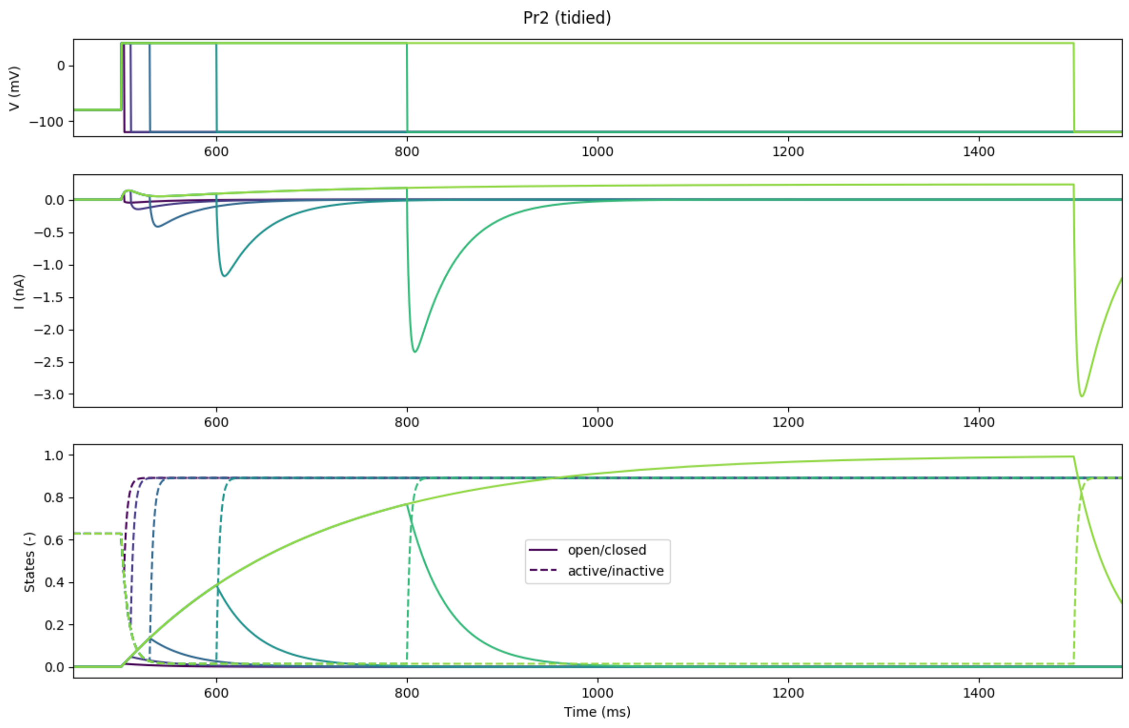

Figure 18: Pr2 (tidied, steps overlayed). Notice the currents at each step follow the same, slow upwards line, before dropping down at the end of the step. In other words, the activation is the same at each step (as expected given that they step to the same voltage, from the same pre-condition).

As a result, the full trace of the longest step provides all information encoded in the shorter steps. If we measure the current at the end of each step, this provides a low-sampling rate digitisation of the longest step. The difference between Pr1 and Pr2 is the height of this step (0mV vs 40mV), so that effectively Pr1 and Pr2 give us bad digitisations of the activation process at two different voltages.

Both steps are preceded by the same sequence of a short time at -80mV, then -120mV (to measure leak), then back to -80mV. Each repeat of the activation protocol Pr3 starts with the same sequence, followed by a very long (5s) step to several voltages, including 0mV and 40mV. As a result, the activation protocol Pr3 contains the longest step from Pr1 and the longest step from Pr2. So we can conclude that *using whole-trace fitting, the information we aim to extract with Pr1 and Pr2 is already contained in Pr3.*

Other parts of Pr1 and Pr2 might be interesting for aspects other than activation (although from the figures above it does not seem to be the case).

5.1.1. Pr1/2: Summary statistic

Pr1 can be used to derive 6 points of the activation curve for a step to 0mV, by taking the final value during the variable length step. Pr1 can be used to derive 6 points of the activation curve for a step to 40mV, by taking the final value during the variable length step.

We can try it out in simulation:

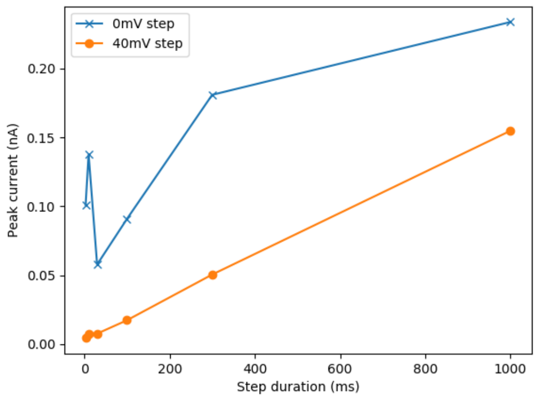

Figure 19: Pr1 and Pr2 summary statistic: Simulated peak current during variable step, versus step duration in Pr1 (0mV) and Pr2 (+40mV).

What happened here? To see why the lines are crooked near the start, we can inspect our simulation results, specifically the first milliseconds of the variable step:

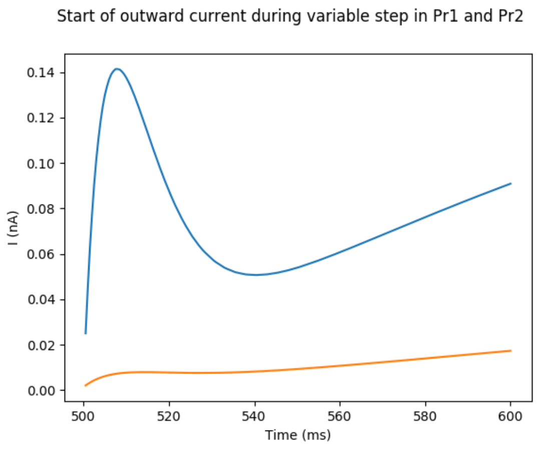

Figure 20: Current during first milliseconds of the longest variable step in Pr1 and Pr2.

So our model starts with a little hump, before settling into the gradual ascent more easily seen in the earlier figures. Looking back at the states, this is the result of inactivation (a dip down from approximately 0.6 to zero). Judging by its time course, this should affect the first 3 steps, which is borne out by the figure above.

If this effect is seen in real cells (but e.g. Zhou et al. (1998) doesn't show it), then this is an issue for this protocol.

5.1.2. Up for discussion

1. The 3ms step in Pr1 and Pr2 is filtered out completely by the 5ms capacitance filter.

2. Note that Kylie's paper says these protocols are *shortened* (so that they could all be performed together on the same cell, with and without dofetilide). So might need to add points for fairer comparison. On the other hand, this way we can do everything on one cell, which is not usually done.

3. Another difference with a standard analysis is that this would use an average of multiple cells. This might 'contaminate' the data with cell-to-cell variability, but could also reduce noise. Perhaps it'd be

best for us to take a few points near each point we want to measure (e.g. a few points of “peak current”) and take the mean of those, to reduce the influence of noise.

5.1.3. *Pr1: Data from cell 5*

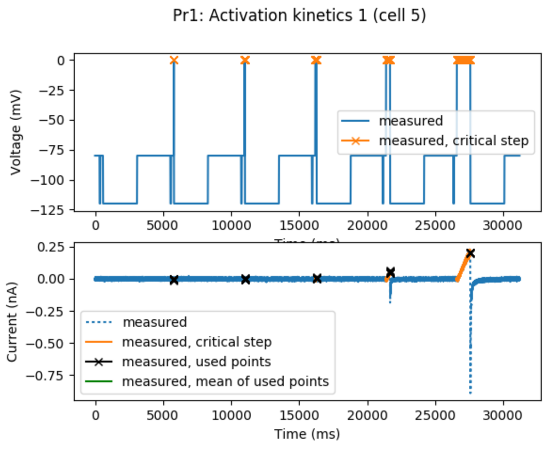

Figure 21: Current in cell 5 during Pr1

Fig. 21 shows that the real data isn’t terribly exciting during Pr1, except for the final two steps. As noted above, the first step is even removed entirely by the capacitance filtering. However, step 2 shows that nothing happens at this duration, so we probably don’t need to worry about that.

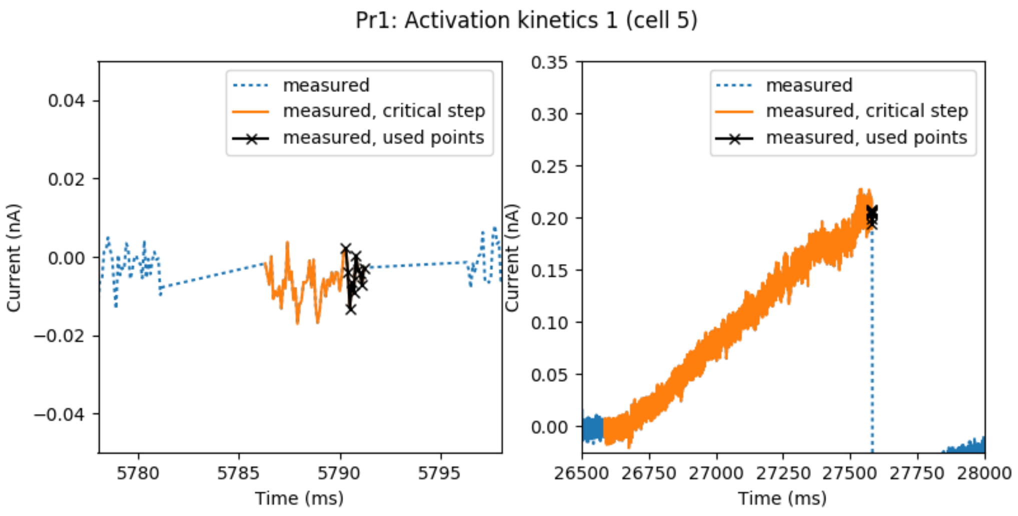

Figure 22: Current in cell 5 during Pr1, second and last step

Fig. 22 shows the second and last steps, with the parts used for the summary statistic highlighted. Instead of using the final point in the trace, I took 10 points (1 µs) and took the mean. As the right panel shows, this still doesn’t look like a great estimate of where a line fit through the entire highlighted current would go.

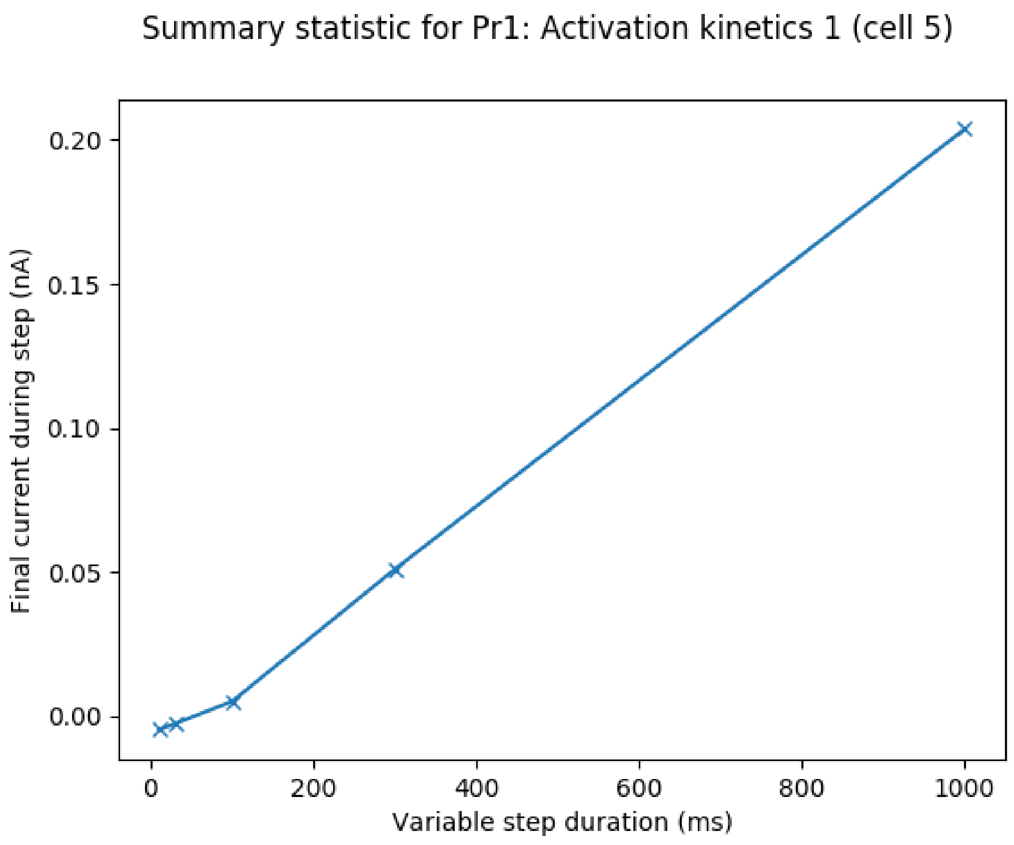

Figure 23: Summary statistic for Pr1 on Cell 5.

257 *5.1.4. Pr1 and Pr2: Real vs simulated*

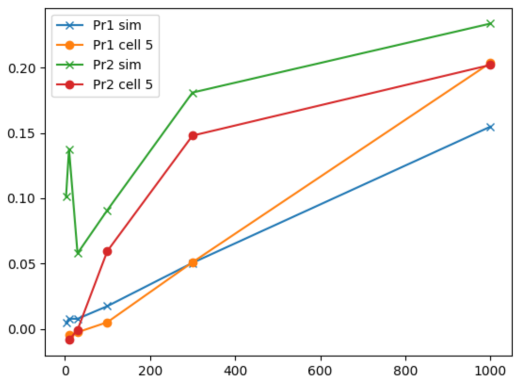

Figure 24: Pr1 and Pr2: Cell 5 data and simulation (with tidied protocol). The longer steps are ok, but the lower steps show an artefact in the simulation that's not observed in real cells.

5.2. *Pr3: Steady-state activation*

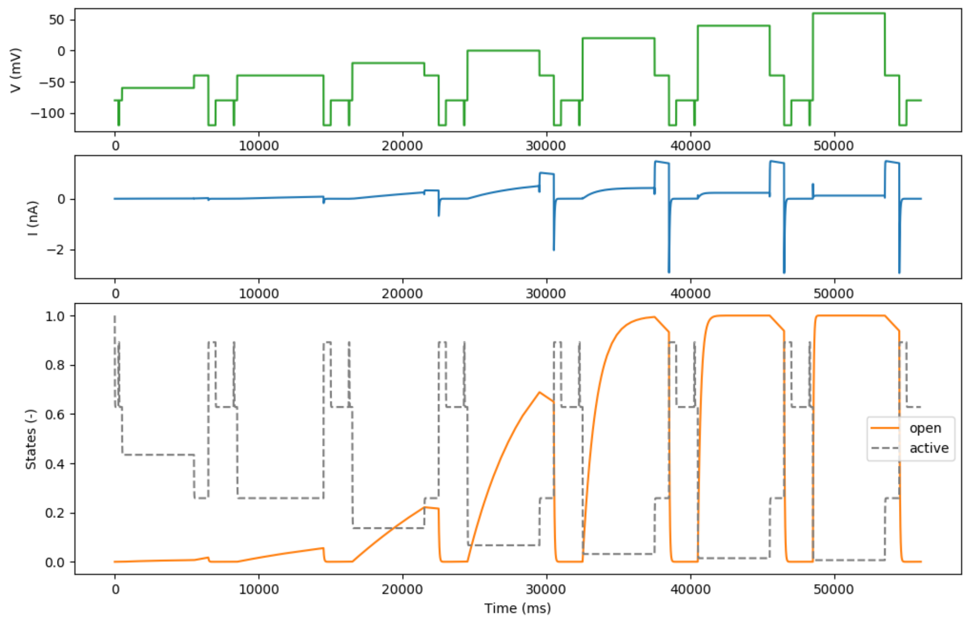

Figure 25: Pr3 (tidied)

This protocol does a good job of opening the channel, as Fig. 25 shows. This is good, as it creates some signal for us to observe. It steps to a number of different voltages too, so we can get some voltage dependence info.

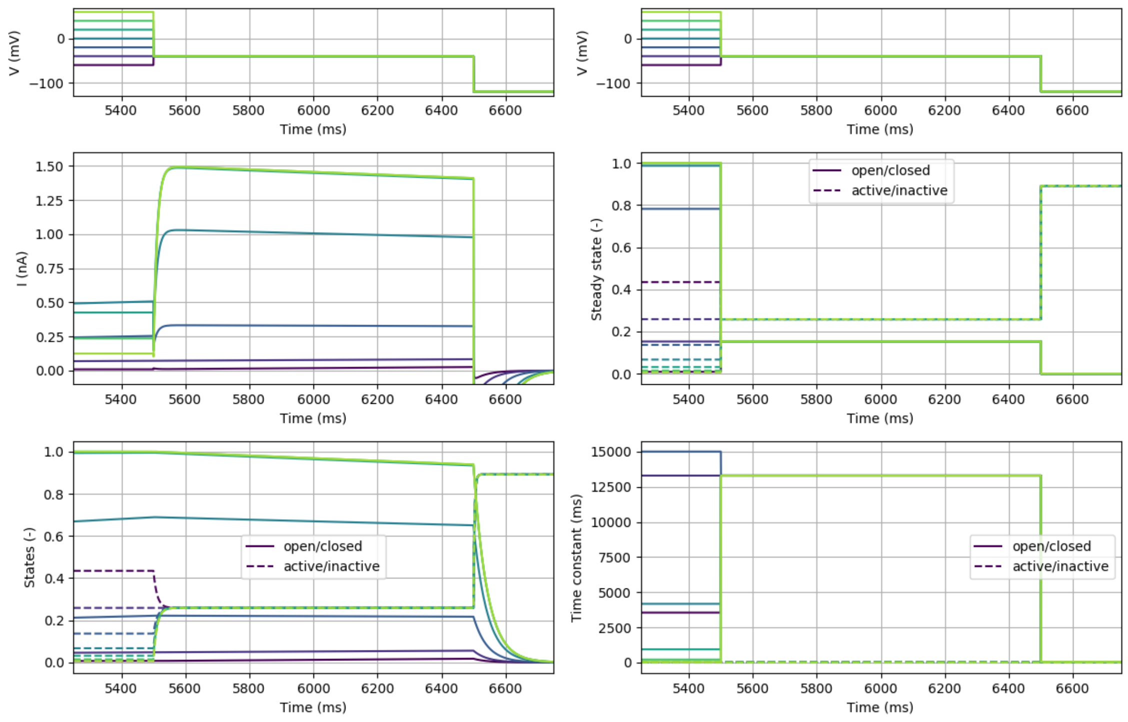

Figure 26: Pr3 (tidied, steps overlayed). Simulated with Kylie's model using the parameters obtained for the fit to the full traditional traces.

The critical bit of the protocol is the step to -40mV *after* the varying voltage step. Fig. 26 shows what happens during this step. The current shown here is quite different from what is observed in real cells, see e.g. Sanguinetti and Jurkiewicz (1990).

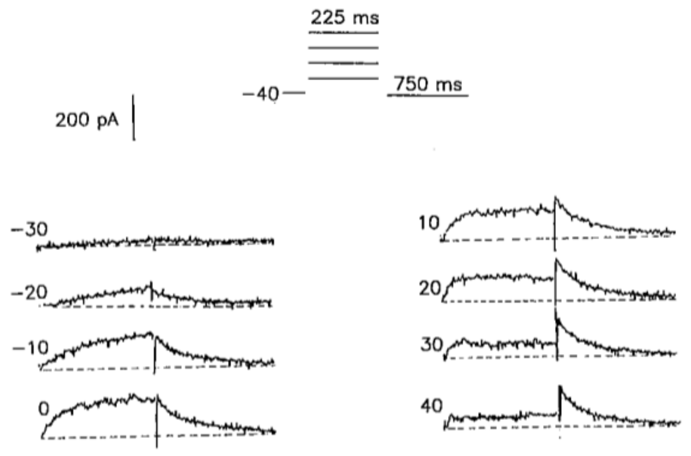

FIGURE 7. **E-4031-sensitive currents. The drug-sensitive currents were measured at the indicated test potentials, ranging from −30 to +40 mV. Data are from same cell as Fig. 6.**

Figure 27: The first recording of IKr (Sanguinetti and Jurkiewicz, 1990). Note the immediate rise after the voltage step to -40mV (the tail current), followed by exponential decay.

Fig. 27 shows the expected behaviour during this step: an exponentially decreasing tail current, whose peak (right at the start of the fixed-voltage step) can be measured.

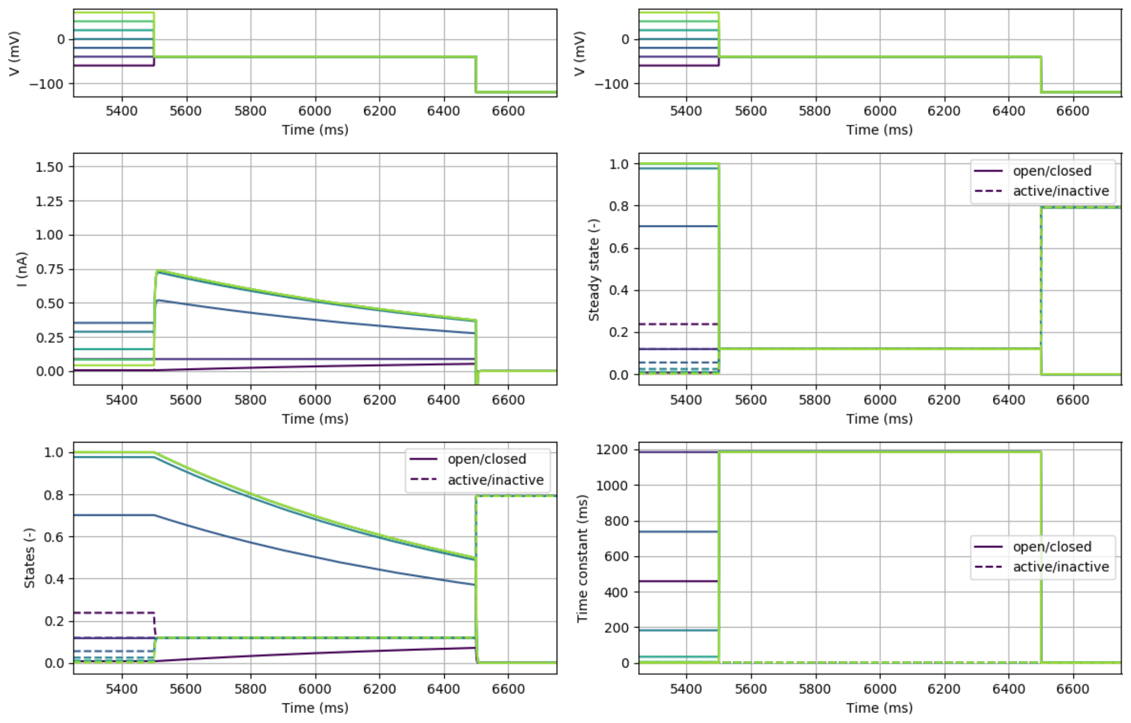

Figure 28: Pr3 (tidied, steps overlayed). Simulated with the I_{Kr} model from ten Tusscher and Panfilov (2006).

The model by ten Tusscher and Panfilov (2006) does show the (qualitatively) correct behaviour, as shown in Fig. 28. The middle-right panel shows that, just before the step to -40mV, the open/closed state variable has its steady state at values ranging from 0mV to 1mV. This is good, as this is the variable we'd like to measure. Upon the step to -40mV, inactivation immediatly kicks in (bottom left panel), because its time constant at this voltage is very small (bottom right panel). The time constant for open/closed is much larger, so that the peak current during the -40mV has a fixed activation/inactivation (since it's all at -40mV) but a variable open/closed (since it's still at the value it obtained during the variable voltage step). This means that, by normalising, we can get an estimate of the steady state of opening/closing.

Note however, that all this depends on the time-constants being just right! What happens in the Fig. 26? First, it seems that the time constant of inactivation at -40mV is too large, so that we can clearly see the rising current at the start of the -40mV, instead of the near instantaneous jump in Fig. 27 and Fig. 28. Second, the time constant of opening/closing is 10 times larger than in the TNNP model, leading to the very slow decay after the peak. So for the behaviour *is* there, just time-scaled beyond recognition.

Note that the TNNP model uses a tricky formulation that separates the steady-states and time constants (basing one on an α and β but specifying the other independently). Perhaps for similar reasons? It could be that a two-state HH model is not able to correctly reproduce this behaviour (at least not when parametrised to also capture other desired behaviours).

5.3. *Pr3: Summary statistic*

5.4. *Pr3 in cell 5*

Interestingly, cell 5 showed the same 'unexpected' behaviour as the simulations based on the full-trace cell 5 fits.

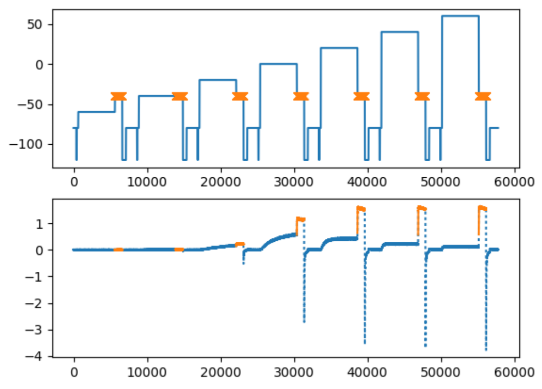

Figure 29: Pr3 in cell 5.

Zooming in, we can clearly see the slow rise at the start of the step, and the subsequent decay is so slow that the noise obscured the true peak location.

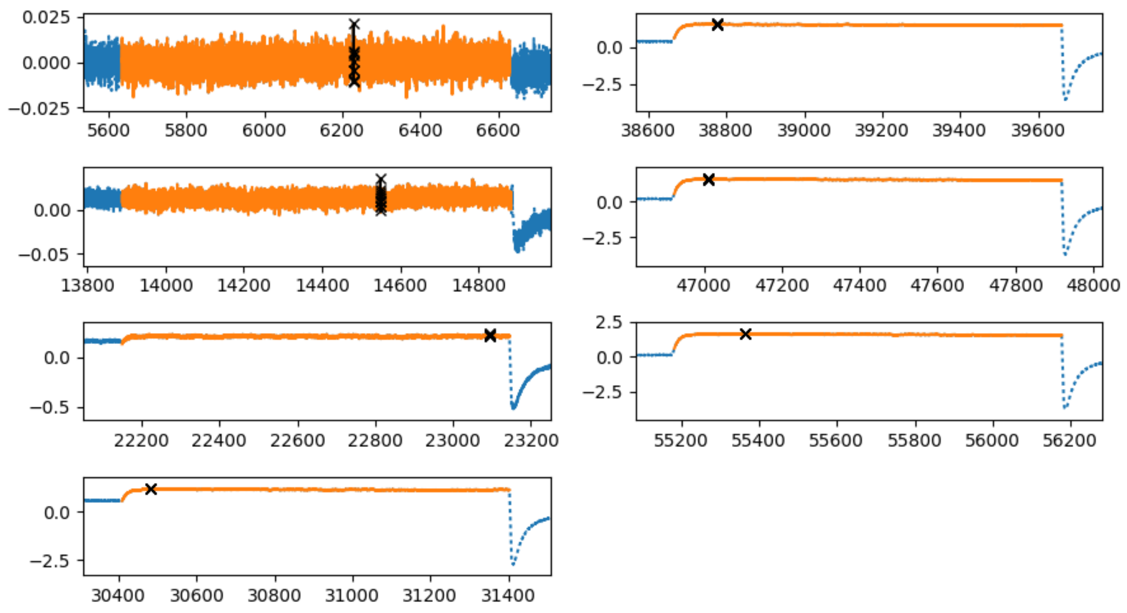

Figure 30: Pr3 in cell 5, zoomed in on the critical steps.

Is this perhaps due to the lower temperature (21-22 deg C in Kylie's data, versus 35 in Sanguinetti and Jurkiewicz (1990) and the Zhou et al. (1998) data that the model by ten Tusscher and Panfilov (2006) is based on.

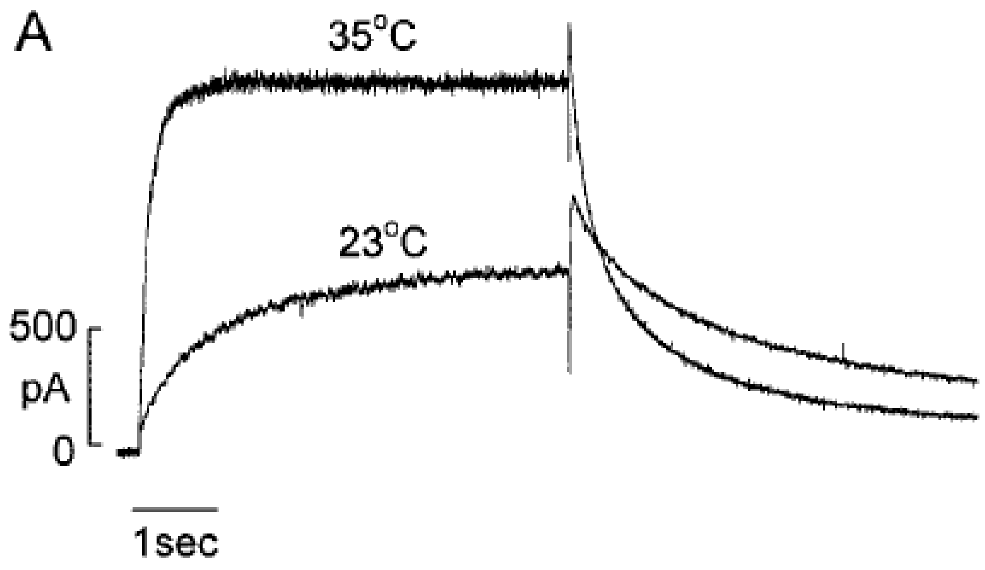

FIGURE 8 Temperature dependence of HERG current. (*A*) Current traces recorded from a stably transfected HEK 293 cell at 23°C and 30 s later at 35°C. The cell was held at −80 mV and clamped to 0 mV for 5 s, and tail current was recorded with a repolarizing step to −50 mV.

Figure 31: IKr (HERG) tail current in a HEK cell at 23 and 35 deg C, from (Zhou et al., 1998).

Fig. 31 shows that this could be partially the case.

IDEA

Ultimately, this is all another argument in favour of not using a HH style analysis that depends on assumptions about the time constants of the system you're about to measure! Maybe that's the best argument actually, regardless of quality of fit, signal-to-noise ratios, analysis differences, etc. The analysis & protocol *can* depend on pre-existing knowledge of the system, in a way they must, and optimal design certainly assumes so. *But* they should be robust against large variations in the parameters we aim to discover. *Especially* in the light of biological variability, not to mention drug block, mutations and other instances where we actually want to *show* that the parameters have changed. We should maybe think of some way to measure how well our protocols work over wide ranges of possible parameters

6. Comparing different types of fitting

6.1. AP validation

Comparing methods by looking at model prediction for AP signal.

Loaded protocol from matlab, cell 5 data from matlab.

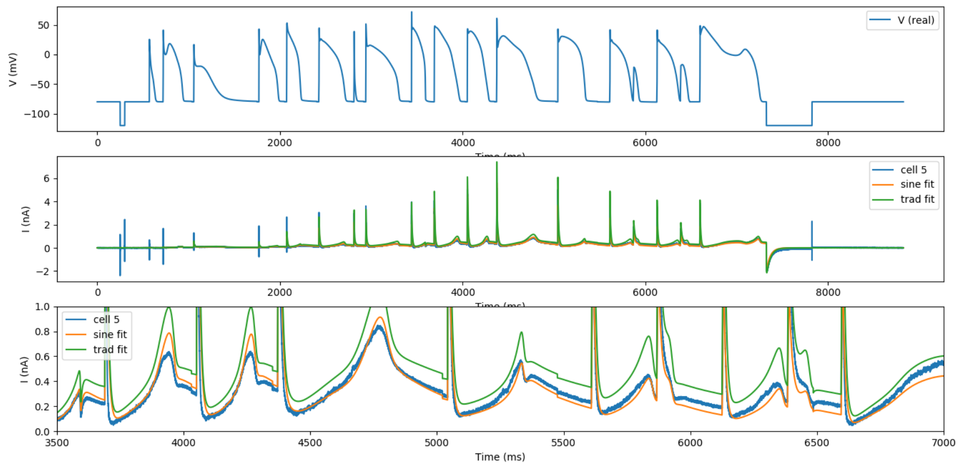

Figure 32: AP protocol: Data from cell 5 plus simulation based on sine-wave fit parameters and trad. prot. fit parameters.

Table 8: AP protocol validation log-likelihoods.

Fit	**Log-posterior**
Sine wave	-21273679.167
Trad. prot.	-74641604.9933

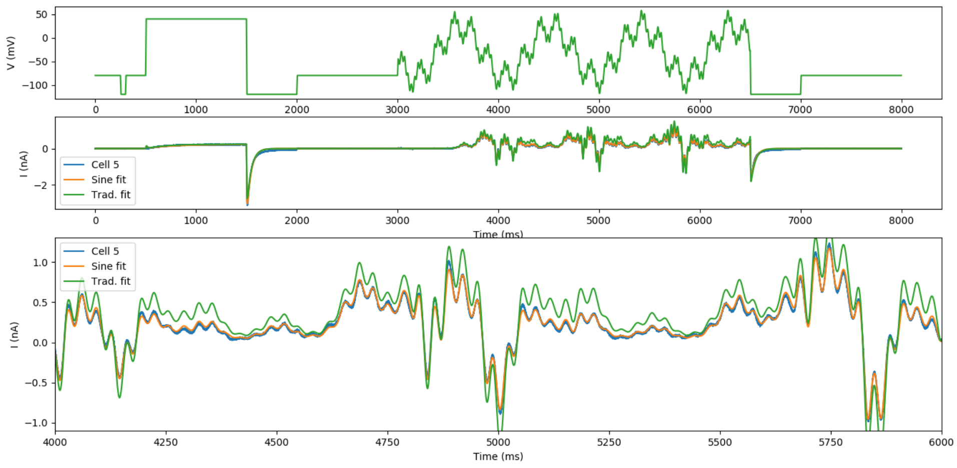

Figure 33: Sine-wave protocol: Data from cell 5 plus simulation based on sine-wave fit parameters and trad. prot. fit parameters.

Table 9: Sine wave validation log-likelihoods.

Fit	**Log-posterior**
Sine wave	-1509487.67775
Trad. prot.	-18791120.0047

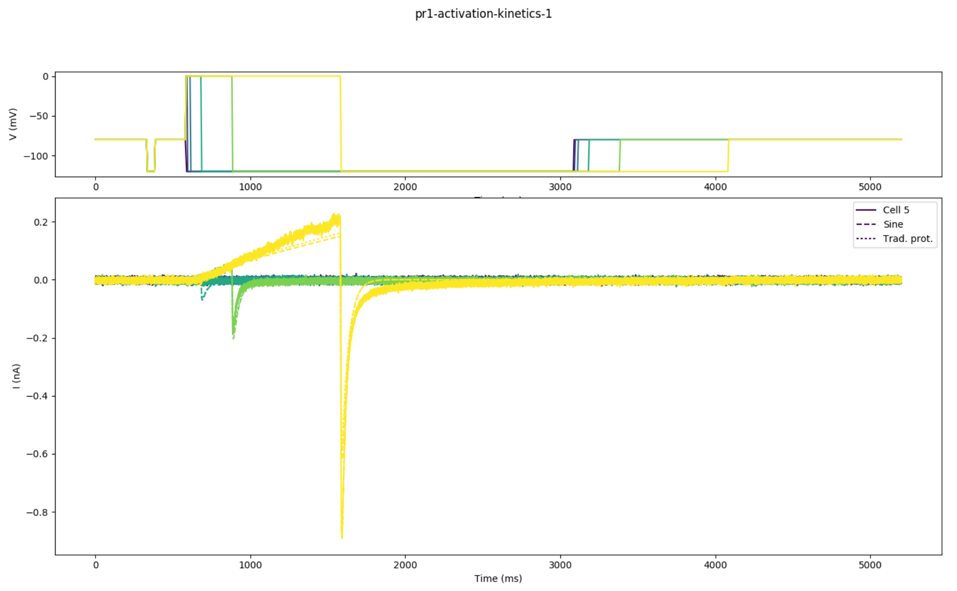

Figure 34: Pr1

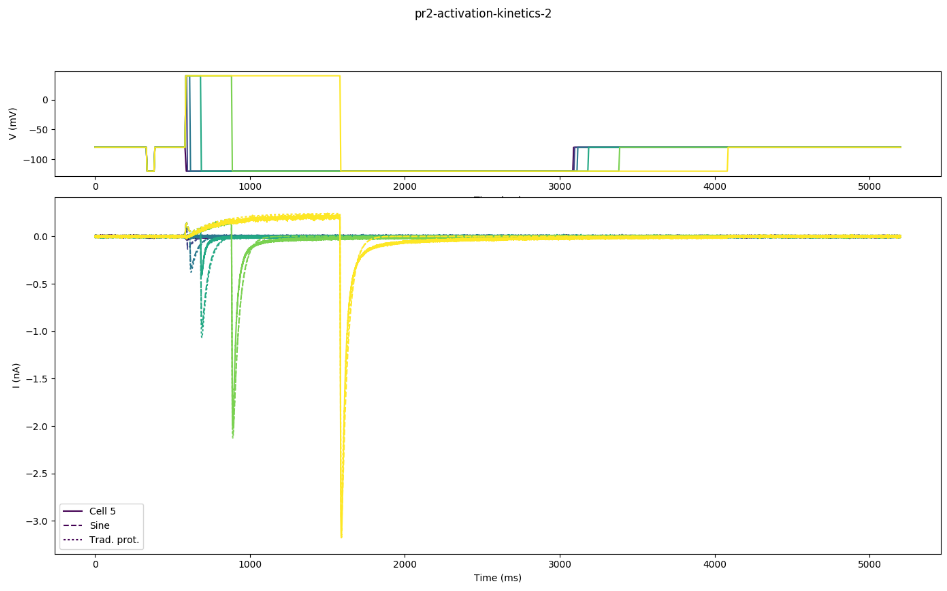

Figure 35: Pr2

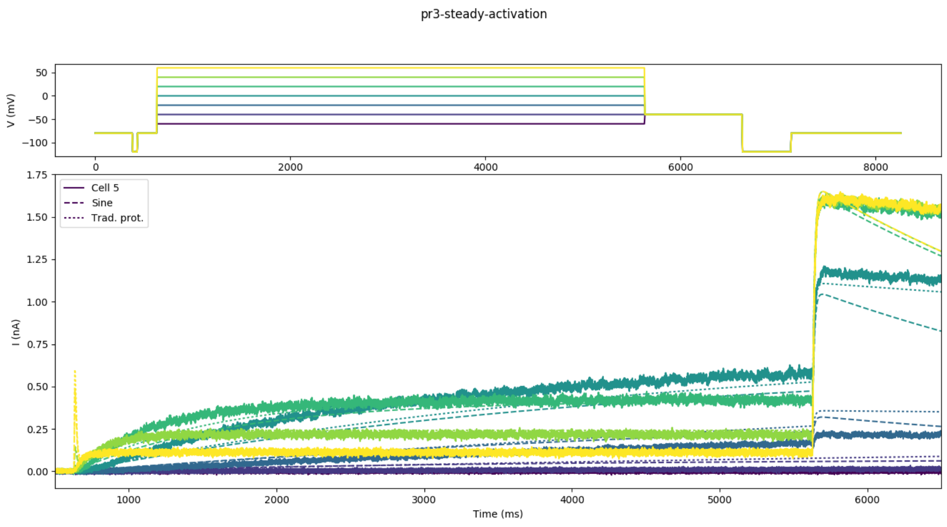

Figure 36: Pr3

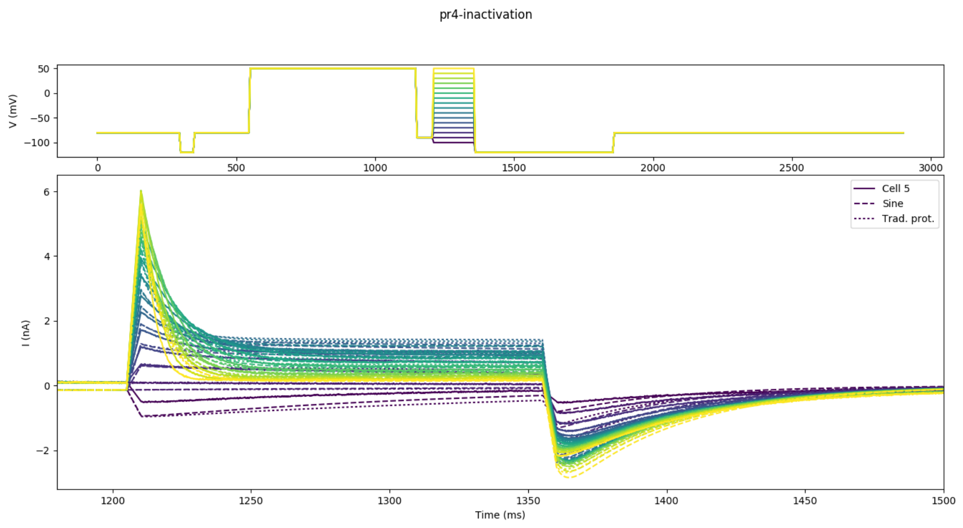

Figure 37: Pr4

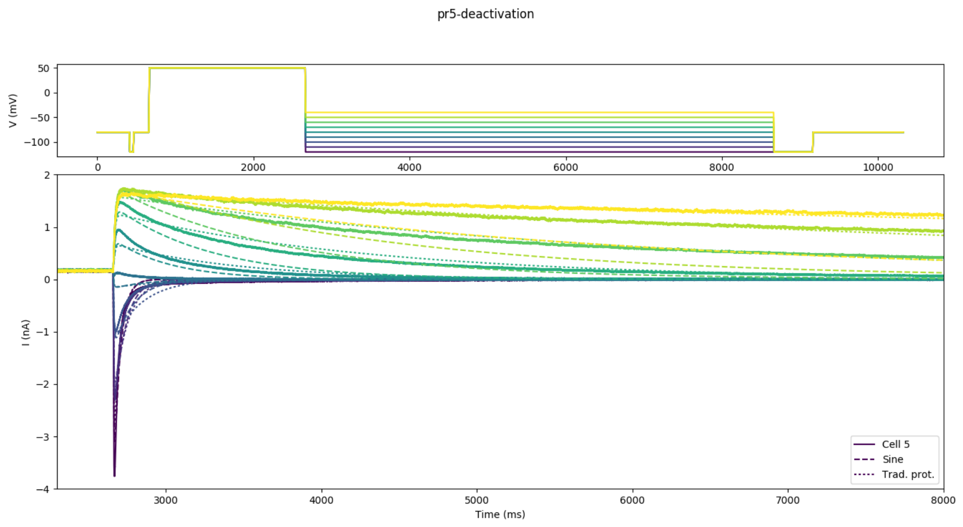

Figure 38: Pr5

309 *6.3. Trad. prot. cross-validation*

Table 10: Trad. prot. validation log-likelihoods.

Prot.	**Fit**	**Log-posterior**
Pr1 (ak-1)	**sine**	473746
Pr1 (ak-1)	trad	441919
Pr2 (ak-2)	**sine**	-6329912
Pr2 (ak-3)	trad	-7443985
Pr3 (s.act)	sine	-80888280
Pr3 (s.act)	**trad**	-39387497
Pr4 (inact)	**sine**	-14161029
Pr4 (inact)	trad	-19444287
Pr5 (deact)	sine	-1188041457
Pr5 (deact)	**trad**	-41635915
Combined	sine	-1288946932
Combined	**trad**	-107469765

Appendix A. Problem statement

We have some noisy time-series data and a forward model (simulation) that can be used to replicate it. We'd like to find out which parameter values are compatible with the experimental evidence.

Appendix A.1. Single series

1. Observations $D = \{(t_1, z_1), ..., (t_n, z_n)\}$ where t is time ($t_i > t_{i-1}$ for $i > 1$) and $z_i \in \mathbb{R}$ is a noisy measurement at time t_i.
2. Forward model $\mu(t|\theta) \to \mathbb{R}$ with m parameters, bundled in θ.
3. The parameters live in some bounded space $\theta \in \Theta \subset \mathbb{R}^m$
4. *Only* normally distributed, time-independent noise with some unknown variance σ^2, such that $z_i = \mu(t_i|\theta_{true}) + \mathcal{N}(0, \sigma^2)$.

Numbers 1,2,3 are definitions. Number 4 is a dodgy assumption that we'll revisit later.

For the single series problem, we can now define the *likelihood* of a parameter set θ as the probability *density* of obtaining observations D for parameters θ:

$$l(\theta|D) \equiv f(D|\theta) \tag{A.1}$$

With our dodgy assumption of purely random noise, the measurement error at any point in time is independent of the signal history, so that:

$$f(D|\theta) = \prod_{i=1}^{n} f(t_i, z_i|\theta) \tag{A.2}$$

With normally distributed noise, the probability density function for observing D is then:

$$f(x|\theta, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu(t_i|\theta))^2}{2\sigma^2}\right) \tag{A.3}$$

And taking the logarithm, we find

$$\log l(\theta, \sigma|D) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(z_i - \mu(t_i|\theta))^2 \tag{A.4}$$

Appendix A.1.1. The value of sigma

There are two strategies for finding σ:

1. Find a flat bit of signal, and use the sample standard deviation.
2. Add σ to the parameter vector θ.

Appendix A.1.2. Influence of sampling rate

Assuming the noise is truly stochastic, we can double the sampling rate (so that a new point is inserted between any two subsequent points) and get more random noise.

$$\log l(\theta, \sigma|D) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(z_i - \mu(t_i|\theta))^2 \tag{A.5}$$

This will double the first two terms, and - if the original number of samples was large - approximately double the last term as well.

This seems wrong. Ideally, we'd have some way to calculate the likelihood that would give a very wide distribution on θ for a low n, and then get more narrow as n decreases, *until it stabilises at the point where the signal contains enough information.* On the other hand, the more specific we are about exactly what random noise signal we measure, the less likely this exact result should become, so from the point of view of quantifying the likelihood of some random noise it makes sense...

Appendix A.1.3. Relation to optimisation problems

Given the loglikelihood equation

$$\log l(\theta, \sigma|D) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(z_i - \mu(t_i|\theta))^2 \tag{A.6}$$

Since n and σ are properties of the signal, indepdent of the model, we can treat the optimisation problem as minimising

$$\sum_{i=1}^{n}(z_i - \mu(t_i|\theta))^2 \tag{A.7}$$

Similarly, we could start from the root-mean squared error (RMSE), which is basically the sample standard deviation (with n instead of $n-1$:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(z_i - \mu(t_i|\theta))^2}{n}} \tag{A.8}$$

and this would give the same result.

Appendix A.2. Combining multiple traces

Following the dodgy assumption

$$f(D|\theta) = \prod_{i=1}^{n} f(t_i, z_i|\theta) \tag{A.9}$$

combining traces can be achieved by simply adding more points to D. (Another way to think about it would be to split a single series in 2, and work out what should happen knowing that the result should be unchanged).

More generally, we can look at the general MCMC problem

$$f(\theta|D) = \frac{f(D|\theta)f(\theta)}{f(D)} \tag{A.10}$$

and add a second data set:

$$f(\theta|D, E) = \frac{f(D, E|\theta)f(\theta)}{f(D, E)} \tag{A.11}$$

if D and E are independent, this becomes

$$f(\theta|D, E) = \frac{f(D|\theta)f(E|\theta)f(\theta)}{f(D, E)} \tag{A.12}$$

so we can replace the likelihood in our original problem by the product of two independent likelihoods!

For dependent D and E we'd have to solve

$$f(\theta|D,E) = \frac{f(D|E,\theta)f(E|\theta)f(\theta)}{f(D,E)} \tag{A.13}$$

since we can't universally define $f(D|E,\theta)$ there's no point trying to do this in a general way in Pints (though users can always hack it in).

But what about weighting? Is there any reason we should? For example, if we care about n, it may make sense to weigh by $1/n_i$ for each experiment i?

Making the link to miminising errors, it looks like we can just add up errors then, unless we care about n.

Appendix B. Distributions

Following Papoulis and Pillai (2002).

Let $\boldsymbol{x}$ be a random variable

$$F(x) \equiv P\{\boldsymbol{x} \leq x\} \tag{B.1}$$

then

$$\begin{align} F(-\infty) &= 0 \tag{B.2} \\ F(+\infty) &= 1 \tag{B.3} \end{align}$$

$$\begin{align} x_1 < x_2 &\rightarrow F(x_1) \leq F(x_2) \tag{B.4} \\ F(x_0) = 0, x < x_0 &\rightarrow F(x) = 0 \tag{B.5} \end{align}$$

$$\begin{align} \{\boldsymbol{x} \leq x_2\} &= \{\boldsymbol{x} \leq x_1\} \cup \{x_1 < \boldsymbol{x} \leq x_2\} \tag{B.6} \\ P\{\boldsymbol{x} \leq x_2\} &= P\{\boldsymbol{x} \leq x_1\} + P\{x_1 < \boldsymbol{x} \leq x_2\} \tag{B.7} \\ P\{x_1 < \boldsymbol{x} \leq x_2\} &= P\{\boldsymbol{x} \leq x_2\} - P\{\boldsymbol{x} \leq x_1\} \tag{B.8} \\ &= F(x_2) - F(x_1) \tag{B.9} \end{align}$$

if also

$$\begin{align} F(x^+) &\equiv \lim_{\epsilon \downarrow 0} F(x+\epsilon) \tag{B.10} \\ F(x^-) &\equiv \lim_{\epsilon \uparrow 0} F(x+\epsilon) \tag{B.11} \end{align}$$

then

$$F(x_2) - F(x_1^-) = P\{x_1 \leq \boldsymbol{x} \leq x_2\} \tag{B.12}$$

Appendix B.1. Probability density functions

Definition:

$$f(x) \equiv \frac{d}{dx}F(x) \tag{B.13}$$

and

$$\int_{-\infty}^{+\infty} f(x)dx = 1 \tag{B.14}$$

$$F(x) = \int_{-\infty}^{x} f(x)dx \tag{B.15}$$

$$P\{x_1 < \boldsymbol{x} \leq x_2\} = F(x_2) - F(x_1) = \int_{x_1}^{x_2} f(x)dx \tag{B.16}$$

See Papoulis and Pillai (2002) page 81.

Appendix B.1.1. Independent PDFs

Vector of random variables

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\} \tag{B.17}$$

$$F(\boldsymbol{X}) = F(\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}) = P\{\boldsymbol{x}_1 \leq x_1, \boldsymbol{x}_2 \leq x_2, ..., \boldsymbol{x}_n \leq x_n\} \tag{B.18}$$

PDF

$$f(X) \equiv \frac{\partial^n F(x_1, x_2, ..., x_n)}{\partial x_1 \partial x_2 \cdots \partial x_n} \tag{B.19}$$

$$P\{\boldsymbol{X} \in D\} = \int_D f(X)dX, \qquad X = [x_1, x_2, ..., x_n] \tag{B.20}$$

where D is a region (some volume).

For *independent variables*:

$$F(x_1, x_2, ..., x_n) = F(x_1)F(x_2)\cdots F(x_3) \tag{B.21}$$

$$\frac{\partial}{\partial x_j} F_{i \neq j} = 0 \tag{B.22}$$

$$f(X) = f(x_1)f(x_2)\cdots f(x_n) \tag{B.23}$$

Appendix B.2. PDFs conditional on an event

$$F(x_1|A) = P\{\boldsymbol{x} \le x_1|A\} = \frac{P\{\boldsymbol{x} \le x_1, A\}}{P(A)} \tag{B.24}$$

$$\begin{aligned} &F(-\infty|A) = 0 && \text{(B.25)} \\ &F(+\infty|A) = 1 && \text{(B.26)} \\ &P\{x_1 < \boldsymbol{x} \le x_2|A\} = F(x_2|A) - F(x_1|A) && \text{(B.27)} \\ & && \text{(B.28)} \end{aligned}$$

$$f(x|A) \equiv \frac{d}{dx}F(x|A) \tag{B.29}$$

See Papoulis and Pillai (2002) page 98.

Bayes rule

$$f(x|A) = \frac{P(A|\boldsymbol{x} = x)f(x)}{P(A)} \tag{B.30}$$

See Papoulis and Pillai (2002) page 103.

Appendix B.3. PDFs conditional on a random variable

$$F(x|M) = P\{\boldsymbol{x} \le x|M\} = \frac{P\{\boldsymbol{x} \le x, M\}}{P(M)} \tag{B.31}$$

$$F(x, y|M) = P\{\boldsymbol{x} \le x, \boldsymbol{y} \le y|M\} = \frac{P\{\boldsymbol{x} \le x, \boldsymbol{y} \le y, M\}}{P(M)} \tag{B.32}$$

We can use this to find the definition of

$$F(y|x) \equiv F\{y|\boldsymbol{x} = x\} \tag{B.33}$$

by setting $M = \{x_1 < x \le x_2\}$ and using some sneaky limits

$$F_y(y|x_1 < \boldsymbol{x} \le x_2) = \frac{P\{x_1 < \boldsymbol{x} \le x_2, \boldsymbol{y} \le y\}}{P\{x_1 < \boldsymbol{x} \le x_2\}} = \frac{F(x_2, y) - F(x_1, y)}{F_x(x_2) - F_x(x_1)} \tag{B.34}$$

Take $\frac{\partial}{\partial y}$ to find

$$f_y(y|x_1 < \boldsymbol{x} \le x_2) = \frac{\int_{x_1}^{x_2} f(x,y)dx}{F_x(x_2) - F_x(x_1)} \tag{B.35}$$

Now set $x_1 = x$ and $x_2 = x + \Delta x$:

$$f_y(y|x < \boldsymbol{x} \le x + \Delta x) = \frac{\int_{x}^{x+\Delta x} f(\alpha,y)d\alpha}{F_x(x+\Delta x) - F_x(x)} \tag{B.36}$$

where $\int_{x}^{x+\Delta x} f(\alpha,y)d\alpha \approx f(x,y)\Delta x$ and $\frac{F_x(x+\Delta x)-F_x(x)}{\Delta x} \approx f_x(x)$
so that

$$f_y(y|x < \boldsymbol{x} \le x + \Delta x) = \frac{\int_{x}^{x+\Delta x} f(\alpha,y)d\alpha}{F_x(x+\Delta x) - F_x(x)} \approx \frac{f(x,y)\Delta x}{f_x(x)\Delta x} \tag{B.37}$$

finally, take the limit to zero to find

$$f_y(y|\boldsymbol{x} = x) = \lim_{\Delta x \to 0} f_y(y|x < \boldsymbol{x} \le x + \Delta x) = \frac{f(x,y)}{f_x(x)} \tag{B.38}$$

If we drop the f_x and f_y notation (assuming that the ambiguous f is clear from context), this becomes:

$$f(y|x) = \frac{f(x,y)}{f(x)} \qquad f(x|y) = \frac{f(x,y)}{f(y)} \tag{B.39}$$

See Papoulis and Pillai (2002) page 221.

Appendix B.4. Bayes again

From equation B.39 we get:

$$f(x,y) = f(y|x)f(x) = f(x|y)f(y) \tag{B.40}$$

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)} \tag{B.41}$$

See Papoulis and Pillai (2002) page 224.

Appendix B.5. Independent conditions

$$f(x|y,z) = \frac{P(y,z|x)f(x)}{f(y,z)} \tag{B.42}$$

Assuming independence:

$$f(x|y,z) = \frac{f(y,z|x)f(x)}{f(y,z)} \tag{B.43}$$

$$= \frac{f(y|x)f(z|x)f(x)}{f(y)f(z)} \tag{B.44}$$

$$= \frac{f(y|x)f(z|x)f(x)}{f(y)f(z)}\frac{f(x)}{f(x)} \tag{B.45}$$

$$= \frac{f(y|x)f(x)}{f(y)}\frac{f(z|x)f(x)}{f(z)}\frac{1}{f(x)} \tag{B.46}$$

$$= \frac{f(x|y)f(x|z)}{f(x)} \tag{B.47}$$

References

Beattie, K.A., Hill, A.P., Bardenet, R., Cui, Y., Vandenberg, J.I., Gavaghan, D.J., de Boer, T.P., Mirams, G.R., 2017. Sinusoidal voltage protocols for rapid characterization of ion channel kinetics. bioRxiv 100677.

Clerx, M., Collins, P., de Lange, E., Volders, P.G.A., 2016. Myokit: A simple interface to cardiac cellular electrophysiology. Progress in Biophysics and Molecular Biology 120, 100–114.

Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S., 2005. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software 31, 363–396.

Papoulis, A., Pillai, S.U., 2002. Probability, random variables, and stochastic processes. McGraw-Hill Higher Education. 4th edition.

Sanguinetti, M.C., Jurkiewicz, N.K., 1990. Two components of cardiac delayed rectifier k+ current. differential sensitivity to block by class iii antiarrhythmic agents. The Journal of general physiology 96, 195–215.

ten Tusscher, K.H., Panfilov, A.V., 2006. Alternans and spiral breakup in a human ventricular tissue model. American Journal of Physiology. Heart and Circulatory Physiology 291, H1088–H1100.

Zhou, Z., Gong, Q., Ye, B., Fan, Z., Makielski, J.C., Robertson, G.A., January, C.T., 1998. Properties of herg channels stably expressed in hek 293 cells studied at physiological temperature. Biophysical journal 74, 230–241.