



Cside 2018

Statistical Inference for the Cardiac Excitation Model using Pints

Chon Lok Lei¹, Gary R Mirams², Michael Clerx¹

November 26, 2018

1. University of Oxford, Department of Computer Science
2. University of Nottingham, School of Mathematical Sciences

The team



Chon Lok Lei
DPhil student at Oxford
Dept. of Computer Science




Gary Mirams



Michael Clerx

<https://github.com/pints-team/pints>

 [pints-team / pints](#)

 Unwatch ▾ 10

 Unstar 21

 Fork 2

 Code

 Issues 140

 Pull requests 4

 Projects 0

 Insights

Probabilistic Inference on Noisy Time Series <http://pints.readthedocs.io>

[bayesian-methods](#)

[inverse-problems](#)

[parameter-estimation](#)

[numerical-optimization](#)



Martin Robinson



Ben Lambert



Sanmitra Ghosh



David Gavaghan

Introduction

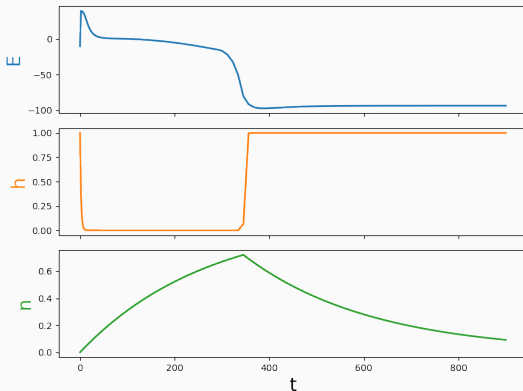
Model 1: cardiac excitation model

Model 1 describes the electrical excitation of cardiac cells. See [Simitev and Biktashev \(2010\)](#).

- 3 states E , h , n
- Stiff, non-linear, coupled ODEs
- 12 parameters to be inferred

Model outputs

Model simulated with default parameters



Defining likelihood

Since we know

$$y = f(t; \Theta) + \epsilon$$

where y is data, f model, Θ parameters, and ϵ noise.

Defining likelihood

Since we know

$$\mathbf{y} = f(t; \boldsymbol{\Theta}) + \boldsymbol{\epsilon}$$

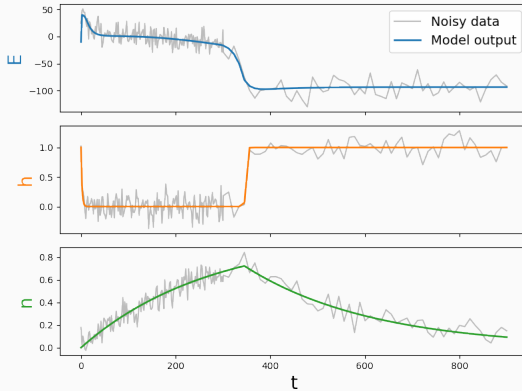
where \mathbf{y} is data, f model, $\boldsymbol{\Theta}$ parameters, and $\boldsymbol{\epsilon}$ noise.

We also know

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$$

with some constant noise level σ_i for each output.

What does noisy data look like:



S

Then we have likelihood

$$p(\mathbf{y}|\mathbf{\Theta}, \boldsymbol{\sigma}) = \prod_{i \text{ output}} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp \left(- \sum_t \frac{(f_i(t; \mathbf{\Theta}) - y_i)^2}{2\sigma_i^2} \right)$$

Overview of what we did

1. Solving the model equations
 - Reimplement the model in Python (using CVODE)

Overview of what we did

1. Solving the model equations
 - Reimplement the model in Python (using CVODE)
2. Optimisation to find optimum (via Pints)

Overview of what we did

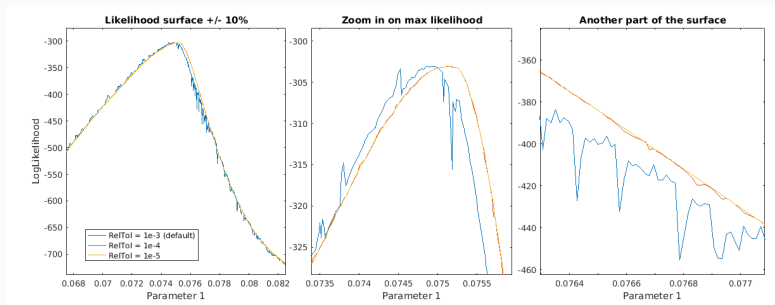
1. Solving the model equations
→ Reimplement the model in Python (using CVODE)
2. Optimisation to find optimum (via Pints)
3. MCMC to explore likelihood around optimum (via Pints)

Overview of what we did

1. Solving the model equations
→ Reimplement the model in Python (using CVODE)
2. Optimisation to find optimum (via Pints)
3. MCMC to explore likelihood around optimum (via Pints)

Effect of ODE solver tolerances

Using solver inappropriately might lead to incorrect result



1. The peak shifted!
2. Low tolerance gives bumpy likelihood surface!

Synthetic data study

Study the model using default parameters

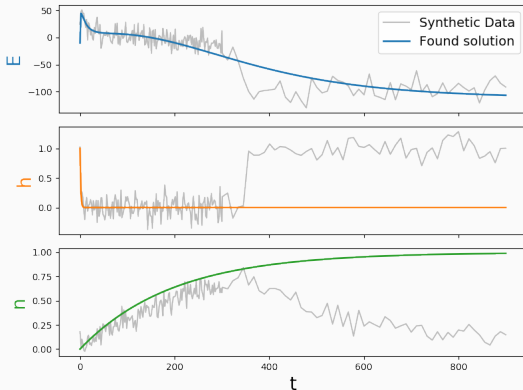
- Help us to understand the model better
- Know the limitations of optimiser/MCMC
- See how well we can fit with the model (and data)

Methods

Global optimisation: CMA-ES (Hansen, 2006)

Directly run an optimisation

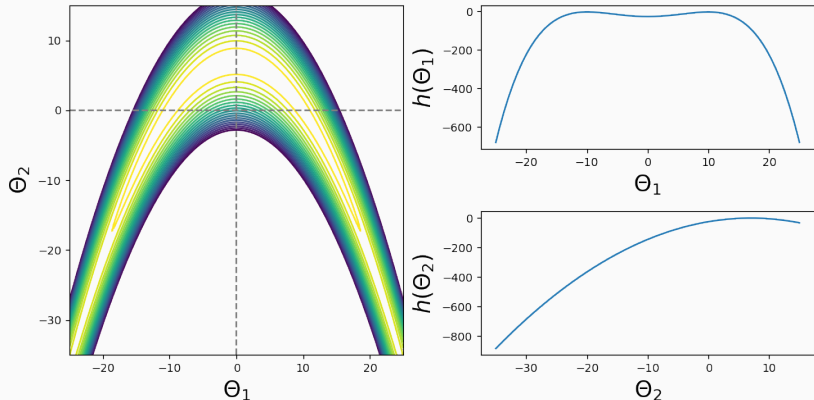
If this had worked, problem solved...



We set boundaries to be $[0.1, 10] \times \Theta_{\text{default}}$. It gets stuck at some weird local minimum in the parameter space...

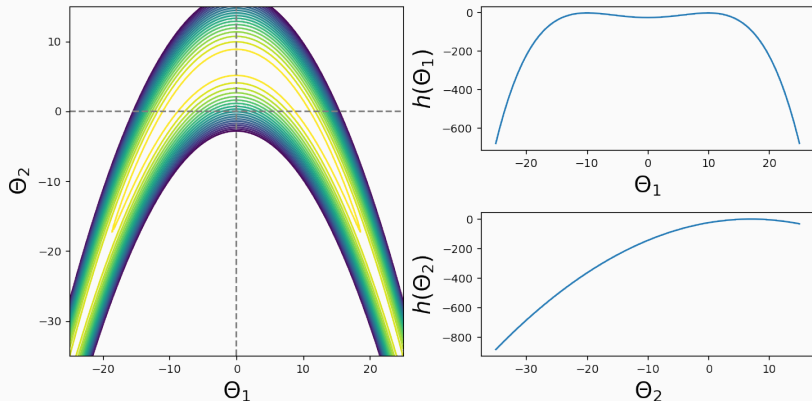
Why is global optimisation not working?

An example of 1D log-likelihood slice



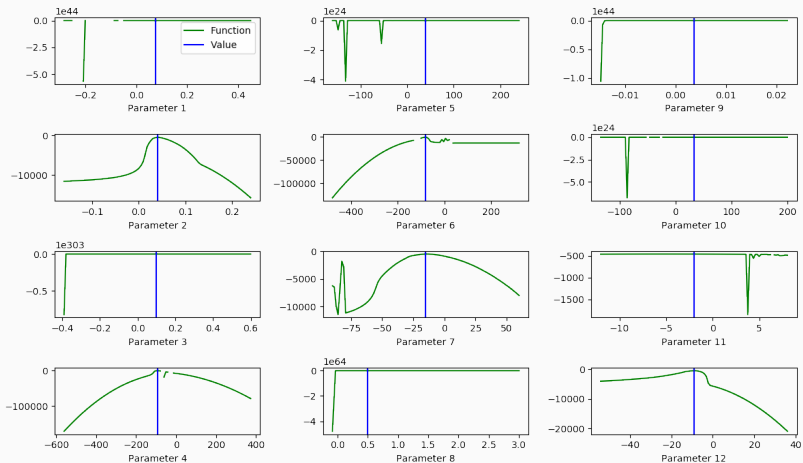
Why is global optimisation not working?

An example of 1D log-likelihood slice



Ideally, we hope $h(\Theta_i)$ behaves convex-like...

1D log-likelihood slices from synthetic data



They look really bad here!

Maybe we can argue from the model equations...

$$\frac{dE}{dt} = G_{Na}(E_{Na} - E)\mathcal{H}(E - E_*)h + \tilde{g}_2(E)n^4 + \tilde{G}(E)$$

$$\frac{dh}{dt} = F_h (\mathcal{H}(E_{\dagger} - E) - h)$$

$$\frac{dn}{dt} = f_n (\mathcal{H}(E - E_{\dagger}) - n)$$

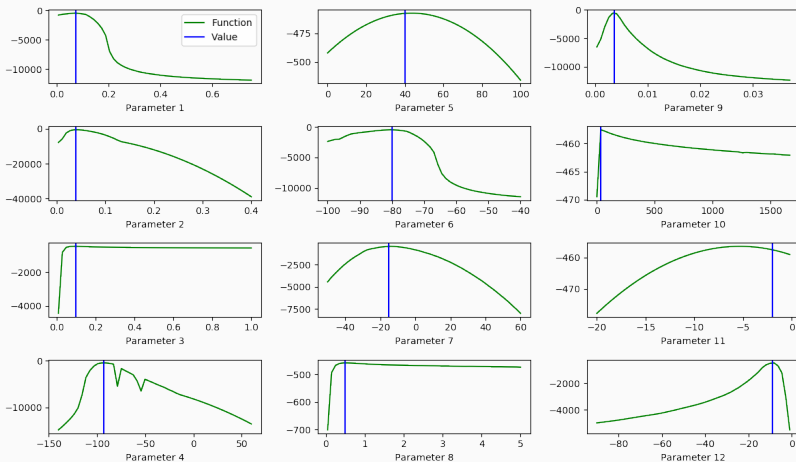
where

$$\tilde{g}_2(E) = g_{21}\mathcal{H}(E_{\dagger} - E) + g_{22}\mathcal{H}(E - E_{\dagger})$$

$$\tilde{G}(E) = \begin{cases} k_1(E_1 - E), & E \in (-\infty, E_{\dagger}) \\ k_2(E - E_2), & E \in [E_{\dagger}, E_*) \\ k_3(E_3 - E), & E \in [E_*, +\infty) \end{cases}$$

$\mathcal{H}(\cdot)$ is the Heaviside unit step function

Better boundaries gives a better 1D slices



Some parameters still look quite bad here...

Maybe we can argue from the model equations...

$$\frac{dE}{dt} = G_{Na}(E_{Na} - E)\mathcal{H}(E - E_*)h + \tilde{g}_2(E)n^4 + \tilde{G}(E)$$

$$\frac{dh}{dt} = F_h (\mathcal{H}(E_{\dagger} - E) - h)$$

$$\frac{dn}{dt} = f_n (\mathcal{H}(E - E_{\dagger}) - n)$$

where

$$\tilde{g}_2(E) = g_{21}\mathcal{H}(E_{\dagger} - E) + g_{22}\mathcal{H}(E - E_{\dagger})$$

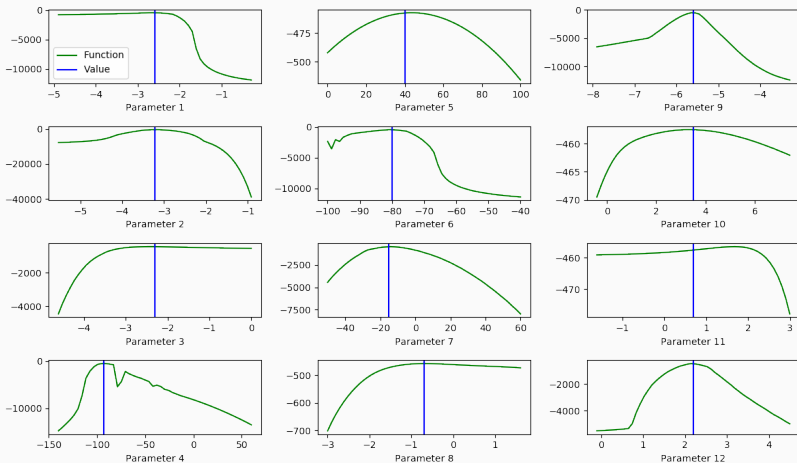
$$\tilde{G}(E) = \begin{cases} k_1(E_1 - E), & E \in (-\infty, E_{\dagger}) \\ k_2(E - E_2), & E \in [E_{\dagger}, E_*) \\ k_3(E_3 - E), & E \in [E_*, +\infty) \end{cases}$$

$\mathcal{H}(\cdot)$ is the Heaviside unit step function

Our choice of boundaries and parameter transformation

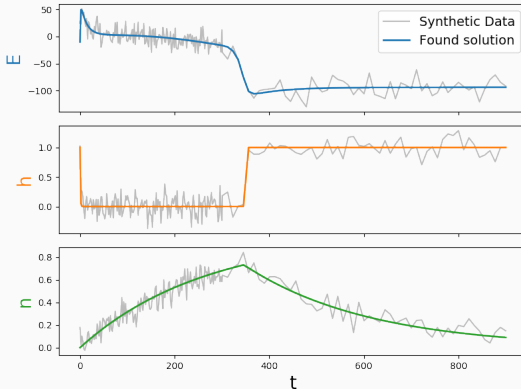
- For conductance/rate constants
 - Use $[0.1, 10] \times \Theta_{default}$ boundaries
 - Use transformation $g(\Theta_i) = \ln(\Theta_i)$
- For reversal potentials
 - Use $[E_{min}, E_{max}]$ boundaries
 - No transformation is used

With our chosen boundaries and parameter transformation



Now the 1D log-likelihood slices are much better!

(Re-)Try global optimisation method



And it is able to find a much closer solution than it was before!

CMA-ES found solution compared to default (true) value

Parameter index	Parameter name	CMA-ES found solution	Default (true) value
1	k1	0.1367	0.75
2	k2	0.0575	0.04
3	k3	0.0799	0.1
4	E1	-93.7497	-93.3333
5	Ena	49.8785	40
6	Edag	-79.0135	-80
7	Estar	-16.7657	-15
8	Fh	1.4804	0.5
9	fn	0.0038	0.0037
10	Gna	313.1379	33.3333
11	g21	-7.9916	-2
12	g22	-7.9098	-9

It is better

CMA-ES found solution compared to default (true) value

Parameter index	Parameter name	CMA-ES found solution	Default (true) value
1	k1	0.1367	0.75
2	k2	0.0575	0.04
3	k3	0.0799	0.1
4	E1	-93.7497	-93.3333
5	Ena	49.8785	40
6	Edag	-79.0135	-80
7	Estar	-16.7657	-15
8	Fh	1.4804	0.5
9	fn	0.0038	0.0037
10	Gna	313.1379	33.3333
11	g21	-7.9916	-2
12	g22	-7.9098	-9

It is better, but some of them are still quite bad...

How can we do it better?

We used adaptive (covariance) MCMC to look for a maximum a posteriori probability (MAP) estimate.

How can we do it better?

We used adaptive (covariance) MCMC to look for a maximum a posteriori probability (MAP) estimate.

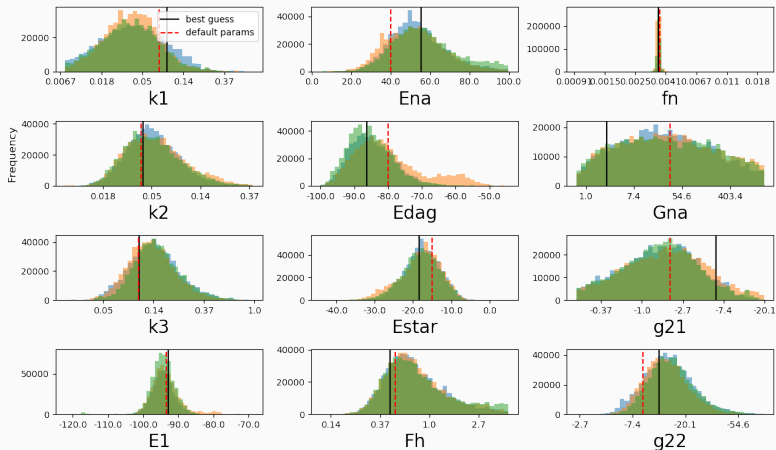
In addition, we can

1. Explore the log-likelihood
2. Quantify the uncertainty

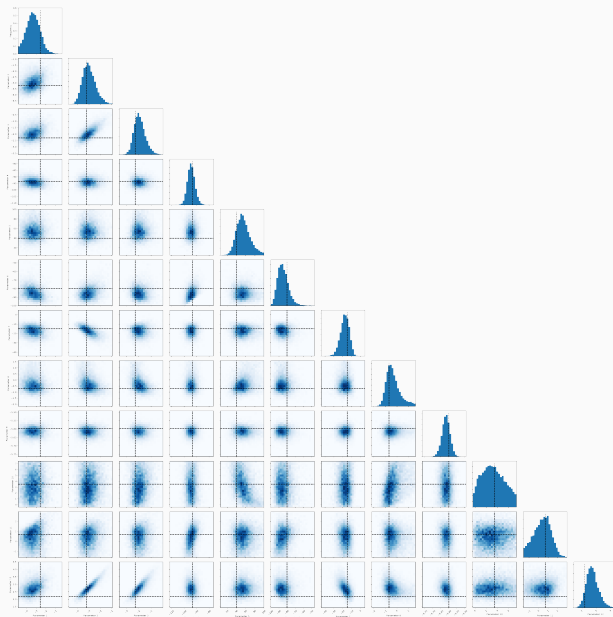
See <https://chi-feng.github.io/mcmc-demo/app.html#AdaptiveMH,banana>

Study obtained posterior distributions

Look at 1D marginal log-posterior distributions



Or look at **2D marginal log-posterior** distributions

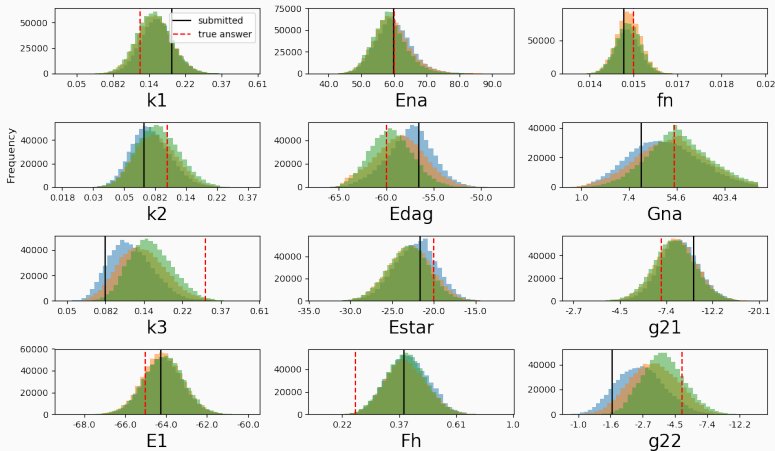


What we learned from synthetic data study?

1. We need a good choice of parameter boundaries
2. Parameter transformation is helpful
3. Running global optimisation can give us a guess of the parameters
4. Running MCMC can get the MAP estimate and get a good idea about uncertainty

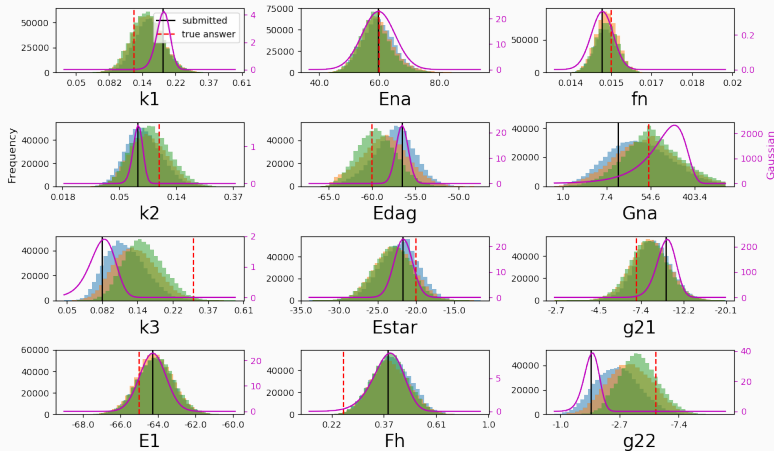
Competition data

MCMC results



Results are what we would expect from our synthetic data study.

Marginal posterior vs precision matrix?



We approximated our **likelihood of the parameters** using precision matrix and Gaussian distribution, but might not be ideal...

Should the noise level be a parameter?

Since we know **exactly**

- how noise ϵ is generated
- what value $\{\sigma_i\}$ should be (SNR=10).

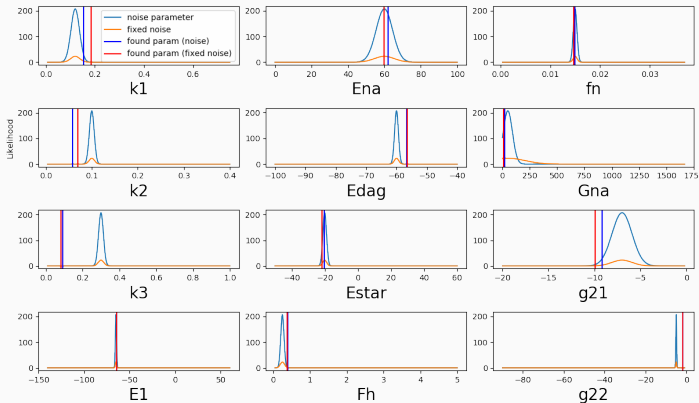
Should the noise level be a parameter?

Since we know **exactly**

- how noise ϵ is generated
- what value $\{\sigma_i\}$ should be (SNR=10).

We can estimate the values of σ and turn

$$p(\mathbf{y}|\Theta, \sigma) \rightarrow p(\mathbf{y}, \sigma|\Theta)$$

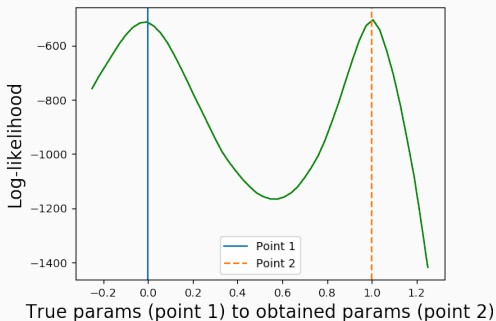


Fixing noise level would

- pros: reduce number of parameters to search
- cons: reduce 'precision'

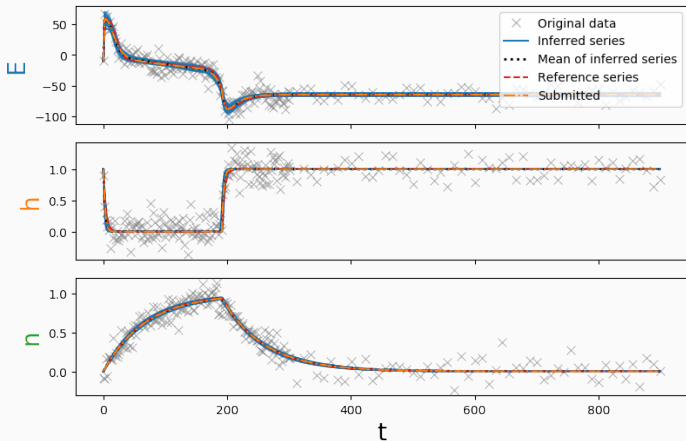
Finally, why we are off...

Plot log-likelihood along the hyperline between our obtained parameters and true parameters



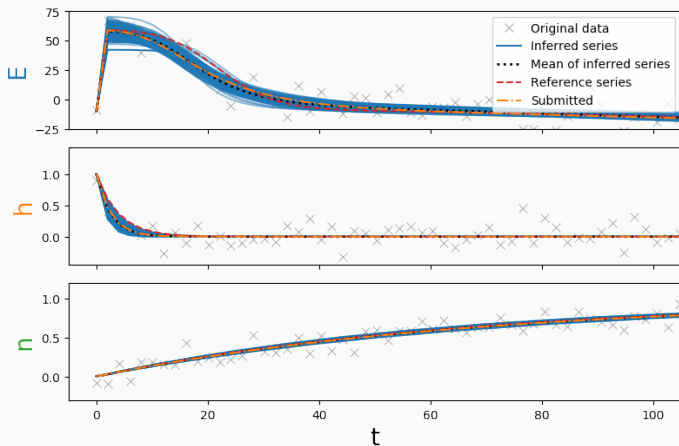
Our obtained best parameters has a better log-likelihood than the true parameters!

Simulations with samples from our distribution



Our obtained best parameters look very alike the true solution, and it is within our posterior predictions!

Simulations with samples from our distribution



Our obtained best parameters look very alike the true solution, and it is within our posterior predictions!

Conclusion

To sum up...

Methods that we used

1. Inspect 1D log-likelihood slices
2. Define parameter transformation and boundaries
3. Run global optimisation (CMA-ES)
4. Run MCMC (Adaptive covariance-MCMC)
5. Study obtained posterior distribution

All can be done within *Pints*

<https://github.com/pints-team/pints>

pints-team / pints

Unwatch

10

Unstar

21

Fork

2

Code

Issues 140

Pull requests 4

Projects 0

Insights

Probabilistic Inference on Noisy Time Series <http://pints.readthedocs.io>

bayesian-methods

inverse-problems

parameter-estimation

numerical-optimization

README.md

build passing

build passing

codecov 100%

pfunk running

launch binder

docs passing

What is Pints?

Pints (Probabilistic Inference on Noisy Time-Series) is a framework for optimisation and bayesian inference problems with ODE models of noisy time-series, such as arise in electrochemistry and cardiac electrophysiology.

Thank You. 😊

Questions?

Extra things could be done...

- Reduce model dimension: fitting spline to E and fit h and n separately
- Try other MCMC algorithms: e.g. Population MCMC, SMC, or tempering methods etc.
- If in real life... we know data do not contain enough information for some of the parameter, we can do experimental-design!

Bayesian inference

What Bayesian inference can provide:

- Estimate the parameters, θ , of a model $y = f(x; \theta)$
- Perform model comparison or compute the evidence
- Make predictions from a model

Bayesian inference (cont.)

- Posterior:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} \propto p(X|\theta)p(\theta)$$

- Evidence (or marginal likelihood):

$$p(X) = \int_{\theta} p(X|\theta)p(\theta)d\theta$$

- Posterior predictive:

$$p(\tilde{x}|X) = \int_{\theta} p(\tilde{x}|\theta)p(\theta|X)d\theta$$

Posterior for our problem

We have posterior

$$\begin{aligned} p(\boldsymbol{\Theta}, \boldsymbol{\sigma} | \mathbf{y}) &= \frac{p(\boldsymbol{\Theta}, \boldsymbol{\sigma}) p(\mathbf{y} | \boldsymbol{\Theta}, \boldsymbol{\sigma})}{p(\mathbf{y})} \\ &\propto p(\boldsymbol{\Theta}) p(\mathbf{y} | \boldsymbol{\Theta}, \boldsymbol{\sigma}) \end{aligned}$$

where $p(\boldsymbol{\Theta})$ is our prior which is defined as

$$p(\boldsymbol{\Theta}) \sim \mathcal{U}(\boldsymbol{\Theta}_{min}, \boldsymbol{\Theta}_{max})$$

where $\boldsymbol{\Theta}_{min}, \boldsymbol{\Theta}_{max}$ are boundaries of our parameter values.