Spring project summary

Mark Blyth

Last meeting

Discussion about single-cell and multi-cell approaches

Single-cell:

- Strong literature precedent for what to expect

Last meeting

Discussion about single-cell and multi-cell approaches

Single-cell:

- Strong literature precedent for what to expect
- Lots of accepted models to test *in-silico*

Last meeting

Discussion about single-cell and multi-cell approaches

Single-cell:

- Strong literature precedent for what to expect
- Lots of accepted models to test *in-silico*
- Easy-to-spot bifurcations

Last meeting

Discussion about single-cell and multi-cell approaches

Single-cell:

- Strong literature precedent for what to expect
- Lots of accepted models to test *in-silico*
- Easy-to-spot bifurcations
  - Hopf, fold both easily detectable with CBC

Last meeting

Discussion about single-cell and multi-cell approaches

Single-cell:

- Strong literature precedent for what to expect
- Lots of accepted models to test *in-silico*
- Easy-to-spot bifurcations
  - Hopf, fold both easily detectable with CBC
- Reuse Bath single-cell microfluidics device

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system
- Search experimentally and numerically for PDE bifurcations

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system
- Search experimentally and numerically for PDE bifurcations
  - Not an area I know much about (yet. . .)

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system
- Search experimentally and numerically for PDE bifurcations
  - Not an area I know much about (yet. . .)
- Can build on work by Krasi's Munich collaborators

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system
- Search experimentally and numerically for PDE bifurcations
  - Not an area I know much about (yet. . .)
- Can build on work by Krasi's Munich collaborators
- Or could reuse Bath microfluidic device

Last meeting

Discussion about single-cell and multi-cell approaches

Multi-cell:

- Assume there's an arbitrarily large number of cells
  - Neural continuum fields (limited descriptive ability)
  - Spatially extended cubic Lienard system
- Search experimentally and numerically for PDE bifurcations
  - Not an area I know much about (yet. . .)
- Can build on work by Krasi's Munich collaborators
- Or could reuse Bath microfluidic device
  - Would require minor alterations to increase spatial resolution

Single- vs multi-cell

Deciding factors:

- No lab access for the forseeable future
  - Work can be guided less by experiments
- Single-cell easier than multi-cell
  - I know enough about single-cell CBC to start working on it

Conclusion: work on single-cell case

Current goals

- Single-cell *in-silico* CBC
- Tutorial-review paper for numerical continuation

Challenges of *in-silico* CBC

Data aren't ideal to work with:

- Real signals are noise-corrupted
  - Difficult to filter off, since spikes contain lots of high-frequency components
  - Hard to run continuation on stochastic and noisy signals
- Neurons are fast-spiking
  - Fourier discretisation won't work
  - Discretisations need to be very high-dimensional, making Jacobian very slow to find

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise
  - Must handle experimental issues (missing measurements, uneven spacing, small amounts of data)

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise
  - Must handle experimental issues (missing measurements, uneven spacing, small amounts of data)
- Surrogate model can then be analysed as desired

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise
  - Must handle experimental issues (missing measurements, uneven spacing, small amounts of data)
- Surrogate model can then be analysed as desired
  - Smooth, clean datasource (no noise, no missing points, full interpolation)

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise
  - Must handle experimental issues (missing measurements, uneven spacing, small amounts of data)
- Surrogate model can then be analysed as desired
  - Smooth, clean datasource (no noise, no missing points, full interpolation)
  - Accurate derivatives, of arbitrary order

Issue 1: noise corruption

Instead of running continuation on noisy signal measurements, let's run it on a surrogate data source

- Assume no knowledge of the system
  - CBC is a model-free method
- Come up with some surrogate model to replace the observed data
  - Must filter out the noise
  - Must handle experimental issues (missing measurements, uneven spacing, small amounts of data)
- Surrogate model can then be analysed as desired
  - Smooth, clean datasource (no noise, no missing points, full interpolation)
  - Accurate derivatives, of arbitrary order
  - Allows for accurate delay embeddings, collocation discretisation – conventional continuation can be used on it

Candidate surrogate models

- Truncated Fourier series

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition
  - ‘Enveloped’ oscillations

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition
  - 'Enveloped' oscillations
- Splines

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition
  - ‘Enveloped’ oscillations
- Splines
  - Piecewise-polynomial model

These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition
  - ‘Enveloped’ oscillations
- Splines
  - Piecewise-polynomial model
- Gaussian process regression

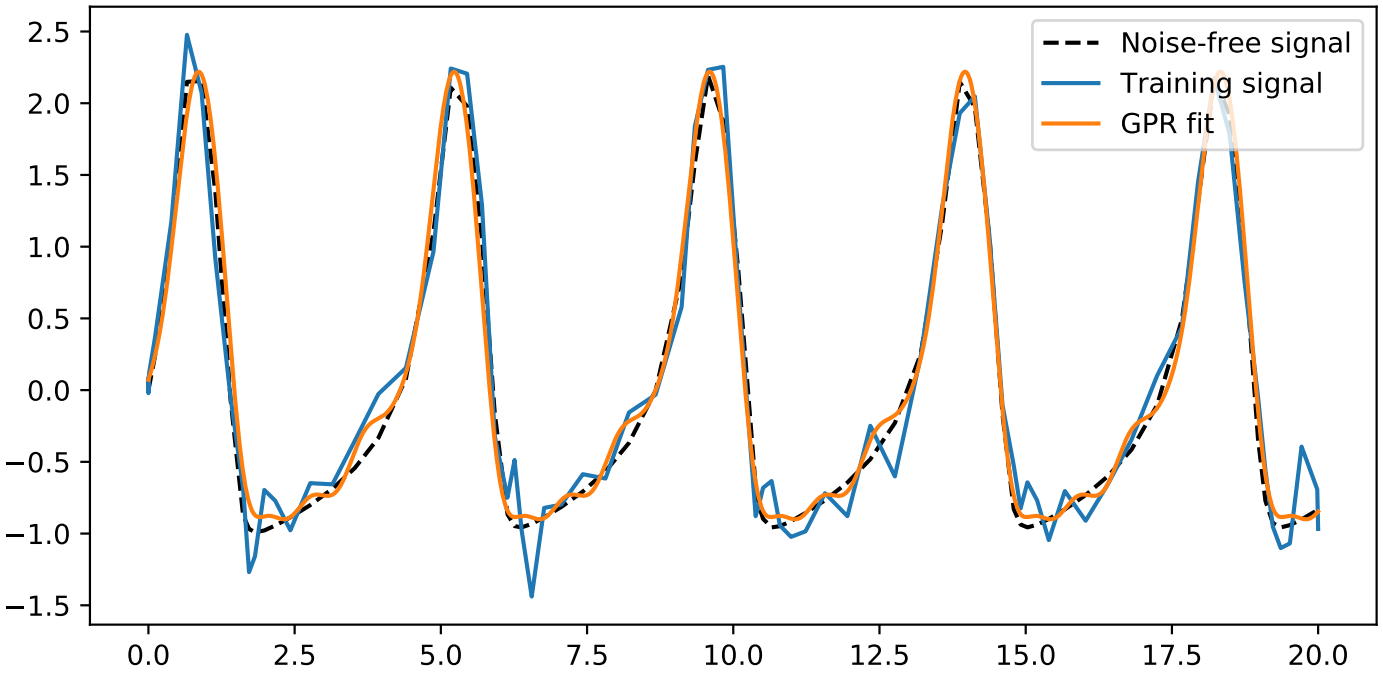
These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Candidate surrogate models

- Truncated Fourier series
  - Currently used in CBC
  - Bad for noisy or spiking data
- Wavelet decomposition
  - 'Enveloped' oscillations
- Splines
  - Piecewise-polynomial model
- Gaussian process regression
  - Statistically optimal, when noise is Gaussian

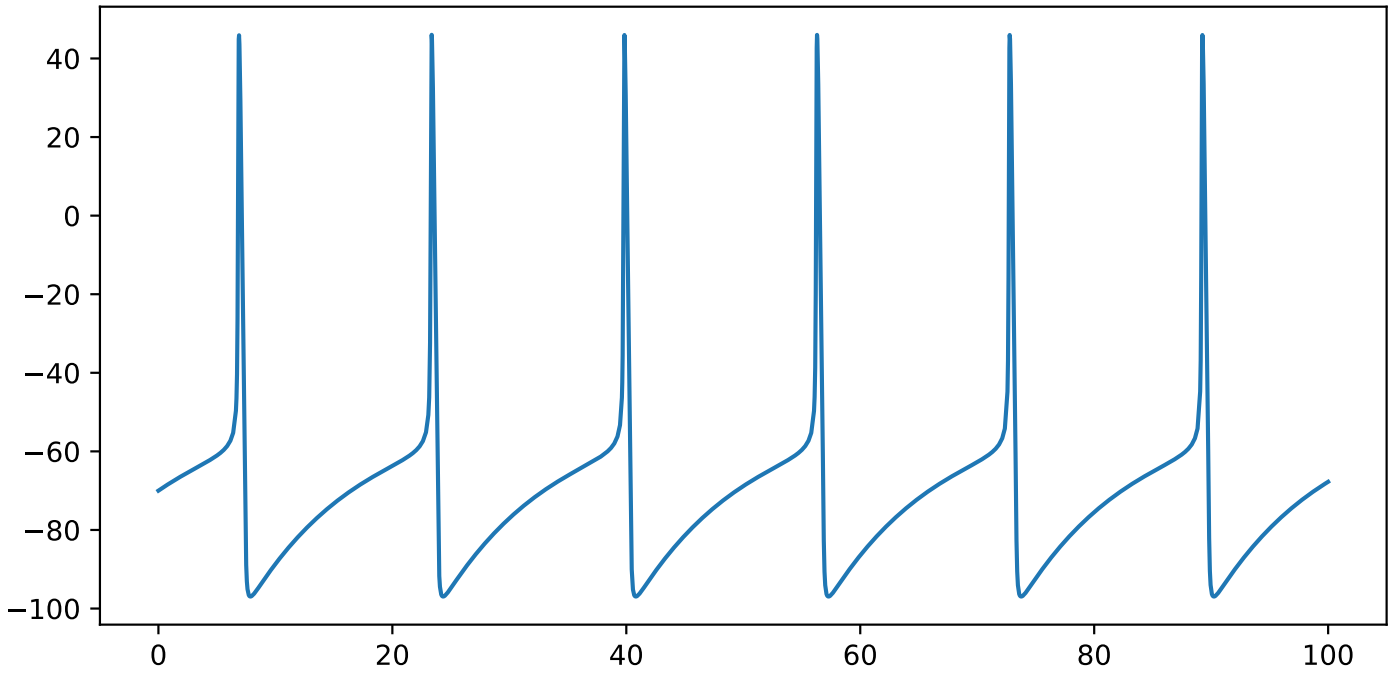
These require no preexisting knowledge *[loosely speaking]*, and work well with sparse data

Gaussian process regression



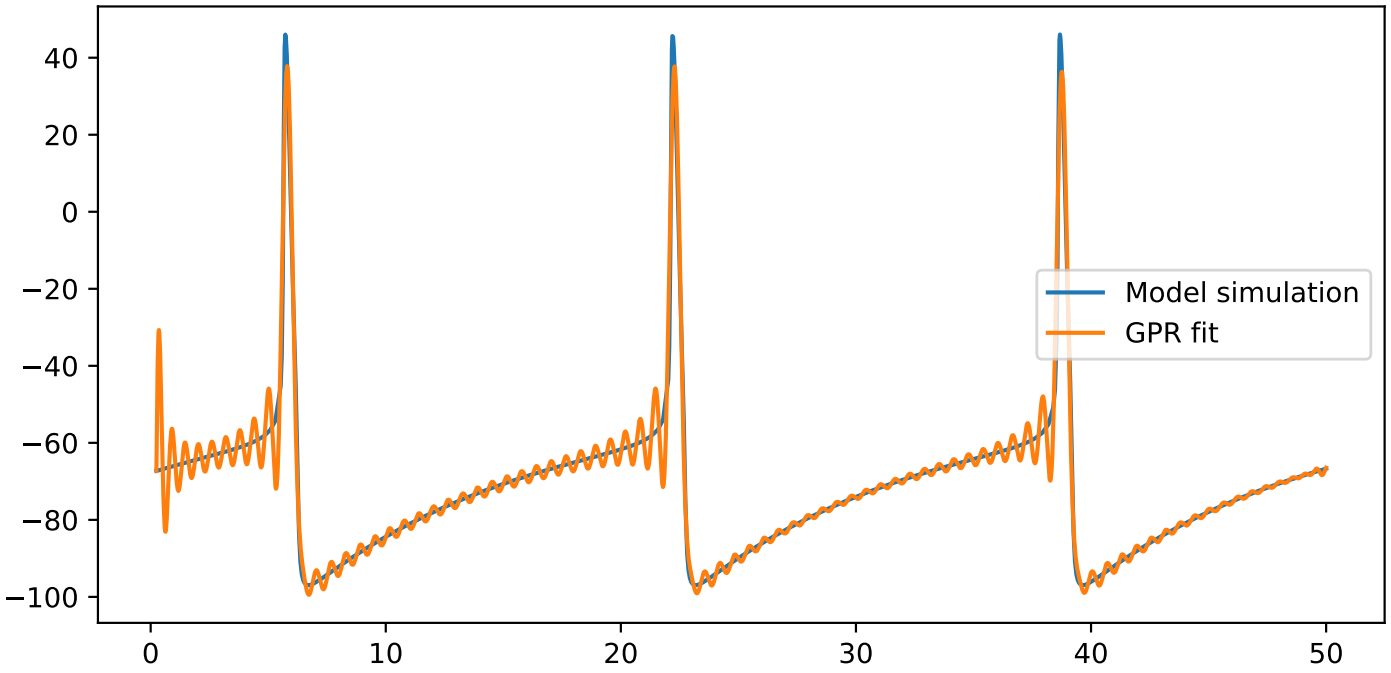
GPR can recover the underlying signal from noise-corrupted observations

Surrogate models



GP surrogate models don't always work well!

Surrogate models



GP surrogate models don't always work well!

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method
  - Kernel specifies a distribution over basis functions

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method
  - Kernel specifies a distribution over basis functions
  - Good kernel choice = good data fit

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method
  - Kernel specifies a distribution over basis functions
  - Good kernel choice = good data fit
- Most kernels are stationary

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method
  - Kernel specifies a distribution over basis functions
  - Good kernel choice = good data fit
- Most kernels are stationary
  - Assumes statistical properties are time-invariant *[they're not]*

Current goal: find a good approach to fitting a surrogate model

Machine learning for dynamical systems

- Current approach: Gaussian process regression; predict new points as an intelligently weighted sum of example points
- Bayesian kernel method
  - Kernel specifies a distribution over basis functions
  - Good kernel choice = good data fit
- Most kernels are stationary
  - Assumes statistical properties are time-invariant *[they're not]*
  - Can't handle the spiking behaviours of neurons

Current goal: find a good approach to fitting a surrogate model

Next questions

- Surrogate models on real data
- Predictor-corrector design
- Stochastic models

Continuation issues

- Discretisation is required to make predictor-corrector methods work
  - Can't run continuation on a function; must discretise it into a vector
- Discretisation has issues when used on fast-spiking data
  - Requires lots of datapoints
  - Slow to find a Jacobian for Newton-iterations
  - High noise-sensitivity
- Surrogate models and discretisation-free predictor-correctors might help overcome these

Alternative continuation approach

Predictor-corrector design:

- We could try discretisation-free predictor steps, using a surrogate model
  - Let $f_i(t)$ be the surrogate model for system behaviours at parameter λ_i
  - Given periodic orbits $f_{i-1},\ f_i$, predict $f_{i+1} = f_i + h\big[f_i - f_{i-1}\big]$
- Corrector step would be harder

Stochastic models

Another challenge: real neurons are stochastic

- Stochasticity introduces new challenges
  - Coherence and stochastic resonance
  - Random attractors
  - Stochastic calculus
  - Not an area I know much about *[yet. . .]*
- Current work: CBC on noise-corrupted simulations
- Next work: CBC on truly stochastic models

Big question: how different would truly stochastic models be?

Goals

Actions:

- Find a surrogate modelling method for neural data
- Attempt a discretisation-free corrector?
- Run CBC on deterministic models, then stochastic

Results:

- Write up surrogate modelling into a conference abstract *[July]*
  - Maybe a conference paper *[September]*
- Use surrogate modelling for an *in-silico* CBC paper *[next year?]*

BONUS: Week's work

- Functional kernel learning now works

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half
  - Could optionally do an error integral, since we have a continuous model

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half
  - Could optionally do an error integral, since we have a continuous model
- Looked into noise-training

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half
  - Could optionally do an error integral, since we have a continuous model
- Looked into noise-training
  - Couldn't find anything

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half
  - Could optionally do an error integral, since we have a continuous model
- Looked into noise-training
  - Couldn't find anything
- Non-stationary kernel is not working

BONUS: Week's work

- Functional kernel learning now works
  - Stationary kernel method
  - Performs well on Fitzhugh-Nagumo
  - Performs badly on Hodgkin-Huxley
- New model validation method
  - Run a high-accuracy solver, for lots of datapoints
  - Downsample
  - Train models on half the datapoints, test them on the other half
  - Could optionally do an error integral, since we have a continuous model
- Looked into noise-training
  - Couldn't find anything
- Non-stationary kernel is not working
- Support vector regression

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data
  - Doesn't always average out noise well

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data
  - Doesn't always average out noise well
- Another idea: ensemble models

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data
  - Doesn't always average out noise well
- Another idea: ensemble models
  - Fit a few different models (GPR, splines, SVR, etc)

Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data
  - Doesn't always average out noise well
- Another idea: ensemble models
  - Fit a few different models (GPR, splines, SVR, etc)
  - Use something analogous to sensory fusion, to combine model predictions

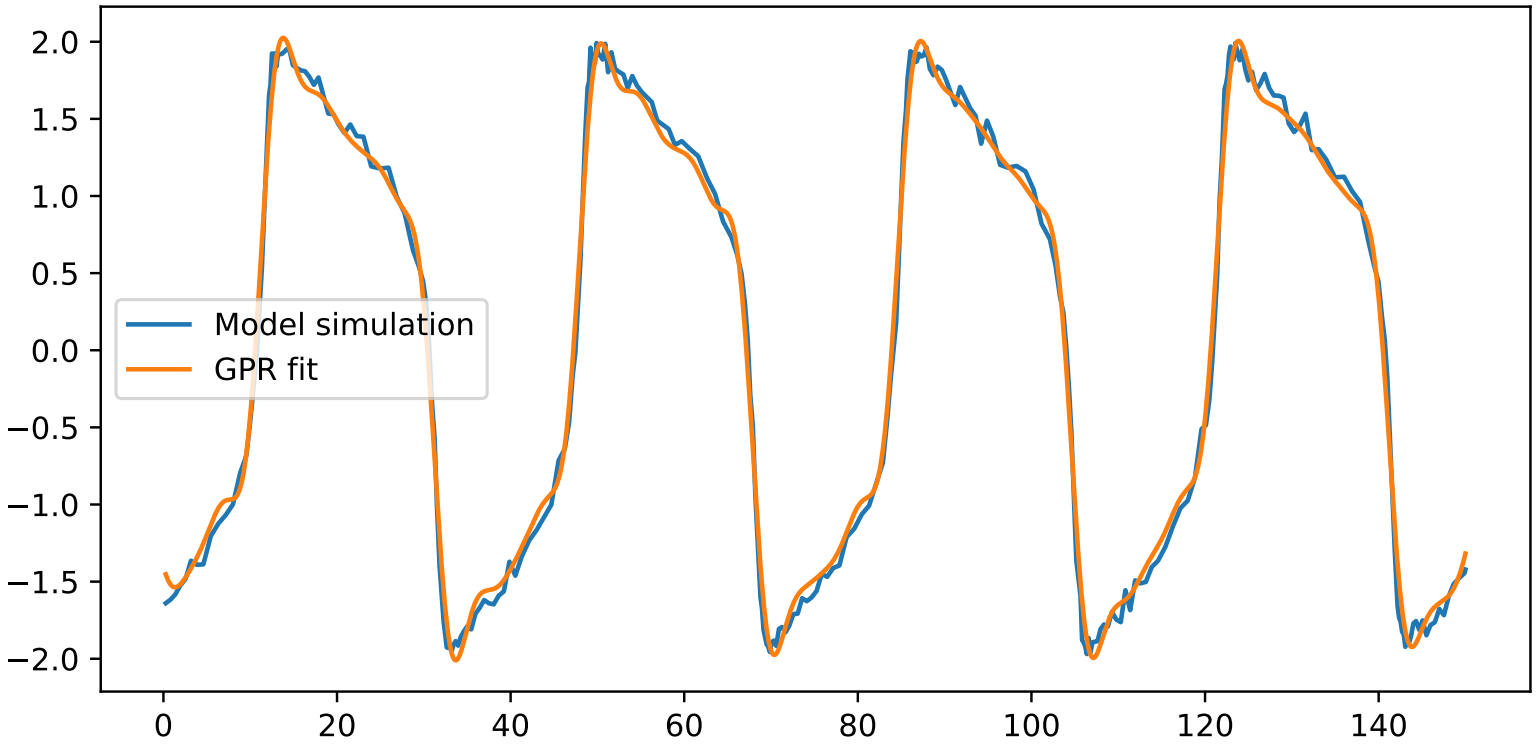
Only worth trying after I've tested all the other regression methods

BONUS: SVR

- SVR is the regression-equivalent of a support vector machine
  - Another popular kernel method
- It works moderately well on neuron data
  - Fast!
  - Fair performance on non-stationary data
  - Doesn't always average out noise well
- Another idea: ensemble models
  - Fit a few different models (GPR, splines, SVR, etc)
  - Use something analogous to sensory fusion, to combine model predictions
  - Gradient boosting: combine several weak learners to make a single strong learner

Only worth trying after I've tested all the other regression methods

BONUS: SVR


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
120
140
Model simulation
GPR fit