Periodic splines discretisation

Mark Blyth

Recent activities

- Emulation methods
- Discretisation methods
- Working on manuscripts
  - Numerical continuations paper
    - Word-count down to 6050+400
  - Emulators and discretisors abstract

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal
   - Useful for applying Fourier *[wavelets?]* to noisy signals

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
    - Fancy *[adaptive]* filtering; removes noise but not signal
    - Useful for applying Fourier *[wavelets?]* to noisy signals
    - Can produce a statistically optimal noise removal, which traditional filtering does not

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal
   - Useful for applying Fourier *[wavelets?]* to noisy signals
   - Can produce a statistically optimal noise removal, which traditional filtering does not
2. Discretise multiple-timescale signals

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal
   - Useful for applying Fourier *[wavelets?]* to noisy signals
   - Can produce a statistically optimal noise removal, which traditional filtering does not
2. Discretise multiple-timescale signals
   - Fourier gives overly-high-dimensional discretisations on spiking signals

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal
   - Useful for applying Fourier *[wavelets?]* to noisy signals
   - Can produce a statistically optimal noise removal, which traditional filtering does not
2. Discretise multiple-timescale signals
   - Fourier gives overly-high-dimensional discretisations on spiking signals
   - Demonstrate an alternative *[lower-dimensional]* method

Conference paper goals

Two separate-but-related goals:

1. Demonstrate the use of surrogate modelling
   - Fancy *[adaptive]* filtering; removes noise but not signal
   - Useful for applying Fourier *[wavelets?]* to noisy signals
   - Can produce a statistically optimal noise removal, which traditional filtering does not
2. Discretise multiple-timescale signals
   - Fourier gives overly-high-dimensional discretisations on spiking signals
   - Demonstrate an alternative *[lower-dimensional]* method
   - Use simple standard methods to do so

Surrogate modelling

This part is done, doesn't require much work

- Shows how to reconstruct a signal from noisy observations
  - Replace real signal with noise-free surrogate model
  - Surrogate model gives a nice clean, interpolable *[no Nyquist cap]* signal
  - Alternative to low-pass filter, that works on spiking signals
- Useful for applying Fourier to noisy signals
- Tested methods: free-knot splines, Gaussian process regression

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit
  - We need more – need to track the full orbit *without* a model

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit
  - We need more – need to track the full orbit *without* a model
- Discretisation is just some lower-dimensional representation of a signal

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit
  - We need more – need to track the full orbit *without* a model
- Discretisation is just some lower-dimensional representation of a signal
  - Eg. parameters of a regression model that reconstructs the signal

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
    - Find polynomials that solve ODEs at points on orbit
    - Tracks a point on an orbit; point on orbit + ODE model = full orbit
    - We need more – need to track the full orbit *without* a model
- Discretisation is just some lower-dimensional representation of a signal
    - Eg. parameters of a regression model that reconstructs the signal
    - Complex method: coefficients of Bayesian periodic splines, or inducing points of sparse periodic GPR *[previous work]*

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit
  - We need more – need to track the full orbit *without* a model
- Discretisation is just some lower-dimensional representation of a signal
  - Eg. parameters of a regression model that reconstructs the signal
  - Complex method: coefficients of Bayesian periodic splines, or inducing points of sparse periodic GPR *[previous work]*
  - Simple method: coefficients of *[non-Bayesian]* periodic splines *[new work]*

New discretisation

Part (1) let us use Fourier on noisy signals, but discretisation will be high-dimensional with neurons

- Collocation etc. requires an ODE model; inapplicable, even on surrogates
  - Find polynomials that solve ODEs at points on orbit
  - Tracks a point on an orbit; point on orbit + ODE model = full orbit
  - We need more – need to track the full orbit *without* a model
- Discretisation is just some lower-dimensional representation of a signal
  - Eg. parameters of a regression model that reconstructs the signal
  - Complex method: coefficients of Bayesian periodic splines, or inducing points of sparse periodic GPR *[previous work]*
  - Simple method: coefficients of *[non-Bayesian]* periodic splines *[new work]*
- NODYCON discretisation method: *[non-Bayesian]* periodic splines

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose 'knot points' $\xi_0, \ldots, \xi_n$

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions
  - C^2 smoothness across whole curve

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions
  - C^2 smoothness across whole curve
  - Curve passes through knots

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
    - Periodic boundary conditions
    - C^2 smoothness across whole curve
    - Curve passes through knots
- BSpline formulation: $f(x) = \sum \beta_i b_i(x)$, b_i = i’th basis spline

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions
  - C^2 smoothness across whole curve
  - Curve passes through knots
- BSpline formulation: $f(x) = \sum \beta_i b_i(x)$, b_i = i’th basis spline
  - Basis splines depend only on knot locations

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \ldots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions
  - C^2 smoothness across whole curve
  - Curve passes through knots
- BSpline formulation: $f(x) = \sum \beta_i b_i(x)$, b_i = i’th basis spline
  - Basis splines depend only on knot locations
  - Knots specify where splines should be flexible

Periodic splines discretisation

Simple non-parametric method for periodic signals

- Choose ‘knot points’ $\xi_0, \dots, \xi_n$
- Fit cubic polynomial between $\xi_i,\ \xi_{i+1}$
  - Periodic boundary conditions
  - C^2 smoothness across whole curve
  - Curve passes through knots
- BSpline formulation: $f(x) = \sum \beta_i b_i(x)$, b_i = i'th basis spline
  - Basis splines depend only on knot locations
  - Knots specify where splines should be flexible
- To discretise, fit knots, b_i at the start; β_i become discretisation

Using periodic splines

1. Find the period

Using periodic splines

1. Find the period
   - Autocorrelation or nonlinear least squares F_0 estimation

Using periodic splines

1. Find the period
    - Autocorrelation or nonlinear least squares F_0 estimation
    - Fourier?

Using periodic splines

1. Find the period
   - Autocorrelation or nonlinear least squares F_0 estimation
   - Fourier?
2. ‘Stack’ periods

Using periodic splines

1. Find the period
   - Autocorrelation or nonlinear least squares F_0 estimation
   - Fourier?
2. 'Stack' periods
   - Re-label data t_is to phase $\phi_i = \frac{t_i}{T} \mod 1$ or $\phi_i = t_i \mod T$

Using periodic splines

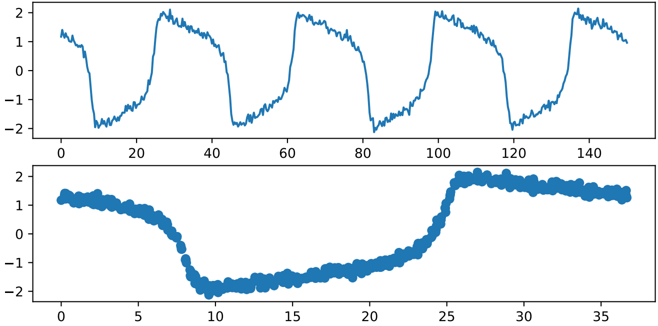
1. Find the period
   - Autocorrelation or nonlinear least squares F_0 estimation
   - Fourier?
2. 'Stack' periods
   - Re-label data t_is to phase $\phi_i = \frac{t_i}{T} \mod 1$ or $\phi_i = t_i \mod T$
3. Build splines model

Using periodic splines

1. Find the period
   - Autocorrelation or nonlinear least squares F_0 estimation
   - Fourier?
2. 'Stack' periods
   - Re-label data t_is to phase $\phi_i = \frac{t_i}{T} \mod 1$ or $\phi_i = t_i \mod T$
3. Build splines model
   - Discretisation = BSpline coefficients

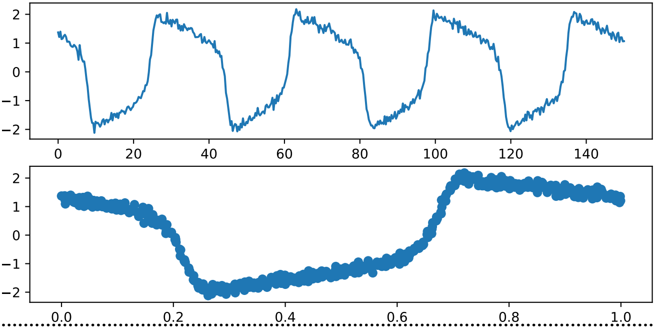
Period stacking

Uses ACF to estimate frequency, then NLS to refine estimate



Period stacking

Uses ACF to estimate frequency, then NLS to refine estimate; removes period from continuation scheme



Fit a splines model

- Challenge: determine knot points

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!
- Well-chosen knot points are needed for a low-dimensional discretisation

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!
- Well-chosen knot points are needed for a low-dimensional discretisation
  - Hard to choose good knots *a priori*; method 2 is automated trial and error

Fit a splines model

- Challenge: determine knot points
  - Given a target 'smoothness' λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!
- Well-chosen knot points are needed for a low-dimensional discretisation
  - Hard to choose good knots *a priori*; method 2 is automated trial and error
  - Standard methods don't give a minimal knotset

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!
- Well-chosen knot points are needed for a low-dimensional discretisation
  - Hard to choose good knots *a priori*; method 2 is automated trial and error
  - Standard methods don’t give a minimal knotset
  - Hard to optimise knotsets due to lots of local minima

Fit a splines model

- Challenge: determine knot points
  - Given a target ‘smoothness’ λ, we can find a knotset and LSQ BSpline curve
  - Standard method; optimizes loss $+ \lambda \times$ total curvature
- Knot selection is still an unanswered problem
  - Smoothing splines approach 1: put a knot at every datapoint
  - Smoothing splines approach 2: start off with few knots, keep adding more until we get target smoothness
  - Overparameterises – neither of these give low-dimensional discretisation!
- Well-chosen knot points are needed for a low-dimensional discretisation
  - Hard to choose good knots *a priori*; method 2 is automated trial and error
  - Standard methods don’t give a minimal knotset
  - Hard to optimise knotsets due to lots of local minima
- Turns out we can make a naive method, that works very well…

Knot fitting

1. Choose the desired number of knots

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’
   - Fitzhugh-Nagumo: 4 turning points, so try 8 knots

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’
   - Fitzhugh-Nagumo: 4 turning points, so try 8 knots
2. Choose knots at random

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’
   - Fitzhugh-Nagumo: 4 turning points, so try 8 knots
2. Choose knots at random
3. Numerically optimize the knot vector

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’
   - Fitzhugh-Nagumo: 4 turning points, so try 8 knots
2. Choose knots at random
3. Numerically optimize the knot vector
4. Repeat steps 2,3 lots, and choose the best result

Knot fitting

1. Choose the desired number of knots
   - A mixture of intuition and experimentation
   - Heuristic: put a knot either side of the signal ‘turning points’
   - Fitzhugh-Nagumo: 4 turning points, so try 8 knots
2. Choose knots at random
3. Numerically optimize the knot vector
4. Repeat steps 2,3 lots, and choose the best result
   - Helps overcome the local minima issue

Knot fitting

This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison…

- Bayesian automatically selects required number of knots

Knot fitting

This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison...

- Bayesian automatically selects required number of knots
  - No trial and error

Knot fitting

This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison...

- Bayesian automatically selects required number of knots
  - No trial and error
  - Takes human out the loop

Knot fitting

This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison...

- Bayesian automatically selects required number of knots
  - No trial and error
  - Takes human out the loop
- Bayesian automatically finds the best knots

Knot fitting

This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison...

- Bayesian automatically selects required number of knots
  - No trial and error
  - Takes human out the loop
- Bayesian automatically finds the best knots
- Bayesian could overcome the period estimation problem *[see later]*

Knot fitting

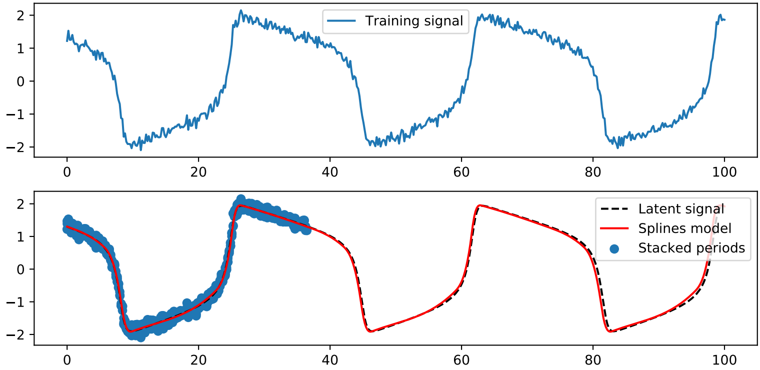
This works surprisingly well:

- Few knots = model quick to fit, easier to optimise
- Nice result would be to analytically derive a LSQ fitting procedure

For comparison. . .

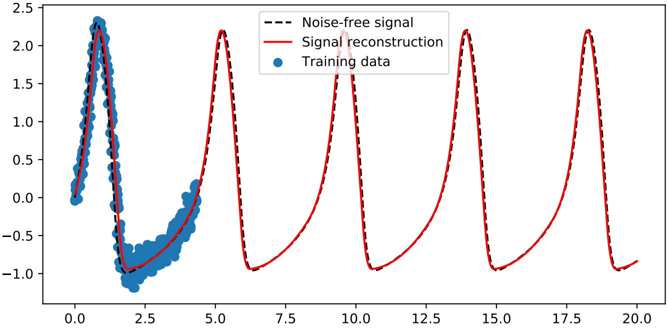
- Bayesian automatically selects required number of knots
  - No trial and error
  - Takes human out the loop
- Bayesian automatically finds the best knots
- Bayesian could overcome the period estimation problem *[see later]*
- This method gets good results much more simply

Optimizer fit



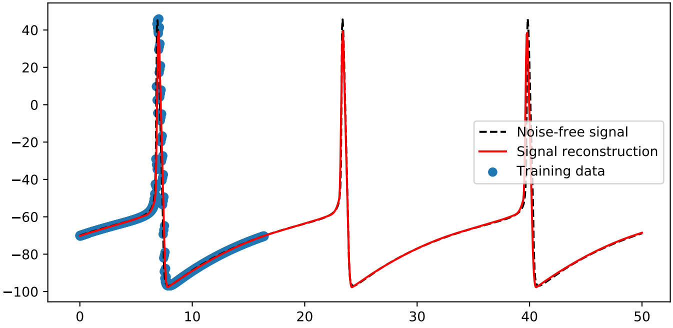
8 interior knots

Optimizer fit



Also works on more neuron-like data

Optimizer fit

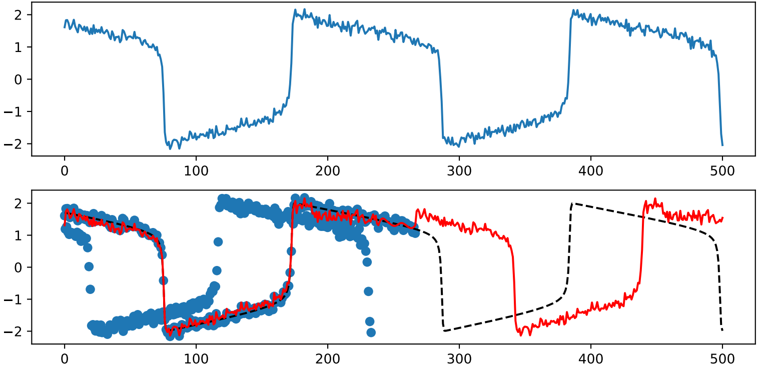


Also works on more neuron-like data

Issue

- Inaccuracies in the period will add up to a big phase shift over time
- Bad period estimate can have disastrous results!

The period-estimation problem



The period-estimation problem

- Increasing the timescale separation 'squares up' the signal, but breaks F0 period estimation
- NLS F0 estimation also uses Fourier harmonics, so breaks on the same signals Fourier discretisation would break on *[only tried one test-case!]*
- Playing with the F0 estimation parameters / methods helps with this, but adds more mysterious hyperparameters
  - Bayesian methods also offer a way around this

Next steps

- Keep working on paper
- Compare reconstruction error for a given number of knots, for Fourier and splines
- Use discretisation in CBC
  - Treat knot positions as a fixed hyperparameter
  - BSpline coefficients become a signal discretisation
- Mini-review knot selection methods
  - Worth discussing the alternative methods in a paper