NODYCON

Abstract review is back!

Last week's work

- Implemented Fourier discretisation
- Compared it against splines
- Some ideas about surrogates and CBC
- Manuscript editing [c. 5400 words]

Discretisation

From last time…

Fitting periodic splines:

1. Find the period

Discretisation

From last time…

Fitting periodic splines:

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

Discretisation

From last time...

Fitting periodic splines:

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

Discretisation

From last time…

Fitting periodic splines:

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

Discretisation

From last time...

Fitting 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 $\phi_i = t_i \mod T$

Discretisation

From last time…

Fitting 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

From last time…

Fitting 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

University of BRISTOL

Discretisation

From last time...

Choosing knots is hard; since we're wanting a minimal knot set...

1. Choose the desired number of knots

Discretisation

From last time. . .

Choosing knots is hard; since we're wanting a minimal knot set. . .

1. Choose the desired number of knots
2. Choose knots at random

Discretisation

From last time...

Choosing knots is hard; since we're wanting a minimal knot set...

1. Choose the desired number of knots
2. Choose knots at random
3. Numerically optimize the knot vector

University of BRISTOL

Discretisation

From last time...

Choosing knots is hard; since we're wanting a minimal knot set...

1. Choose the desired number of knots
2. Choose knots at random
3. Numerically optimize the knot vector
   - Minimise training error of a splines model fitted with these knots

University of BRISTOL

Discretisation

From last time. . .

Choosing knots is hard; since we're wanting a minimal knot set. . .

1. Choose the desired number of knots
2. Choose knots at random
3. Numerically optimize the knot vector
   - Minimise training error of a splines model fitted with these knots
4. Repeat steps 2,3 lots, and choose the best result

Discretisation

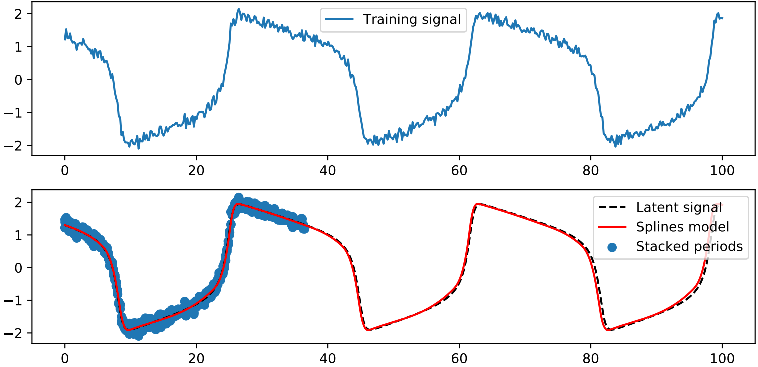
From last time...

Choosing knots is hard; since we're wanting a minimal knot set...

1. Choose the desired number of knots
2. Choose knots at random
3. Numerically optimize the knot vector
   - Minimise training error of a splines model fitted with these knots
4. Repeat steps 2,3 lots, and choose the best result
   - Helps overcome the local minima issue

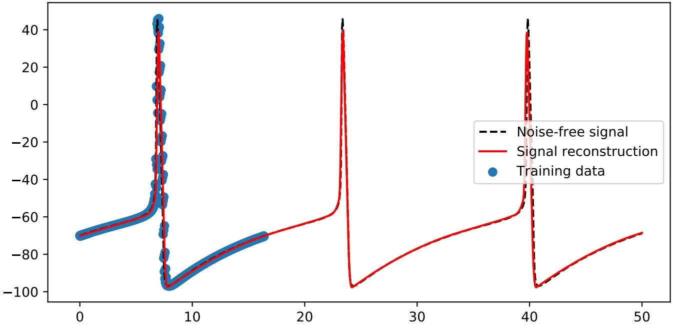
Discretisation

From last time...



Discretisation

From last time...



Discretisation

We can quantify goodness-of-fit:

- Let $\mathrm{reconstruction}(t)$ be the fitted splines model

Discretisation

We can quantify goodness-of-fit:

- Let $\mathrm{reconstruction}(t)$ be the fitted splines model
- Let $\mathrm{signal}(t)$ be the signal we wish to discretise

Discretisation

We can quantify goodness-of-fit:

- Let $\mathrm{reconstruction}(t)$ be the fitted splines model
- Let $\mathrm{signal}(t)$ be the signal we wish to discretise
- Let $\mathrm{error}(t) = \mathrm{signal}(t) - \mathrm{reconstruction}(t)$

University of BRISTOL

Discretisation

We can quantify goodness-of-fit:

- Let $\mathrm{reconstruction}(t)$ be the fitted splines model
- Let $\mathrm{signal}(t)$ be the signal we wish to discretise
- Let $\mathrm{error}(t) = \mathrm{signal}(t) - \mathrm{reconstruction}(t)$
- Goodness-of-fit = $\int_0^T [\mathrm{error}(t)]^2 \, \mathrm{d}t$

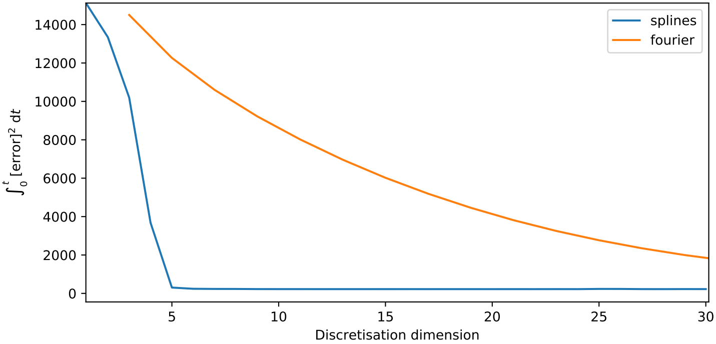
Discretisation

We can quantify goodness-of-fit:

- Let $\mathrm{reconstruction}(t)$ be the fitted splines model
- Let $\mathrm{signal}(t)$ be the signal we wish to discretise
- Let $\mathrm{error}(t) = \mathrm{signal}(t) - \mathrm{reconstruction}(t)$
- Goodness-of-fit = $\int_0^T [\mathrm{error}(t)]^2 \, \mathrm{d}t$
- This gives a metric for comparing splines, Fourier, etc.

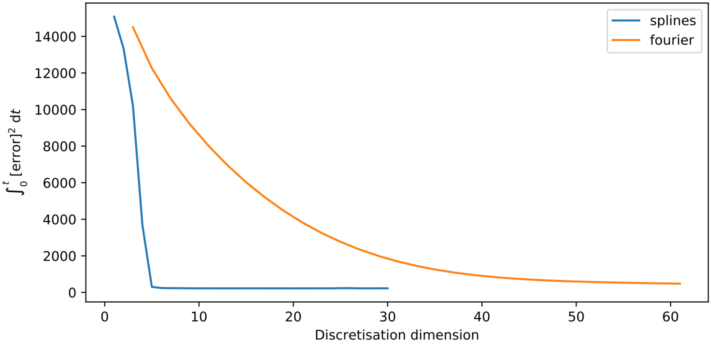
Splines vs Fourier

Hodgkin-Huxley neuron; error decays *significantly* faster with splines



Splines vs Fourier

Hodgkin-Huxley neuron; error decays *significantly* faster with splines



University of BRISTOL

Open problems

- Robustness
  - Does it break down on stochastic systems? Eg. if data aren't fully periodic
  - Do we need a human in the loop, to manually adjust anything?

- Locality
  - Knots are fitted / work well for λ_0; can the same knots model $\lambda_1, \lambda_2, \ldots, \lambda_i$?
  - (They need to for predicting the next PO in an iteration)

CBC approach

Question: there's several ways of performing CBC; which is best for this?

Methods for PO CBC

Method 1. 'Easy' approach for harmonically forced systems

- Set the response amplitude

Methods for PO CBC

Method 1. ‘Easy’ approach for harmonically forced systems

- Set the response amplitude
- Find the input forcing amplitude that gives that response

Methods for PO CBC

Method 1. ‘Easy’ approach for harmonically forced systems

- Set the response amplitude
- Find the input forcing amplitude that gives that response
- Lumps the bifurcation parameter together with the control action

Methods for PO CBC

Method 1. 'Easy' approach for harmonically forced systems

- Set the response amplitude
- Find the input forcing amplitude that gives that response
- Lumps the bifurcation parameter together with the control action
  - Fast and efficient iteration scheme

Methods for PO CBC

Method 1. ‘Easy’ approach for harmonically forced systems

- Set the response amplitude
- Find the input forcing amplitude that gives that response
- Lumps the bifurcation parameter together with the control action
  - Fast and efficient iteration scheme
  - Similar approach exists for continuing equilibria

Method 1 issues

We don’t have a harmonically forced system

University of BRISTOL

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the ‘IO map’ from control-target to system output

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the 'IO map' from control-target to system output
  - Says what the system output is, for any given control target

University of BRISTOL

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the ‘IO map’ from control-target to system output
  - Says what the system output is, for any given control target
- Map fixed point means control-target = system output

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the 'IO map' from control-target to system output
  - Says what the system output is, for any given control target
- Map fixed point means control-target = system output
- *[Claim:]* map fixed point occurs only when there's non-invasive control

University of BRISTOL

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the 'IO map' from control-target to system output
  - Says what the system output is, for any given control target
- Map fixed point means control-target = system output
- *[Claim:]* map fixed point occurs only when there's non-invasive control
- Use Newton iterations to solve for fixed point of discretised map

Methods for PO CBC

Method 2. Harder, fully general approach *[Sieber Krauskopf]*

- Define the ‘IO map’ from control-target to system output
  - Says what the system output is, for any given control target
- Map fixed point means control-target = system output
- *[Claim:]* map fixed point occurs only when there’s non-invasive control
- Use Newton iterations to solve for fixed point of discretised map
- Solution is the noninvasive control target

Method 2 issues

I think this is wrong…

- Integral control $\implies$ no proportional error

Method 2 issues

I think this is wrong…

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target

Method 2 issues

I think this is wrong...

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target

Method 2 issues

I think this is wrong...

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target
- System output == control target $\implies$ every control target is a fixed point of the IO map

Method 2 issues

I think this is wrong...

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target
- System output == control target $\implies$ every control target is a fixed point of the IO map
  - Control target and system output are identical for all targets

Method 2 issues

I think this is wrong…

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target
- System output == control target $\implies$ every control target is a fixed point of the IO map
  - Control target and system output are identical for all targets
- Every point is a fixed point $\implies$ can't find noninvasive control by solving the map

Method 2 issues

I think this is wrong...

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target
- System output == control target $\implies$ every control target is a fixed point of the IO map
  - Control target and system output are identical for all targets
- Every point is a fixed point $\implies$ can't find noninvasive control by solving the map
- My claim: IO-map fixed point is a necessary but not sufficient condition for noninvasiveness

Method 2 issues

I think this is wrong...

- Integral control $\implies$ no proportional error
- No proportional error $\implies$ system output == control target
  - System output exactly tracks control target
- System output == control target $\implies$ every control target is a fixed point of the IO map
  - Control target and system output are identical for all targets
- Every point is a fixed point $\implies$ can't find noninvasive control by solving the map
- My claim: IO-map fixed point is a necessary but not sufficient condition for noninvasiveness
- Feels like a big claim to say the paper's wrong, but I haven't found any way to resolve this...

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies \exists$ proportional error

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\Longrightarrow$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive

University of BRISTOL

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies \exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation
  - Let $\mathbf{x} =$ system output discretisation

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation
  - Let $\mathbf{x} =$ system output discretisation
  - Equality $\implies$ no proportional error $\implies$ zero control action, noninvasiveness, etc.

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation
  - Let $\mathbf{x} =$ system output discretisation
  - Equality $\implies$ no proportional error $\implies$ zero control action, noninvasiveness, etc.
  - $\mathbf{u}^*$ = $\mathbf{x}$ can therefore be solved for noninvasive $\mathbf{u}^*$

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation
  - Let $\mathbf{x} =$ system output discretisation
  - Equality $\implies$ no proportional error $\implies$ zero control action, noninvasiveness, etc.
  - $\mathbf{u}^*$ = $\mathbf{x}$ can therefore be solved for noninvasive $\mathbf{u}^*$
- This is exactly the method proposed in Sieber Krauskopf

Method 2 solutions

Approach 1. Only use P, PD control

- No integral controller $\implies$ $\exists$ proportional error
- Proportional error = 0 $\iff$ control action is zero (noninvasiveness)
  - System output = control target $\iff$ control is noninvasive
  - IO map fixed point $\iff$ control is noninvasive
- Can then use Newton iterations to solve for noninvasiveness
  - Let $\mathbf{u}^* =$ control target discretisation
  - Let $\mathbf{x} =$ system output discretisation
  - Equality $\implies$ no proportional error $\implies$ zero control action, noninvasiveness, etc.
  - $\mathbf{u}^*$ = $\mathbf{x}$ can therefore be solved for noninvasive $\mathbf{u}^*$
- This is exactly the method proposed in Sieber Krauskopf
- Downside: locked into a single control method

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control

University of BRISTOL

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
- Solve for $\mathbf{u}^*$ that sets total control action to zero

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
- Solve for $\mathbf{u}^*$ that sets total control action to zero
  - Underdetermined – n inputs, one output; minimisation problem

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
- Solve for $\mathbf{u}^*$ that sets total control action to zero
  - Underdetermined – n inputs, one output; minimisation problem
  - Eg. gradient descent on $\mathbf{u}^*$ with Broyden Jacobian update

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
- Solve for $\mathbf{u}^*$ that sets total control action to zero
  - Underdetermined – n inputs, one output; minimisation problem
  - Eg. gradient descent on $\mathbf{u}^*$ with Broyden Jacobian update
  - This is similar to standard Newton iterations

University of BRISTOL

Method 2 solutions

Approach 2. Reformulate the zero problem

- Explicitly solve for noninvasive control
- Total control action = $\int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
- Solve for $\mathbf{u}^*$ that sets total control action to zero
  - Underdetermined – n inputs, one output; minimisation problem
  - Eg. gradient descent on $\mathbf{u}^*$ with Broyden Jacobian update
  - This is similar to standard Newton iterations
- Downsides: minimisation might be slower; no literature precedent

Optimal design of gradient-descent method

Maybe we don't need to find $\mathbf{u}^*$ that sets control action to zero...

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there

University of BRISTOL

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model
  - Moves calculations out of experiment and onto computer, where they can be performed much faster

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model
  - Moves calculations out of experiment and onto computer, where they can be performed much faster
  - No need for experimental Newton iterations, gradient descent, Jacobians, finite differences

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model
  - Moves calculations out of experiment and onto computer, where they can be performed much faster
  - No need for experimental Newton iterations, gradient descent, Jacobians, finite differences
- Fit local model on 'maximally informative' datapoints

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model
  - Moves calculations out of experiment and onto computer, where they can be performed much faster
  - No need for experimental Newton iterations, gradient descent, Jacobians, finite differences
- Fit local model on 'maximally informative' datapoints
  - Choose datapoints that maximise our certainty of the minima location

Optimal design of gradient-descent method

- Much like tracking bifurcations optimally – don't need to see the actual bifurcation point, as long as we're confident it's there
- Find a local surrogate model of total invasiveness $I(\mathbf{u}^*) = \int u(\mathbf{u}^*, t)^2 \mathrm{d}t$
  - Maps a discretisation $\mathbf{u}^*$ to total control action required to stabilise it
  - Quantifies invasiveness of target $\mathbf{u}^*$
  - $I(\mathbf{u}^*) = 0 \implies u^*$ is noninvasive, so natural system behaviour
- Solve for $I(\mathrm{u}^*) = 0$ on the local surrogate model
  - Moves calculations out of experiment and onto computer, where they can be performed much faster
  - No need for experimental Newton iterations, gradient descent, Jacobians, finite differences
- Fit local model on 'maximally informative' datapoints
  - Choose datapoints that maximise our certainty of the minima location
- Experimentally test $\mathbf{u}_i^*$ that solves $I(\mathbf{u}^*) = 0$, to ensure that's the noninvasive solution

Proposed route

Initially, use PD control, IO map with Newton iterations

- Standard method, so don't have to develop anything new
- Need to use PD control, but that also means no need to develop any fancy controller
- Gets results quickly!

If PD doesn't work well, develop the surrogate gradient descent method

- Makes it truly control-strategy independent
- Extends fully-general CBC to systems that are harder to control with PD

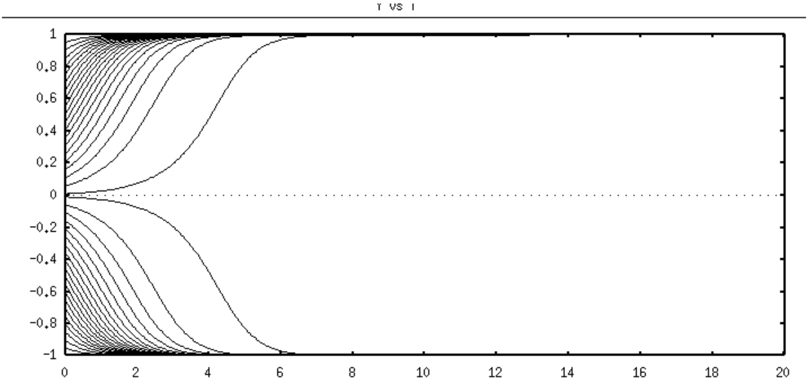
An aside

Interesting aside: control-free continuation

- Some systems are hard to control
- Can we run CBC *without needing a controller*?

Control-free continuation

We can deduce the existence of an unstable equilibrium



Control-free continuation

- Stable features are easy to spot – the system converges to them

Control-free continuation

- Stable features are easy to spot – the system converges to them
- We can often deduce the existence of unstable features

Control-free continuation

- Stable features are easy to spot – the system converges to them
- We can often deduce the existence of unstable features
- Easy method: fit a local surrogate, find unstable features from that

Control-free continuation

- Stable features are easy to spot – the system converges to them
- We can often deduce the existence of unstable features
- Easy method: fit a local surrogate, find unstable features from that
  - Eg. fit a neural ODE / neural GP to the previous bistable system

Control-free continuation

- Stable features are easy to spot – the system converges to them
- We can often deduce the existence of unstable features
- Easy method: fit a local surrogate, find unstable features from that
    - Eg. fit a neural ODE / neural GP to the previous bistable system
    - Simple root-finding for locating equilibria

Control-free continuation

- Stable features are easy to spot – the system converges to them
- We can often deduce the existence of unstable features
- Easy method: fit a local surrogate, find unstable features from that
  - Eg. fit a neural ODE / neural GP to the previous bistable system
  - Simple root-finding for locating equilibria
  - More optimal-experimental-design opportunities, for increasing confidence at equilibrium locations

Control-free continuation

General method:

1. Collect some data

University of BRISTOL

Control-free continuation

General method:

1. Collect some data
   - Set the system running

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models
   - If we keep the system near some feature of interest, the models will be locally accurate there

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models
   - If we keep the system near some feature of interest, the models will be locally accurate there
   - Can use standard methods to locate unstable POs from the models

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models
   - If we keep the system near some feature of interest, the models will be locally accurate there
   - Can use standard methods to locate unstable POs from the models
   - Could use standard continuation on the models, or. . .

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models
   - If we keep the system near some feature of interest, the models will be locally accurate there
   - Can use standard methods to locate unstable POs from the models
   - Could use standard continuation on the models, or...
   - ...Could find POs, UPOs at lots of parameter values, to track them without control or continuation

University of BRISTOL

Control-free continuation

General method:

1. Collect some data
   - Set the system running
   - Every time instabilities drive it away from the region of interest, restart with the system where we want it
2. Reconstruct state space from recorded time series
3. Fit a neural / GP ODE model to reconstructed state space
4. Run standard analyses on the models
   - If we keep the system near some feature of interest, the models will be locally accurate there
   - Can use standard methods to locate unstable POs from the models
   - Could use standard continuation on the models, or...
   - ...Could find POs, UPOs at lots of parameter values, to track them without control or continuation

Nice example of what surrogates could do

Back on topic. . .

When are surrogates useful?

- Conference abstract discusses surrogates
- Paper needs to make their usage cases clear

When are surrogates useful?

Slow signals:

- No high harmonics $\implies$ Fourier discretisation works fine $\implies$ no need for a novel discretisation
- No high harmonics $\implies$ low-pass filtering works fine $\implies$ no need for a surrogate

No need for surrogates

When are surrogates useful?

Fast signals:

- Lots of high harmonics $\implies$ Fourier discretisation doesn't work $\implies$ need a novel discretisation
- Novel discretisation $\implies$ no need for a surrogate as well

No need for surrogates

When are surrogates useful?

Medium-speed signals:

- Can be more efficiently discretised with splines than Fourier
- Harmonically forced systems: faster to use Fourier iterations than Newton
  - Splines discretise more efficiently, but we still use Fourier
- Enough HF harmonics that we wouldn't want to use LP filtering $\implies$ we need a surrogate

This is surrogates usage case

When are surrogates useful?

- Better to use Fourier iterations than Newton iterations on harmonically forced systems
- Surrogates are useful for Fourier iteration on faster signals
- Example usage: cleaning noise from a highly nonlinear forced Duffing

When are surrogates useful?

Type	Harmonically forced	Unforced
Slow signal	Fourier iter's, LP filters	Newton iter's, LP filters
Fast signal	Fourier iter's, surrogates	Newton iter's, novel discretisation

The two new methods complement each other; one for Newton iter's, one for Fourier iter's; paper should make this clear

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control
- Conference paper needs to be clear / explicit about when surrogates, new discretisations are useful

University of BRISTOL

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control
- Conference paper needs to be clear / explicit about when surrogates, new discretisations are useful
- Interesting aside 1: we need a different approach to use non-PD control with the most general CBC method

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control
- Conference paper needs to be clear / explicit about when surrogates, new discretisations are useful
- Interesting aside 1: we need a different approach to use non-PD control with the most general CBC method
  - Less general methods (where parameter and control action can be lumped together) don't require this

University of BRISTOL

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control
- Conference paper needs to be clear / explicit about when surrogates, new discretisations are useful
- Interesting aside 1: we need a different approach to use non-PD control with the most general CBC method
  - Less general methods (where parameter and control action can be lumped together) don't require this
  - Lots of room for interesting optimal experimental design

Summary

- CBC implementation should use Newton iterations, spline discretisation, PD control
- Conference paper needs to be clear / explicit about when surrogates, new discretisations are useful
- Interesting aside 1: we need a different approach to use non-PD control with the most general CBC method
  - Less general methods (where parameter and control action can be lumped together) don't require this
  - Lots of room for interesting optimal experimental design
- Interesting aside 2: might be possible to run CBC without a controller?

Next steps

1. Test splines generalisation, robustness
2. Write up results so far into a paper
3. Demonstrate splines with CBC

Alternative: skip step 1 altogether and jump in with CBC simulation

Generalisation and robustness [again]

- Generalisation
  - Knots are fitted / work well for λ_0; can the same knots model $\lambda_1, \lambda_2, \ldots, \lambda_i$?
  - (They need to for predicting the next PO in an iteration)

- Robustness
  - Does it break down on stochastic systems? Eg. if data aren't fully periodic
  - Do we need a human in the loop, to manually adjust anything?

Key dates

- Bath maths ML conference, week of Aug.3rd - 7th
- Conference paper submission, September ~~11th~~ 25th
- Goal: start conference paper writing mid-August