Papers, splines, and other ideas

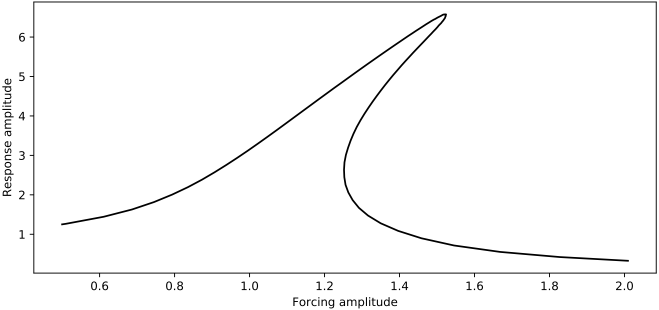
Mark Blyth

Presentation overview

- Splines discretisation progress
- Some ideas: projects that will enhance CBC, and make it more powerful for studying neurons
  - Lots of exciting ideas; these ones are limited to the project-relevant ones

CBC code progress

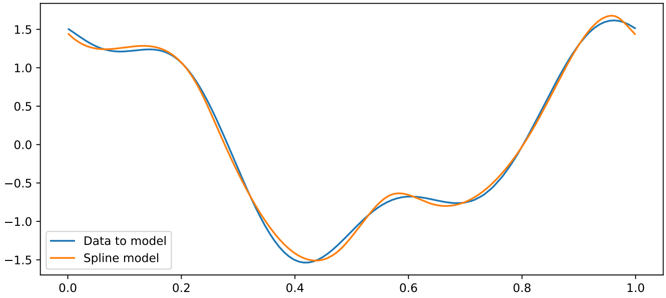
- Rewritten for a new solver
- Works for Fourier



Splines discretisation progress

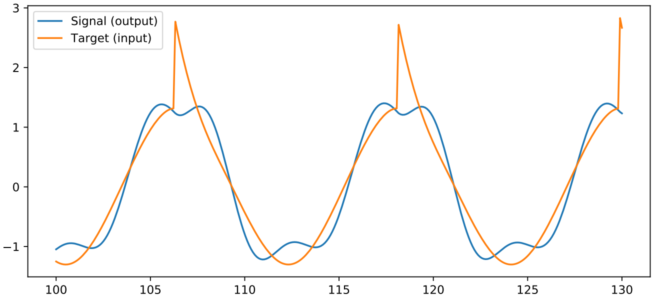
- Newton iterations fail to converge with splines discretisation
- Working hypothesis: splines models are structurally unstable; small perturbations cause big changes
  - Finite differences evaluates gradients by using small perturbations
  - Finite differences perturbations lead to discontinuous changes in the model

Splines discretisation progress



Spline model describes data fairly accurately

Splines discretisation progress



Control-target input is discontinuous during Jacobian estimation

Splines discretisation progress

- Newton iterations fail to converge with splines discretisation
- Working hypothesis: splines models are structurally unstable; small perturbations cause big changes
  - Finite differences evaluates gradients by using small perturbations
  - Finite differences perturbations lead to discontinuous changes in the model
- Currently a best-guess conjecture
  - Need to play about more to see if this is actually the case
  - Need to test different finite-differences stepsizes

Novel discretisors

Ideal: journal paper comparing…

- Splines
  - Knot selection methods?
- Wavelets
- Collocation
- Fourier

Compare for noise-robustness, dimensionality, computational speed

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

- Extension 2: reversible-jump MCMC for Fourier model identification

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

- Extension 2: reversible-jump MCMC for Fourier model identification
  - Automatically determine best number of Fourier harmonics for any given signal

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

- Extension 2: reversible-jump MCMC for Fourier model identification
  - Automatically determine best number of Fourier harmonics for any given signal
  - Similar to how BARS auto-chooses knot numbers and locations, but instead Fourier harmonics

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

- Extension 2: reversible-jump MCMC for Fourier model identification
  - Automatically determine best number of Fourier harmonics for any given signal
  - Similar to how BARS auto-chooses knot numbers and locations, but instead Fourier harmonics

- Why Bayesian?

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace ‘simple’ splines / wavelets / collocation with Bayesian alternative
  - ‘Intelligently’ select the best discretisor

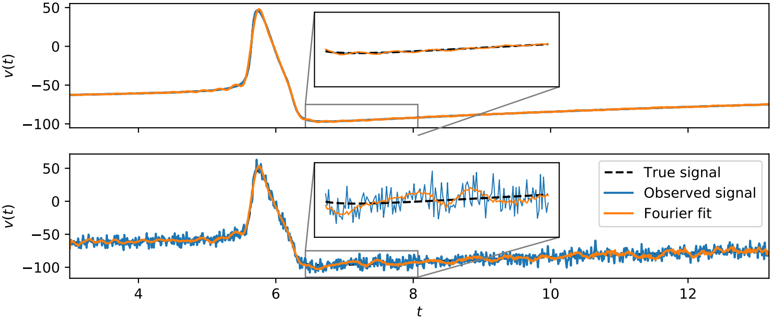
- Extension 2: reversible-jump MCMC for Fourier model identification
  - Automatically determine best number of Fourier harmonics for any given signal
  - Similar to how BARS auto-chooses knot numbers and locations, but instead Fourier harmonics

- Why Bayesian?
  - Automatic Occam’s razor

Idea 1: Extended novel discretisors

- Extension 1 [discussed in meetings]: selecting discretisation method based on model evidence
  - Produce Bayesian inference method for Fourier harmonics
  - Replace 'simple' splines / wavelets / collocation with Bayesian alternative
  - 'Intelligently' select the best discretisor
- Extension 2: reversible-jump MCMC for Fourier model identification
  - Automatically determine best number of Fourier harmonics for any given signal
  - Similar to how BARS auto-chooses knot numbers and locations, but instead Fourier harmonics
- Why Bayesian?
  - Automatic Occam's razor
  - Optimally balances goodness-of-fit against model complexity; ensures simplest, most generalizable discretisations; statistically optimal

The sweetspot problem



- Sweetspot problem: too few harmonics don't fit; too many harmonics overfit noise
- Bayesian Fourier model selection would find the sweet spot by automatically trading goodness-of-fit against model complexity

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic
  - Subtract out surrogate's fundamental harmonic

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic
  - Subtract out surrogate's fundamental harmonic
  - Replace fundamental harmonic with harmonic forcing term

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic
  - Subtract out surrogate's fundamental harmonic
  - Replace fundamental harmonic with harmonic forcing term
  - Feed the modified surrogate back into the system as the new control target

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic
  - Subtract out surrogate's fundamental harmonic
  - Replace fundamental harmonic with harmonic forcing term
  - Feed the modified surrogate back into the system as the new control target

- Should converge rapidly, implicitly represent an infinite number of harmonics, improve computational speed by avoiding DFT, improve noise-robustness by employing surrogate noise-filtering

Idea 2: discretisation-free CBC

- As removed from NODYCON abstract
  - Project surrogate of system output onto fundamental harmonic
  - Subtract out surrogate's fundamental harmonic
  - Replace fundamental harmonic with harmonic forcing term
  - Feed the modified surrogate back into the system as the new control target
- Should converge rapidly, implicitly represent an infinite number of harmonics, improve computational speed by avoiding DFT, improve noise-robustness by employing surrogate noise-filtering
- Further extension: something resembling a Picard iteration

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven’t converged, or just because of noise?

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven’t converged, or just because of noise?
- Solver approach: input - output = 0

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - 'True' stochasticity – measurement noise means evaluating the IO map for the same input won't necessarily give the same output
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven’t converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2
- These two approaches are identical in the noise-free case

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - 'True' stochasticity – measurement noise means evaluating the IO map for the same input won't necessarily give the same output
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it's a solution to the other

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - 'True' stochasticity – measurement noise means evaluating the IO map for the same input won't necessarily give the same output
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes $(\text{input - output})^2$
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it's a solution to the other
- Minimizer approach is better in the noisy case

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - 'True' stochasticity – measurement noise means evaluating the IO map for the same input won't necessarily give the same output
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it's a solution to the other
- Minimizer approach is better in the noisy case
  - If the system is subject to noise or stochastics, output will never truly equal input

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven’t converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it’s a solution to the other
- Minimizer approach is better in the noisy case
  - If the system is subject to noise or stochastics, output will never truly equal input
  - Noninvasive solution will therefore minimize control action *[minimizer works]*, but will not result in input-output=0 *[solver won’t work]*

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - ‘True’ stochasticity – measurement noise means evaluating the IO map for the same input won’t necessarily give the same output
  - How do we decide when we’ve found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven’t converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes (input - output)^2
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it’s a solution to the other
- Minimizer approach is better in the noisy case
  - If the system is subject to noise or stochastics, output will never truly equal input
  - Noninvasive solution will therefore minimize control action *[minimizer works]*, but will not result in input-output=0 *[solver won’t work]*
  - Bonus: *lots* of research on stochastic gradient descent (stochastic optimization) to help with this

Idea 3: noise-robust numerical solvers

- CBC strategies solve for input=output
  - Harder to solve if output is corrupted by noise, or subject to stochastic dynamics – both prevalent with neurons
  - 'True' stochasticity – measurement noise means evaluating the IO map for the same input won't necessarily give the same output
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Solver approach: input - output = 0
- Minimizer approach: find input that minimizes $(\text{input - output})^2$
- These two approaches are identical in the noise-free case
  - Control target is a solution to one if and only if it's a solution to the other
- Minimizer approach is better in the noisy case
  - If the system is subject to noise or stochastics, output will never truly equal input
  - Noninvasive solution will therefore minimize control action *[minimizer works]*, but will not result in input-output=0 *[solver won't work]*
  - Bonus: *lots* of research on stochastic gradient descent (stochastic optimization) to help with this
- Research output: comparison of convergence time, noise-robustness for

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\to$output map

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\rightarrow$output map
  - Choose next function evaluation based on knowledge-gradient or expected improvement

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\rightarrow$output map
  - Choose next function evaluation based on knowledge-gradient or expected improvement
  - With a probabilistic approach, this should converge much faster than Newton, and in a noise-robust way

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\rightarrow$output map
  - Choose next function evaluation based on knowledge-gradient or expected improvement
  - With a probabilistic approach, this should converge much faster than Newton, and in a noise-robust way
  - Use a splines model for the input-output map, as GPR doesn't handle MIMO well

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\to$output map
  - Choose next function evaluation based on knowledge-gradient or expected improvement
  - With a probabilistic approach, this should converge much faster than Newton, and in a noise-robust way
  - Use a splines model for the input-output map, as GPR doesn't handle MIMO well
  - Extension of Barton-Renson conference paper: uses explicit MIMO models, knowledge gradients, explicit noise modelling

Idea 4: alternative noise-robust numerical solvers

- Covers same issues as before, but in a different way
  - How do we decide when we've found a solution to a system, if the system is stochastic?
  - How do we decide if input$\neq$ output because we haven't converged, or just because of noise?
- Newton iterations are slow, even with Jacobian updates
  - Especially big issue for larger-dimensional discretisations (as with neurons!)
- IDEA: surrogates-based approach
  - Build a surrogate of the input$\rightarrow$output map
  - Choose next function evaluation based on knowledge-gradient or expected improvement
  - With a probabilistic approach, this should converge much faster than Newton, and in a noise-robust way
  - Use a splines model for the input-output map, as GPR doesn't handle MIMO well
  - Extension of Barton-Renson conference paper: uses explicit MIMO models, knowledge gradients, explicit noise modelling
- Aim main paper at a numerical methods audience; possibly additional journal paper comparing CBC speeds, noise-robustnesses for various numerical solvers

Results

- Efficient discretisation methods

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals
- Intelligent discretisation methods

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals

- Intelligent discretisation methods
  - Bayesian'ly determine the best discretisor class *[eg. Fourier vs. splines]*, best discretisation approach within that class *[eg. number of knots / harmonics]*

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals

- Intelligent discretisation methods
  - Bayesian'ly determine the best discretisor class *[eg. Fourier vs. splines]*, best discretisation approach within that class *[eg. number of knots / harmonics]*
  - Allows the optimal discretisation of spiking, neuron-like signals, without needing a human in the loop

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals
- Intelligent discretisation methods
  - Bayesian'ly determine the best discretisor class *[eg. Fourier vs. splines]*, best discretisation approach within that class *[eg. number of knots / harmonics]*
  - Allows the optimal discretisation of spiking, neuron-like signals, without needing a human in the loop
  - Ensures CBC is accurate, by guaranteeing discretisation actually describes the target signal

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals
- Intelligent discretisation methods
  - Bayesian'ly determine the best discretisor class *[eg. Fourier vs. splines]*, best discretisation approach within that class *[eg. number of knots / harmonics]*
  - Allows the optimal discretisation of spiking, neuron-like signals, without needing a human in the loop
  - Ensures CBC is accurate, by guaranteeing discretisation actually describes the target signal
- Noise-robust solvers for accurate CBC calculations

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Results

- Efficient discretisation methods
  - Alternative approaches to Fourier (splines, collocation, wavelets)
  - Allows the discretisation of spiking, neuron-like signals
- Intelligent discretisation methods
  - Bayesian'ly determine the best discretisor class *[eg. Fourier vs. splines]*, best discretisation approach within that class *[eg. number of knots / harmonics]*
  - Allows the optimal discretisation of spiking, neuron-like signals, without needing a human in the loop
  - Ensures CBC is accurate, by guaranteeing discretisation actually describes the target signal
- Noise-robust solvers for accurate CBC calculations
  - Solve the CBC equations accurately, even faced with measurement noise *[and stochasticity?]*

These produce a neuron-capable continuation procedure; paired with a control strategy, they'll allow neuronal CBC

Testing the new methods

Any CBC rigs I could demonstrate these methods on?

Next steps

- Edit NODYCON paper
- Edit continuation paper
- Keep working on splines discretisation code
- Test other discretisor methods