Autonymous CBC

Mark Blyth

Week’s activities

- Implemented van der Pol (vdP) CBC with Fourier
  - Doesn’t work
- Implemented adaptive-knots splines
  - Requried to make BSpline CBC work with vdP
- Implemented vdP CBC with BSplines
  - Doesn’t work
- Read about practical bursters
- Wrote some notes on discretisors

Reading

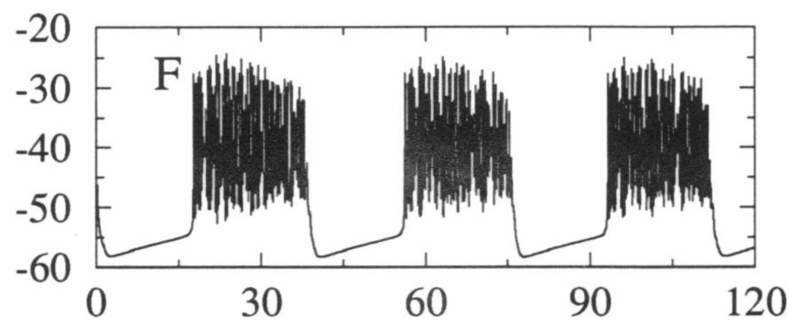
Read about why single cells don't burst but populations do

- Bursting requires quite specific parameter values
- Real cells have enough variation in these parameters that it's rare to find one that can burst
- Coupled cells can take on the dynamics of their averaged parameter values
  - Individual cells rarely lie within the bursting parameter range
  - Collections of cells average out their parameter values and allow bursting

Issue: bursts in networks are a bit messy

- Individual cells in the network no longer show nice neat bursts
- Can't define any nice neat control target
- Limits the scope to which CBC can be applied

Noisy bursts



- These results are from simulations
- Question for Krasi is how biologically realistic this is
  - Can we ever get ‘nice’ bursts in real cells?

Autonymous CBC

- Continuation vector contains parameter, period, discretisation
  - Initialisation parameters chosen by user
  - Initialisation period determined by zero-crossings
  - Initialisation discretisation found by discretising uncontrolled system output at initial parameters
- Continuation equations enforce...
  - Input discretisation = output discretisation
  - Pseudo-arclength condition
  - Integral phase condition, with previous accepted solution as a reference $v(t)$

$$\Psi[x^*] = \int_0^1 \left\langle x^* \left(\frac{t}{T_{x^*}} \right), \dot{v} \left(\frac{t}{T_v} \right) \right\rangle \mathrm{d}t$$

Fourier vdP

Not much benefit to nonadaptive-knots vdP BSpline CBC

- vdP signal is very nonlinear; nonadaptive would need lots of coefficients
- If we're using lots, may as well use the simpler Fourier method

Tested out vdP-Fourier

- Didn't work, Jacobian somehow ended up singular
- SciPy solvers didn't work either

Didn't put much effort into testing why this happens

- Splines would be easier to test
  - Nicer numbers (consistently $\mathcal{O}(1)$)
  - Fewer of them (lower-dimensional discretisation)
- Decided to skip the numerics checking and jump straight to splines

Part 1: initialising adaptive knots

BSplines need adaptive knots to be useful on vdP; this works as follows

Choose good knots at initialisation

- Let ξ be a knot vector, x_0 be the initialiser signal, $\hat{x}(\xi)$ its least-squares spline approximation
- Find $\mathrm{argmin}\xi \quad \|x_0 - \hat{x}(\xi)\|_2^2$, over signal samples
  - Initialise knots as uniformly distributed random variables
  - Numerically optimize
  - Repeat lots of times to avoid local minima
  - Choose the best result

Part 2: Adapting the adaptive knots

Knots are updated after each prediction/correction step

- Initial knots will already be a good guess of optimal knots
  - They were optimal for the previous signal
  - We assume optimal knot set changes smoothly with signal
- Run a single optimization step
  - Fit knots to the newly accepted output signal
- Update current knots to newly optimized result

Part 3: Using adaptive knots

Must re-discretise at each prediction/corrector step

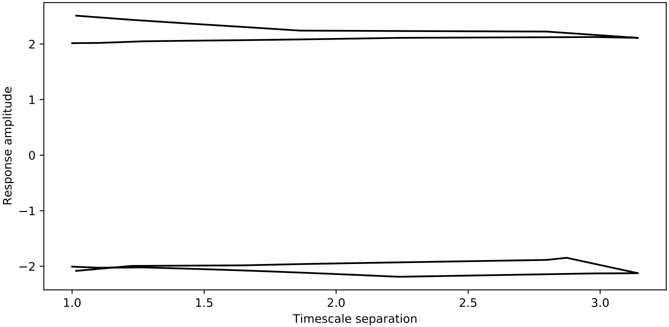
- Discretisation must be consistent between all vectors in any given prediction/corretion step
- Take accepted solutions from previous two results and project on to the current knot set
- Use the rediscretised solutions for secant prediction

Goodness-of-fit of each knot optimization result gives us a good check that the discretisation is still valid

- If goodness-of-fit becomes bad, we might need more knots in the discretisation
- I hold discretisation size fixed, but it could easily be varied

Adaptive-BSpline autonymous CBC

It doesn't work very well



Possible issues and improvements

Code is only just finished

- Haven't had time to test stepsizes, Jacobian methods, solvers, etc.
- Might be as simple as a different stepsize

Phase constraints

Maybe the continuation equations aren't very good?

- No particular reason why Galerkin projections should be robust at estimating period
  - Counter-argument: it's worked fine for other people!
- Phase constraint makes the coefficients similar to their previous values; doesn't encode anything particularly interesting or meaningful
  - Must be specified to get a unique solution, however doesn't encode anything interesting; doesn't help us find a solution in a numerically robust way
- Collocation is a good way to find oscillation periods
  - Keeps extending a solution boundary around the limit cycle, until it finds the period that makes the two boundaries meet
  - Phase constraint encodes where to put one of the boundaries

Collocation might be a generally more robust way of doing things

Next steps

- Play some more with the code and see if anything interesting can be made to happen
- If it can't, try collocation instead
- Also, keep reading and writing