# Knots, collocation, writing

Mark Blyth

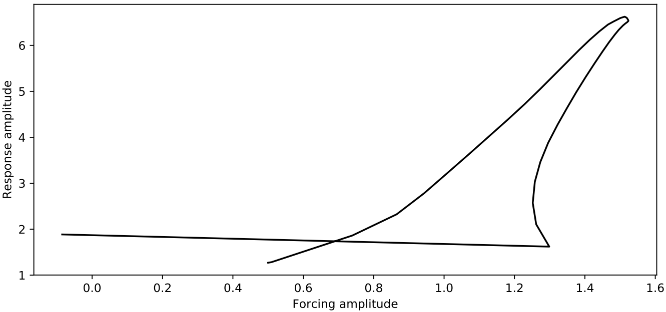
# Week's activities

- Spline-Newton CBC with more knots
  - Goal: more numerical stability
  - Different results, but not really any better
- Looked into collocation references
- Started annual review report

# Newton iteration issues

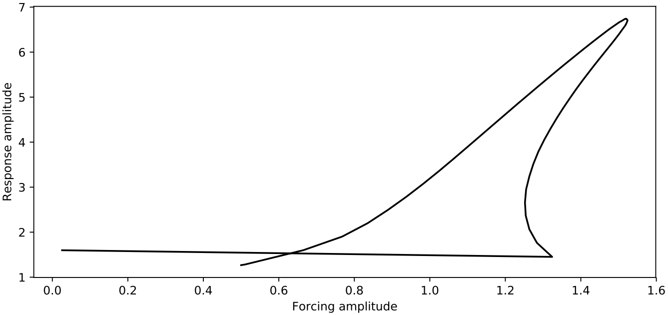
- Converged solution doesn't actually solve continuation equations
  - Newton iterations should, but don't, give a vector that, when passed to the continuation equations, give a zero output
  - More iterations don't help
  - Different convergence criteria don't improve things
- Solution jumps
  - Jacobian is always well-conditioned
  - Probably a finite-differences issue?
- Idea: try more knots!
  - More knots = more attainable accuracy = perhaps better chance of finding a solution

# Baseline: 5 knots



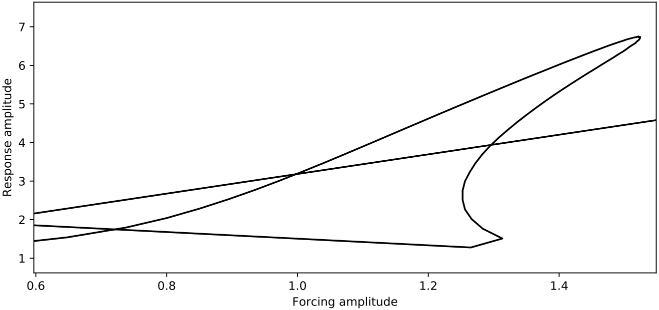

- Minimum 3 interior knots for a valid BSpline model
- Solution jumps
- Converged Newton-iteration vectors don’t solve the continuation equations accurately

# 20 knots




- Simulation is notably slower to run
- Solution still jumps
- Converged Newton-iteration vectors solve system to higher accuracy than before

# 30 knots




- Simulation is even slower to run
- Solution jumps at about the same place
- Converged Newton-iteration vectors again solve system to higher accuracy

# Things to note

- SciPy solvers still get a solution with 5 knots
  - Means the equations can be solved, but not by a Newton solver
  - Doesn't quite make sense. . .
- Solution is jumping after the second fold
  - I'd have expected this to be one of the more numerically stable places

# Other things to try

- Adaptive stepsize
  - Should allow greater accuracy around difficult regions (eg. folds)
- Adaptive knots
  - Essential for ‘harder’ (eg. neuronal) signals
  - (Presumably) unimportant here
- Idea: Jacobian checking
  - Use a secant predictor to estimate the next Jacobian
  - If the finite-differences Jacobian differs much from the secant prediction, try FD again with a new stepsize
  - Extension: adaptive-stepsize finite differences

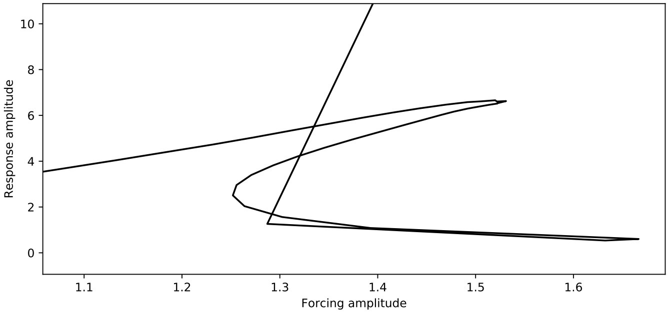
# Effects of control gain

Another thing to try: increasing the control gain

- Was originally using $K_p = 1$
  - This worked fine for Duffing Fourier
  - Keeping $K_p$ as low as possible seems to give the best-possible accuracy with Fourier

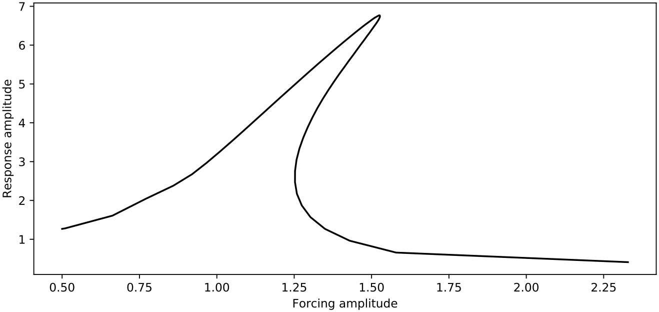
- Intuitively, increasing $K_p$ would make it *harder* to find a correct solution, not easier
  - In limit, large $K_p$ means every control target solves the continuation equations, whether or not they're noninvasive
  - Intuition: smaller $K_p$ gives a larger gradient at the fixed-point, and therefore a more accurate solution can be found

# 5 knots, $K_p = 2$




- Unexpected: slight improvement in results
- Using $K_p = 2$ delayed the 'jump'
  - Jump region is controllable with $Kp = 1$ for Fourier, but not splines
- Still doesn't explain why non-Newton solvers could find a solution at $K_p = 1$!
  - If the SciPy solver can find a solution at $K_p = 1$, why can't a Newton solver?

# 20 knots, $K_p = 2$




- Solution takes a huge leap at the end, but it's a correct leap
- It works, but doesn't seem like it should; opposite result to what was expected
- Still doesn't explain what was going wrong with $K_p = 1$

# Standard continuation

Other work: considering a ‘standard’ (non-control-based) continuation of the Duffing oscillator

- Removes any issue from controllers being weird
- Simplifies down to just a discretisation and predictor/corrector problem
- Plan of action:
  1. Learn about collocation and periodic-orbit continuation *[in progress]*
  2. Learn about BSpline collocation for BVPs *[in progress]*
  3. Combine them
  4. Add in the extras (BSpline periodicity structure, choice of knots, choice of collocation meshpoints, if any)
  5. Code up and test
  6. Make the step 4 extras adaptive

# Next steps

- Lab group presentation
- Annual review report
- Later...
  - More collocation
  - ‘Standard’ continuation
  - Adaptive algos