

# Investigating spline numerics

Mark Blyth

---

## Week's goals

- ✂ Fix splines CBC code
  - ▶ Done for the non-adaptive case
- ✂ Investigate whether the code now works
  - ▶ It doesn't
- ✂ Writing (continuation paper, extended conference paper)
  - ▶ Happening slowly

---

## CBC code issues

1. Results displayed incorrectly

---

## CBC code issues

### 1. Results displayed incorrectly

- ▶ Duffing output amplitude was being calculated incorrectly

---

## CBC code issues

### 1. Results displayed incorrectly

- ▶ Duffing output amplitude was being calculated incorrectly
- ▶ Fixed now!

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
  - ▶ Implemented my own exterior knot calculation method



---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
  - ▶ Implemented my own exterior knot calculation method
  - ▶ Issue: SciPy gives different results when used in different ways; most robust method seems to combine my exterior knots and SciPy exterior knots

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
  - ▶ Implemented my own exterior knot calculation method
  - ▶ Issue: SciPy gives different results when used in different ways; most robust method seems to combine my exterior knots and SciPy exterior knots
3. Adaptive knots might not be used correctly

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
  - ▶ Implemented my own exterior knot calculation method
  - ▶ Issue: SciPy gives different results when used in different ways; most robust method seems to combine my exterior knots and SciPy exterior knots
3. Adaptive knots might not be used correctly
  - ▶ Definitely were correct in the original script

---

## CBC code issues

1. Results displayed incorrectly
  - ▶ Duffing output amplitude was being calculated incorrectly
  - ▶ Fixed now!
2. Exterior knot computation was hacky
  - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
  - ▶ Implemented my own exterior knot calculation method
  - ▶ Issue: SciPy gives different results when used in different ways; most robust method seems to combine my exterior knots and SciPy exterior knots
3. Adaptive knots might not be used correctly
  - ▶ Definitely were correct in the original script
  - ▶ Might not be in the rewrite

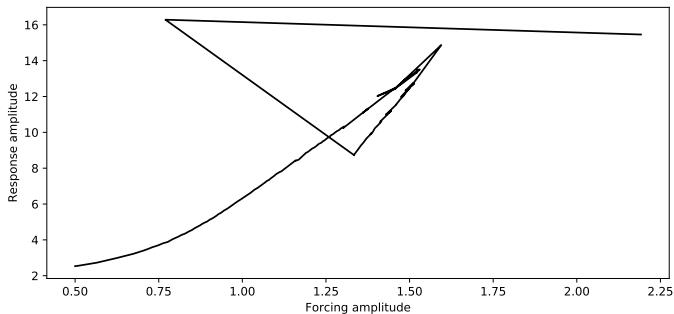
---

## CBC code issues

1. Results displayed incorrectly
    - ▶ Duffing output amplitude was being calculated incorrectly
    - ▶ Fixed now!
  2. Exterior knot computation was hacky
    - ▶ Exterior knots are necessary to fit data at endpoints; I was letting SciPy recalculate them each time
    - ▶ Implemented my own exterior knot calculation method
    - ▶ Issue: SciPy gives different results when used in different ways; most robust method seems to combine my exterior knots and SciPy exterior knots
  3. Adaptive knots might not be used correctly
    - ▶ Definitely were correct in the original script
    - ▶ Might not be in the rewrite
    - ▶ Haven't checked this; avoiding the adaptive knots method for now
-

---

## Fixed plotting

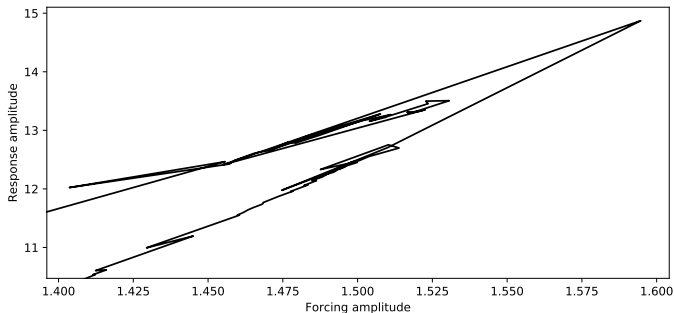


Stepsize 0.1; 3 interior knots; FDSS 0.2

---

---

## Fixed plotting: zoomed in

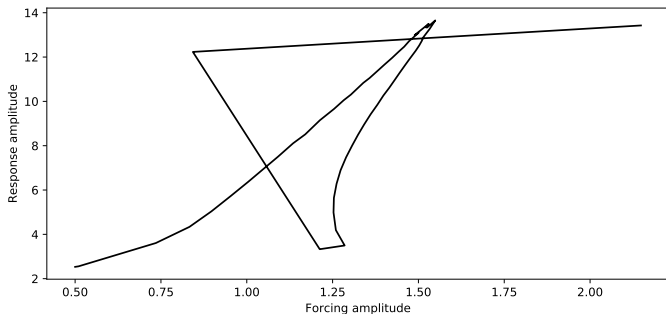


Bigger stepsize?

---

---

## Fixed plotting, bigger steps



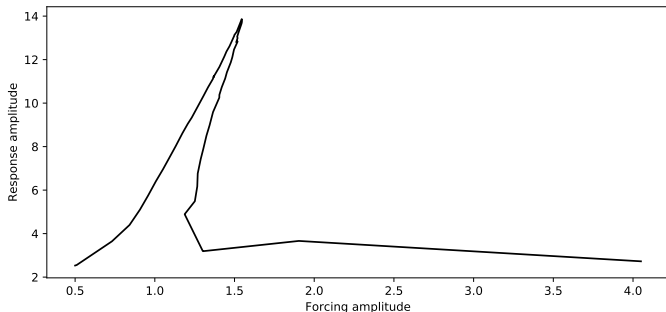
Stepsize 1; 3 interior knots; FDSS 0.2; better, but solution jumps; let's change exterior knots

---



---

## New exterior knots

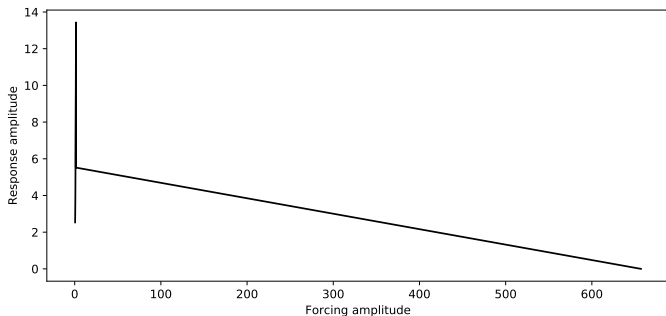


Stepsize 1; 3 interior knots; FDSS 0.5; fixed exterior knots; **converged vectors are often not actually solutions**

---

---

## New convergence criteria

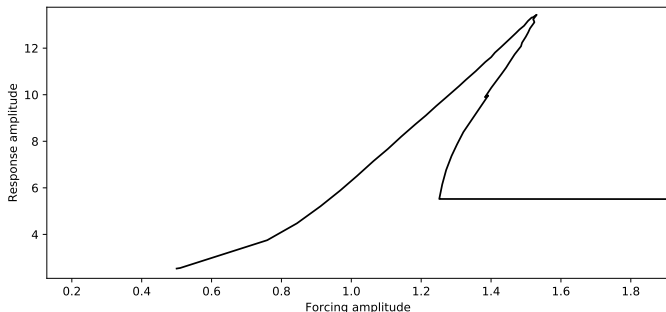


Same hyperparameters as before; convergence declared when continuation equation output has a norm below  $5e-2$

---

---

## New convergence criteria

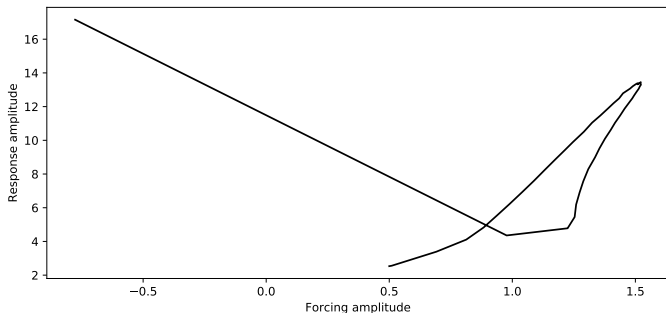


Zoom in to before the jump; more knots might make the results better

---

---

## Old convergence criteria, but more knots

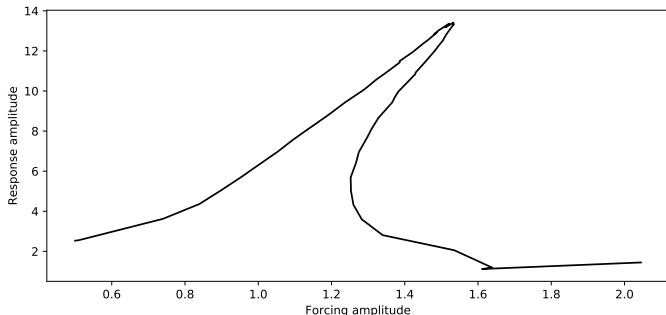


Stepsize 1; 8 interior knots; FDSS 0.5; solution still jumps!

---

---

## Best I could get



‘Simple’ convergence criteria; stepsize 1; 3 interior knots; FDSS 0.2

---

---

## Issues

- ✦ Hyperparameters are very hard to select
  - ▶ Lots of trial and error to get even remotely close to the real results
- ✦ Solution usually jumps near the second fold
  - ▶ Pseudo-arclength condition is met, so the equations are fine
  - ▶ Solution is often not actually a solution!
  - ▶ Either the solver is broken, or the full system is misbehaving

---

## Non-solutions

✿ We're solving for  $F(x_\omega) = 0$

- ▶ Newton iterations: declare convergence when  $\|x_\omega^i - x_\omega^{i-1}\| < tol$
- ▶ Issue: converged  $x_\omega$  typically does not solve  $F(x_\omega) = 0$
- ▶ Alternative: converge when  $\|F(x_\omega)\| < tol$
- ▶ Even for  $tol \in \mathcal{O}(10^{-3})$ , we never converge
- ▶ Solution vector components jump around, rather than converging; unexpected for Newton solvers

✿ Either solver is problematic, or equations are

- ▶ Using a Newton solver; simple code, tried and tested in the Fourier case
- ▶ Finite differences are meaningful:  $\mathcal{O}(0.1)$  perturbations to  $\mathcal{O}(1)$  coefficients
- ▶ If the solver and equations are correct, perhaps the equations are simply unsuitable?

---

## Existence and uniqueness

Does a solution to  $F(x_\omega) = 0$  actually exist?

✿ Continuous case:

- ▶ A natural periodic orbit of the system exists
- ▶ This natural periodic orbit necessarily gives noninvasive control
- ▶ Noninvasive control means  $F(x_\omega) = 0$ , so solutions must exist

✿ Discretised case:

- ▶ We can exactly represent the continuous problem as an infinite-dimensional Fourier problem
- ▶ As the continuous solution exists, so too must the infinite-dimensional discretised problem
- ▶ Due to how the Fourier errors decay, we can be sure that finite-dimensional Fourier discretisation produces a solvable continuation system
- ▶ We don't get this guarantee with splines



---

## Approximate solutions

Does a splines solution exist? When? Thought experiment:

- ✂ Run the system uncontrolled
- ✂ Discretise the output
- ✂ Use the discretised output as a control target

Imperfect discretisation: control target  $\neq$  'natural' oscillations

- ✂ Control becomes invasive
- ✂ Control target is not a solution to the continuation equations
  - ▶ Even though it was obtained from an exact solution, it is not actually a solution; discretisation error stops the natural system behaviour from being a solution

Discretisation error must be negligible for the standard CBC zero problem to become solvable

---

---

## Key result

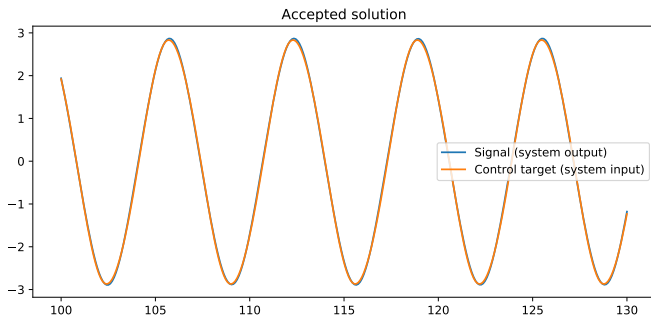
- ✂ If we have no discretisation error, solution exists to continuation equations
- ✂ If we have discretisation error, solution might not exist
- ✂ This explains why Fourier works, splines don't
  - ▶ No discretisation error for infinite Fourier
  - ▶ Can achieve negligible discretisation error for truncated Fourier
  - ▶ Harder to remove spline discretisation error

How accurate are splines?

---

---

## Spline discretisation error

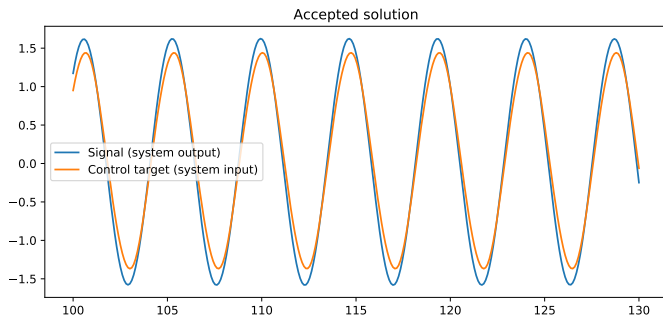


Splines is often very accurate

---

---

## Spline discretisation error



But sometimes not

---

---

## Minimization reformulation

- ✂ A solution is not guaranteed to exist when the spline fit isn't exact
- ✂ We can fix this with new, more general continuation equations
- ✂ Solve for least invasive control target, instead of noninvasive control
  - ▶ Solution will be noninvasive (same solution as for standard continuation equations) when discretisation is exact
  - ▶ Solution is still guaranteed to exist when discretisation is inexact
  - ▶ Solution is noise-robust

---

## Minimization reformulation

✿ Let  $\beta$  be the discretisation

✿ Let  $\text{invasiveness}(\beta) = \int [\text{signal}(t) - \text{target}(\beta, t)]^2 dt$

- ▶ Valid for proportional control
- ▶ Can be easily adapted for other control strategies

✿ Continuation equations:

- ▶  $\frac{\partial \text{invasiveness}}{\partial \beta_i} = 0$
- ▶ predictor  $\perp$  corrector
- ▶ This can be solved using numerical integration and standard Newton iterations; **no need for minimization**: no experimental Hessians needed
- ▶ Alternatively, solve using EGO minimizers; no experimental Jacobians needed

---

## Next steps

- ✂ Write splines without SciPy
- ✂ Try minimizer approach; possibly slower; will guarantee finding an acceptable solution
- ✂ Try adaptive-knots BSplines
  - ▶ In general, optimization-based knot choice will minimize the discretisation error
  - ▶ Duffing is simple enough that adaptive knots shouldn't change the results much
- ✂ Talk to Krasi about approximation and existence of continuation solutions

Also, writing, annual review

---