Surrogate models and novel discretisations

Mark Blyth

Last meeting

- The challenges of working with spiking signals
- Surrogate methods to overcome these

Surrogates

- Given $y_i = f(x_i) + \varepsilon$, estimate $f(x)$
- Splines and Gaussian processes are good methods for estimation

Surrogates

- Given $y_i = f(x_i) + \varepsilon$, estimate $f(x)$
- Splines and Gaussian processes are good methods for estimation
  - Gaussian point estimate: $y \sim \mathcal{N}(f(x), \sigma_\varepsilon^2)$

Surrogates

- Given $y_i = f(x_i) + \varepsilon$, estimate $f(x)$

- Splines and Gaussian processes are good methods for estimation
  - Gaussian point estimate: $y \sim \mathcal{N}(f(x), \sigma_\varepsilon^2)$
  - Gaussian function estimate: $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$

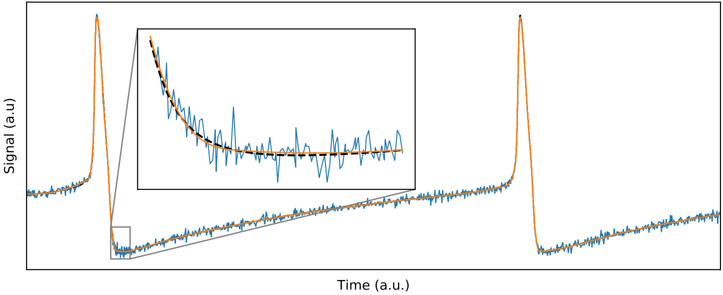
Surrogates

- Given $y_i = f(x_i) + \varepsilon$, estimate $f(x)$

- Splines and Gaussian processes are good methods for estimation
  - Gaussian point estimate: $y \sim \mathcal{N}(f(x), \sigma_\varepsilon^2)$
  - Gaussian function estimate: $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$
  - Splines estimate: $f_i(x) = a_0 + \cdots + a_3 x^3$, for $x \in [\xi_i, \xi_{i+1})$

Developments since last time

- Surrogates tested and working



Developments since last time

- Testing novel discretisations
- Implementing *in-silico* CBC

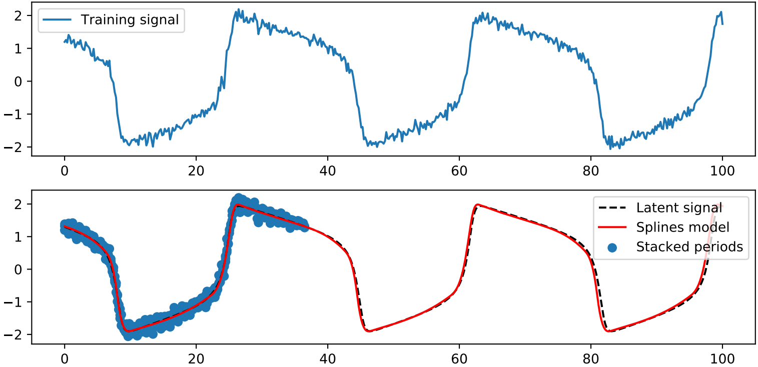
Discretisations

- For $f(x) = \sum \beta_i b_i(x)$, coefficients $\{\beta_i\}$ discretise signal
- Choose basis functions $b_i(x)$ to minimise dimensionality

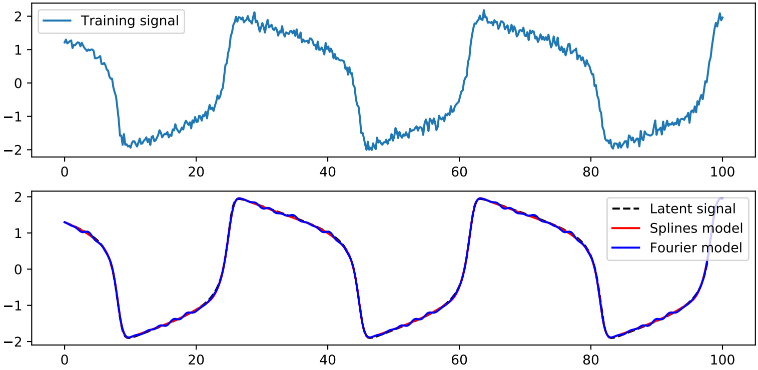
Splines discretisation

- Find a set of basis functions to initial signal $f_0(x)$
  - Find ξ to optimise $\{b_i(x)\}_\xi$ for $f_0(x)$
- Discretisation of $f_i(x)$ given by basis coefficients

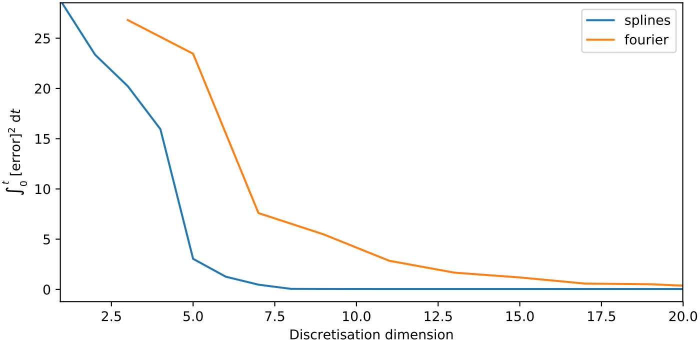
Splines demo



Splines vs Fourier



Goodness-of-fit



Method usage cases

Two approaches to CBC of periodic orbits:

- Harmonically forced:
  - Lump control action with bifurcation parameter; efficiently iterate Fourier harmonics to zero
- Non-harmonically forced:
  - Use Newton iterations to solve for noninvasive control action

In-silico CBC

- Implement an in-silico CBC
- Test it on a variety of toy models
- Use it to demonstrate new discretisation

CBC method

- Use PD control
- Rescale periodics to $t \in [0, 1]$
- Non-adaptive mesh
- Use Newton-iterations to solve for input = output

Discretisors

Discretisors implemented and lightly tested

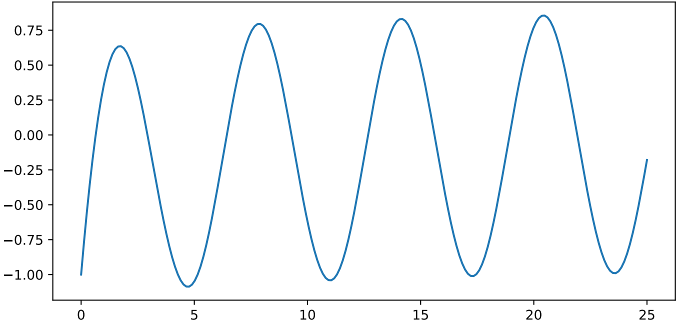
```
discretisation, period = discretisor.discretise(signal)

control_target = discretisor.undiscretise(
    discretisation, period
)
```

Controllers

Controllers implemented and lightly tested

```
controller = Controller(
    "PD", B_matrix, control_target,
    C_matrix=C_matrix, kp=10, kd=10
)
model = Model(
    fitzhugh_nagumo_neuron, ["I"], False, controller
)
solution = model.run_model(
    [0, 25], [-1, -1], I=1, rtol=1e-6
)
```

Control



Continuation

In progress; code written, but not tested

```
next_orbit = prediction_correction_step(
       system, po0, po1, stepsize, discretisor
)
```

Simulation summary

- Control: implemented, lightly tested
- Discretisation: implemented, lightly tested
- Model-continuation interface: unimplemented
- Continuation: implemented, untested
- Results handling: unimplemented

Open questions

- Will splines discretisation work?
- Stationary or adaptive mesh?
- Efficient solving methods?
- Can we interface the code with Simulink?

Next steps

- Finish CBC simulation
- Start conference paper
- Finish continuation review paper