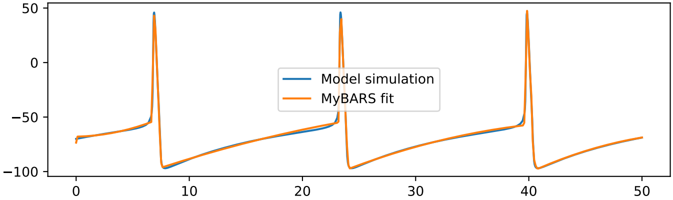
More model fitting

Mark Blyth

Presentation points

- Models for wind tunnel data
- A BARS redesign
- BARS potential pitfalls

An exciting aside



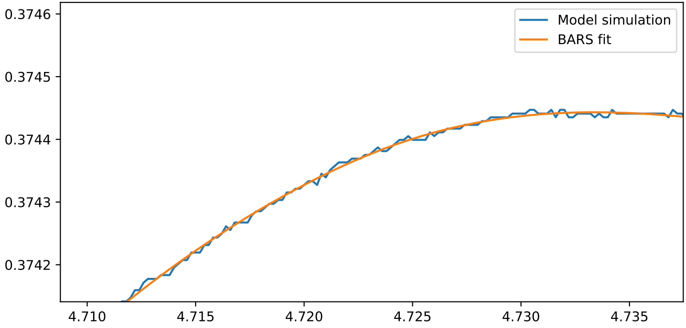
My BARS implementation works!

- Gives okay results *[some sharp corners?]*
- Allows arbitrary interpolation
- Slower than the C implementation

BARS - wind tunnel

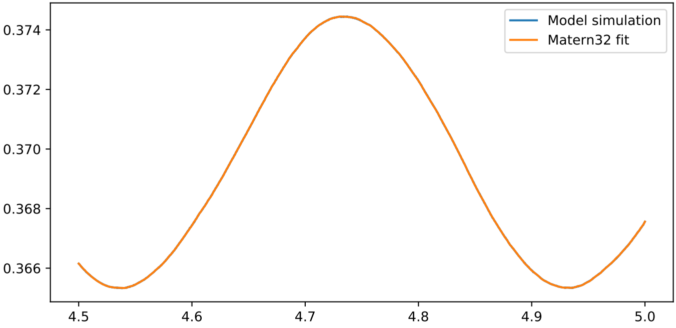
2500 datapoints, 41s run-time

BARS - wind tunnel



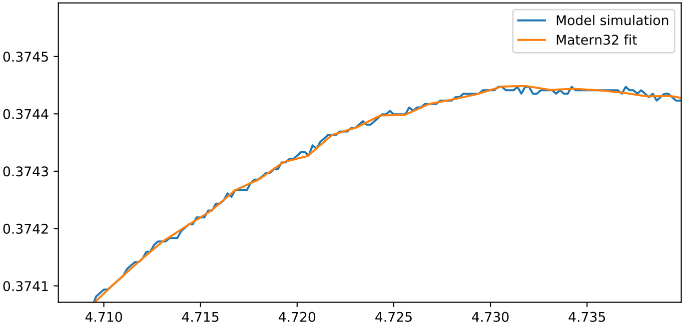
2500 datapoints, 40s run-time

Matern32 (GPR) - wind tunnel



2500 datapoints, 119s run-time

Matern32 (GPR) - wind tunnel

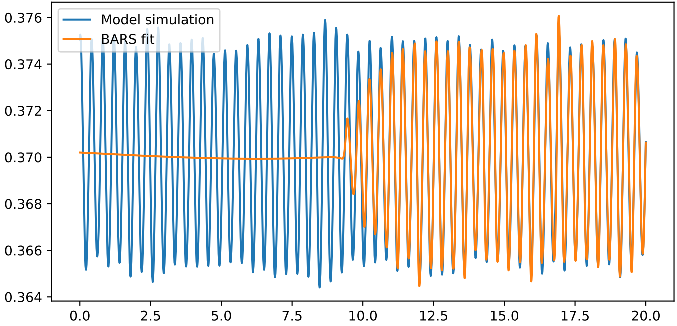


2500 datapoints, 119s run-time

Timings

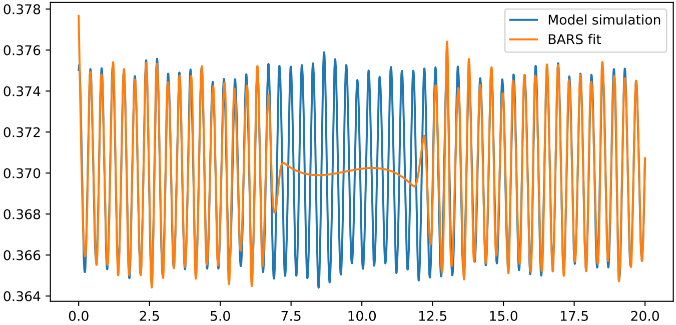
- Unoptimized C code is still faster than optimized python / matlab
  - BARS is using optimized C implementation, GPRs is using my unoptimized python implementation
  - GPFlow, GPyTorch would be much faster for GPR
  - Probabilistic programming might speed up BARS
- Both methods are $\mathcal{O}(n^3)$; require inverting $n \times n$ matrix
  - Both perform poorly on lots of datapoints!

BARS



10,000 datapoints *[crashes with full dataset]*, 142s run-time

BARS



10,000 datapoints *[crashes with full dataset]*, max. number of knots *[80]*, more burn-in; 590s (9 mins 50s) run-time

BARS issues

- C implementation doesn't allow validation
  - Can't interpolate at arbitrary points
  - Python implementation does! But it's slower
- C implementation caps the max. number of knots
  - Python implementation doesn't! But it's slower
  - Doesn't work for long time series
- $\mathcal{O}(n^3)$ means data need downsampling to be used
  - Took about 10 minutes to fit a model to 10% of the data

Experimental data

- If experiments use small numbers of periods (1, 2, 3), current methods work
  - Similar to how I tested models on small numbers of neuron spikes
  - Few periods = less data = fewer knots needed, and quicker fit obtained
- More datapoints mean we need to get creative...
  - Sparse variational BARS
  - Periodic BARS
  - Semiperiodic methods

Or... speed up BARS by not using BARS

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi | y)$

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi | y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi | y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution
- Instead of estimating posterior distribution, why not find posterior mode?

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi|y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution
- Instead of estimating posterior distribution, why not find posterior mode?
  - Use the knot-shuffling steps to evolve an optimal knot-set

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi | y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution
- Instead of estimating posterior distribution, why not find posterior mode?
  - Use the knot-shuffling steps to evolve an optimal knot-set
  - Find $\operatorname{argmax}_{k,\xi} p(y | k, \xi)$ *[MLE]* *or* $\operatorname{argmax}_{k,\xi} p(k, \xi | y)$ *[MAP]*

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi|y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution
- Instead of estimating posterior distribution, why not find posterior mode?
  - Use the knot-shuffling steps to evolve an optimal knot-set
  - Find $\mathrm{argmax}_{k,\xi} p(y|k, \xi)$ *[MLE]* *or* $\mathrm{argmax}_{k,\xi} p(k, \xi|y)$ *[MAP]*
  - Each MCMC knot-shuffle becomes a mutation step

A new BARS approach

IDEA: Bayesian adaptive *evolutionary* splines; genetic algorithm

- Ideal result: find the MAP knot-config *[best given data]* in fewer steps than MCMC takes to estimate posterior distribution
  - BARS uses MCMC to find a posterior knot distribution $p(k, \xi | y)$
  - MCMC uses 10,000+ knot-shuffling steps to estimate this posterior distribution
- Instead of estimating posterior distribution, why not find posterior mode?
  - Use the knot-shuffling steps to evolve an optimal knot-set
  - Find $\operatorname{argmax}_{k,\xi} p(y|k, \xi)$ *[MLE]* *or* $\operatorname{argmax}_{k,\xi} p(k, \xi|y)$ *[MAP]*
  - Each MCMC knot-shuffle becomes a mutation step
  - Each MCMC acceptance/rejection becomes an evolutionary scoring step

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step…

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   - 2.1 Randomly move, add, or delete a knot from each set
   - 2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch...

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch...
   3.1 Take i perturbation steps on each model in the pool

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch...
   3.1 Take i perturbation steps on each model in the pool
   3.2 Store a pool of the j best models

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch...
   3.1 Take i perturbation steps on each model in the pool
   3.2 Store a pool of the j best models
   3.3 Create a new pool by randomly recombining existing pool

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step...
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch...
   3.1 Take i perturbation steps on each model in the pool
   3.2 Store a pool of the j best models
   3.3 Create a new pool by randomly recombining existing pool
4. Pool will *[hopefully]* converge to MLE

Possible evolutionary splines implementation

1. Start with a set of initial (randomly drawn) pool of knot configurations
2. For each perturbation step…
   2.1 Randomly move, add, or delete a knot from each set
   2.2 If the perturbed knot has a better likelihood $p(y|k, \xi)$ than the original, replace the original with it
3. For each epoch…
   3.1 Take i perturbation steps on each model in the pool
   3.2 Store a pool of the j best models
   3.3 Create a new pool by randomly recombining existing pool
4. Pool will *[hopefully]* converge to MLE
5. Can maximise posterior, instead of likelihood, by including a prior term

BARS vs evolution

- Would be interesting to compare this to the MCMC method
  - MCMC sets up a Markov chain whose stationary distribution is the posterior
  - This aims to find a Markov chain whose stationary distribution is the argmax
  - Grounds for a rigorous justification / proof of convergence
- Evolution will likely be faster
  - Could leverage existing genetic optimization packages
  - Easily parallelised for more speed-up
  - No RJ-MCMC makes it easier target for probabilistic programming

SV-BARS

Another idea: remodel BARS to work similarly to sparse GPR

- BARS typically uses 3000+ MCMC steps
  - Each MCMC step requires inverting an $n \times n$ matrix – SLOW
- Choose a *[small]* set of maximally informative surrogate datapoints (x_i^*, y_i^*)
- Run MCMC step on surrogate datapoints
  - Much faster to invert the smaller matrix
- Well-chosen surrogate points means we get the same result as running on real data
- Fewer datapoints means it runs a lot faster

SV-BARS implementation

- Find a set of m inducing points (x_i^*, y_i^*)
  - Find inducing points by minimising $D_{\mathrm{KL}}\left[p_y \| p_{y^*}\right]$
  - Use variational Bayes to approximate this
- Find posterior knots $p(k, \xi | y^*)$ *[instead of $p(k, \xi | y)$]*
  - $\mathcal{O}(m^3), m \ll n$
  - Sparse GPR is $\mathcal{O}(nm^3)$, so if my complexity is correct, we get a bigger speed-up / outperform SVGPR!

Would require learning RJ-MCMC, variational Bayes, sparse GPR in-depth

Periodic BARS

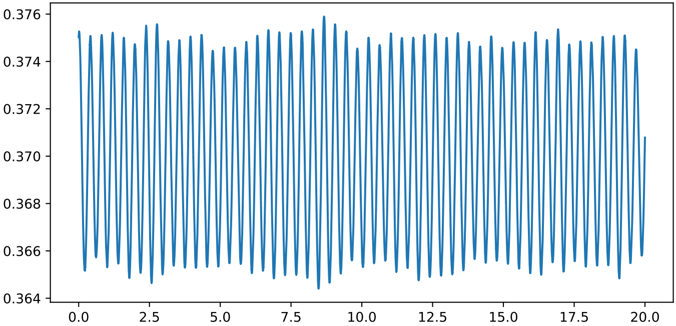
An approach for if we need to model more than a few spikes:

- Assume data are given by $y_i = f(x_i) + \varepsilon$
  - $f(t) = f(t + T)$
- Find either
  - T-periodic knot-set
  - T-periodic basis splines
    - Nicer approach

…in such a way that we…

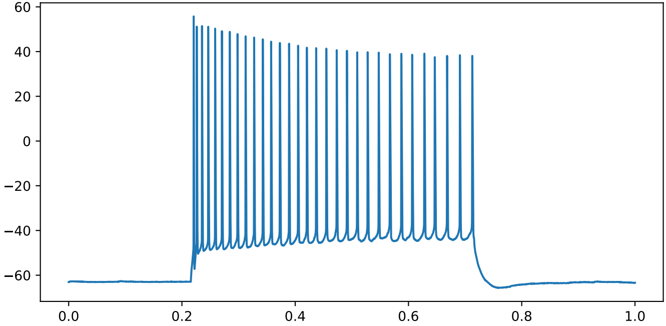
- minimise fitting time
- balance fit against number of knots

Semiperiodic methods



Experimental data aren't perfectly periodic

Semiperiodic methods



Experimental data aren't perfectly periodic *[could Ca^{2+}, or experiment setup!]*

Semiperiodic methods

- Assume data are given by $y_i = A(x_i) f(x_i) + \varepsilon$

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude

Semiperiodic methods

- Assume data are given by $y_i = A(x_i) f(x_i) + \varepsilon$
  - $f(t) = f(t + T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$
  - Only requires one period's worth of data

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$
  - Only requires one period's worth of data
- Fit a model to $A(t)$

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$
  - Only requires one period's worth of data
- Fit a model to $A(t)$
  - Requries a few datapoints across all periods

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$
  - Only requires one period's worth of data
- Fit a model to $A(t)$
  - Requries a few datapoints across all periods
- BARS, GPR, NARMAX all interesting model options

Semiperiodic methods

- Assume data are given by $y_i = A(x_i)f(x_i) + \varepsilon$
  - $f(t) = f(t+T)$ is the periodic behaviour
  - $A(t)$ is the drifting amplitude
  - Might require transforming the data, to a zero-DC offset
- Fit a model to $f(t)$
  - Only requires one period's worth of data
- Fit a model to $A(t)$
  - Requries a few datapoints across all periods
- BARS, GPR, NARMAX all interesting model options
- Would likely give similar results to sparse BARS

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC
- For bigger data, we need something more creative

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC
- For bigger data, we need something more creative
- Sparse variational BARS would be valuable within machine learning community

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC
- For bigger data, we need something more creative
- Sparse variational BARS would be valuable within machine learning community
  - Would very likely give state-of-the-art results!

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC
- For bigger data, we need something more creative
- Sparse variational BARS would be valuable within machine learning community
  - Would very likely give state-of-the-art results!
  - Could be combined with the evolutionary approach for more speedup

Conclusions

- For small numbers of cycles (1, 2, 3), current BARS works well enough for CBC
- For bigger data, we need something more creative
- Sparse variational BARS would be valuable within machine learning community
  - Would very likely give state-of-the-art results!
  - Could be combined with the evolutionary approach for more speedup
- (Semi)periodic BARS would be less generally applicable, but potentially faster when applicable

Possible routes

1. Use current BARS setup for a CBC experiment

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community
3. Demonstrate splines *[and GPR?]* on other problems (NDC, comp-synth-bio, ML)

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community
3. Demonstrate splines *[and GPR?]* on other problems (NDC, comp-synth-bio, ML)
   - Extra paper, builds well on the surrogate-modelling knowledge I'm developing

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community
3. Demonstrate splines *[and GPR?]* on other problems (NDC, comp-synth-bio, ML)
   - Extra paper, builds well on the surrogate-modelling knowledge I'm developing
   - I wouldn't know which problems to apply them to

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community
3. Demonstrate splines *[and GPR?]* on other problems (NDC, comp-synth-bio, ML)
   - Extra paper, builds well on the surrogate-modelling knowledge I'm developing
   - I wouldn't know which problems to apply them to
4. Make a periodic BARS setup, then use that in CBC

Possible routes

1. Use current BARS setup for a CBC experiment
   - Gets CBC results fast
2. Adapt current BARS setup for sparsity / evolution, then use in CBC
   - New splines method would be valuable in ML community
3. Demonstrate splines *[and GPR?]* on other problems (NDC, comp-synth-bio, ML)
   - Extra paper, builds well on the surrogate-modelling knowledge I'm developing
   - I wouldn't know which problems to apply them to
4. Make a periodic BARS setup, then use that in CBC
   - Periodic BARS is a less general method

My proposal

- Validate models
  - Also try simple data transformations for GPR
- Write up notes on *everything* so far

Then...

1. Adapt BARS for sparsity, evolution
   - Fast, SOTA, scalable
   - Needs variational Bayes learned
2. Demonstrate sparse BARS on other problems (NDC, comp-synth-bio, ML)
   - Not sure which problems would benefit from splines?
3. Apply the shiny new splines method to CBC