The Bayesian framework for calibration and UQ with application to a porous media flow model

Damon McDougall

Centre for Predictive Engineering and Computational Sciences
Institute for Computational Engineering and Sciences
The University of Texas at Austin

16th of July, 2015

Collaborators: R. D. Moser, T. A. Oliver, T. Portone

Motivation

Understand physical phenomena

Observations of phenomena

Mathematical model of phenomena (includes some parameters that characterise behaviour)

Numerical model approximating mathematical model

Find parameters in a situation of interest (calibration)

Use the parameters to do something cool

General framework

Model (usually a PDE): $\mathcal{G}(u, \theta)$ where u is the initial condition and θ are model paramaters.

u: perhaps an initial condition

θ: perhaps some interesting model parameters (diffusion, convection speed, permeability field, material properties)

Observations:

$$
\begin{aligned}
& y_{j,k} = u(x_j, t_k) + \eta_{j,k}, \quad \eta_{j,k} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2) \\
\rightsquigarrow \quad & y = \mathcal{G}(\theta) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2 I)
\end{aligned}
$$

Want:

$$p(\theta | y) \propto p(y | \theta) p(\theta)$$

Why?

Do we need Bayes's theorem?

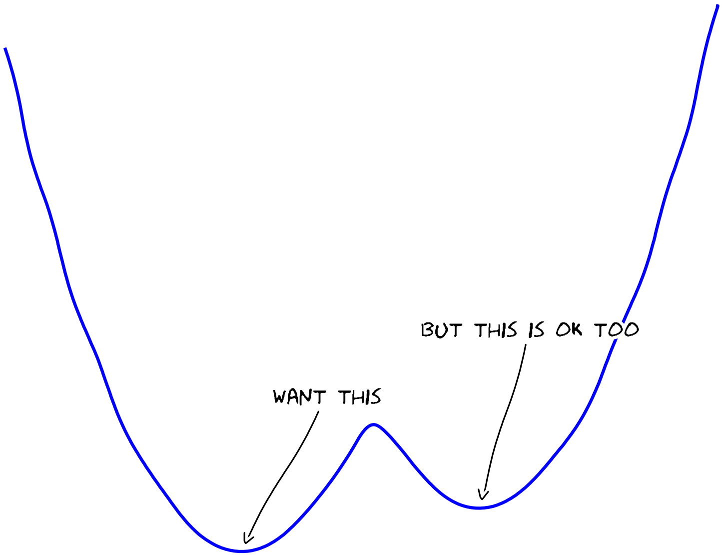
Is Bayes's theorem really necessary? We could minimise

$$J(\theta) = \frac{1}{2\sigma^2}\|\mathcal{G}(\theta) - y\|^2 + \frac{1}{2\lambda^2}\|\theta\|^2$$

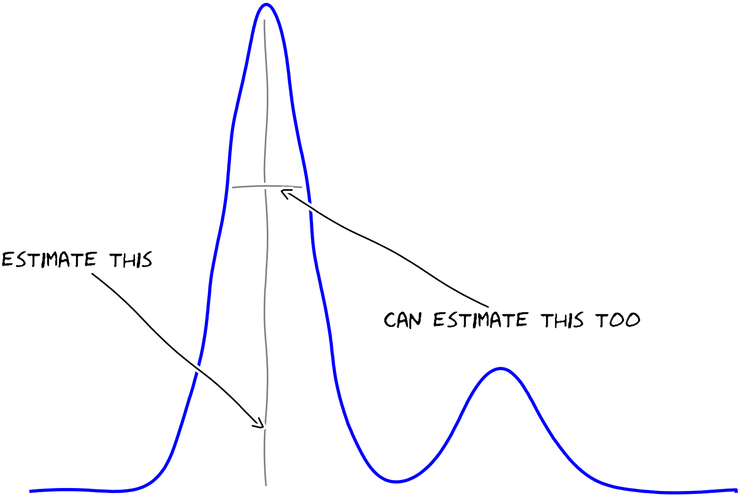
to get

$$\theta^* = \text{argmin}_\theta \, J(\theta)$$

Do we need Bayes's theorem?



Do we need Bayes's theorem?



Do we need Bayes's theorem?

Bayesian methods involve estimating *uncertainty* (as well as mean).
They're equivalent.

Deterministic optimisation:

$$J(\theta) = \underbrace{\frac{1}{2\sigma^2}\|\mathcal{G}(\theta) - y\|^2}_{\textit{misfit}} + \underbrace{\frac{1}{2\lambda^2}\|\theta\|^2}_{\textit{regularisation}}$$

Bayesian framework:

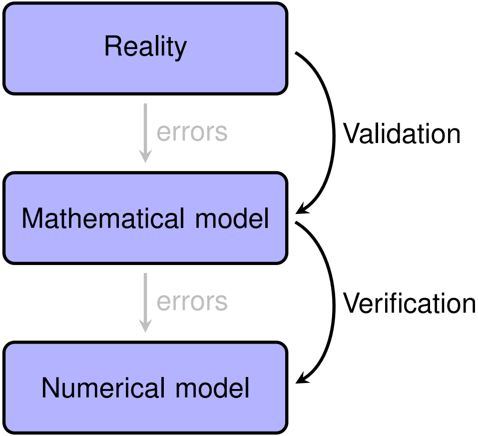
$$\begin{aligned}\exp(-J(\theta)) &= \underbrace{\exp\left(-\frac{1}{2\sigma^2}\|\mathcal{G}(\theta) - y\|^2\right)}_{\textit{likelihood}} \underbrace{\exp\left(-\frac{1}{2\lambda^2}\|\theta\|^2\right)}_{\textit{prior}} \\ &= p(y|\theta)p(\theta) \\ &\propto p(\theta|y)\end{aligned}$$

Method for solving Bayesian inverse problems

- Kalman filtering/smoothing methods
  - Kalman filter (Kalman)
  - Ensemble Kalman filter (Evensen)
- Variational methods
  - 3D VAR (Lorenc)
  - 4D VAR (Courtier, Talagrand, Lawless)
- Particle methods
  - Particle filter (Doucet)
- Sampling methods
  - Markov chain Monte Carlo (Metropolis, Hastings)

This list is not exhaustive. The body of work is prodigious.

Understanding errors



"I reject your reality and substitute my own!"

— Adam Savage (Mythbusters)

Porous media flow application

We will conjur our very own reality. Model is

$$\begin{aligned} \frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} &= \nu \frac{\partial^\alpha c}{\partial x^\alpha}, \quad t > 0, \quad x \in (0,1) \\ c(x,0) &= c_0(x), \\ c(0,t) &= c(1,t), \quad t > 0 \end{aligned}$$

and u is a known fluid velocity independent of x. Observations are

$$\begin{aligned} & y_{jk} = c(x_j, t_k) + \eta_{jk}, \quad \eta_{jk} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2) \\ \rightsquigarrow \quad & y = (y_{11}, \dots, y_{JK})^\top \in \mathbb{R}^{JK} \end{aligned}$$

We get to play God with our reality.

Porous media flow application

- Write down the mathematical model for what we think is happening.
- Know it's some kind of confusion equation, but there might be errors.
- Perhaps, since medium is porous, the flux is weird?

$$\begin{aligned} \frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} &= \bar{\nu}\frac{\partial}{\partial x}\left(\frac{\partial c}{\partial x} + \mathcal{L}(c)\right), \\ &=: \mathcal{A}c \end{aligned}$$

Same initial and boundary conditions as reality.

What does $\mathcal{L}$ look like given the observations y? Bayes to the rescue,

$$p(r_k, \theta_k | y) \propto p(y | r_k, \theta_k) p(r_k, \theta_k),$$

where,

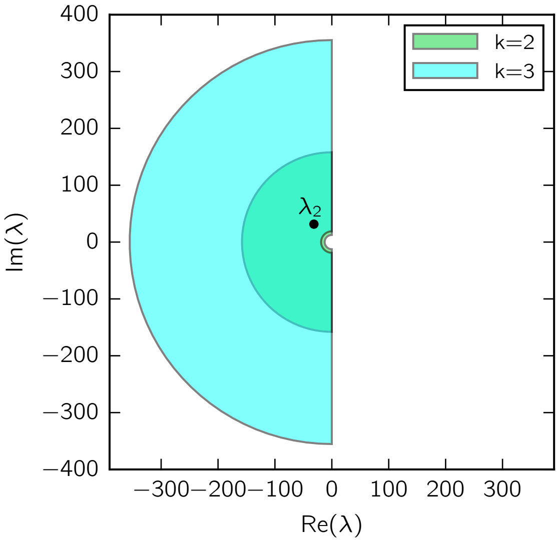
$$\begin{aligned} \lambda_k &= r_k \exp(i\theta_k), \\ \mathcal{A}\phi_k(x) &= \lambda_k \phi_k(x). \end{aligned}$$

The prior: $p(r_k, \theta_k)$

Assumptions:

1. r_j and r_k are independent for $j \neq k$;
2. θ_j and θ_k are independent for $j \neq k$;
3. r_j and θ_k are independent $\forall j, k$;
4. $r_k \sim U[2\pi k, (2\pi k)^2]$;
5. $\theta_k \sim U[-\frac{\pi}{2}, \frac{\pi}{2}]$.

The prior $p(r_k, \theta_k)$

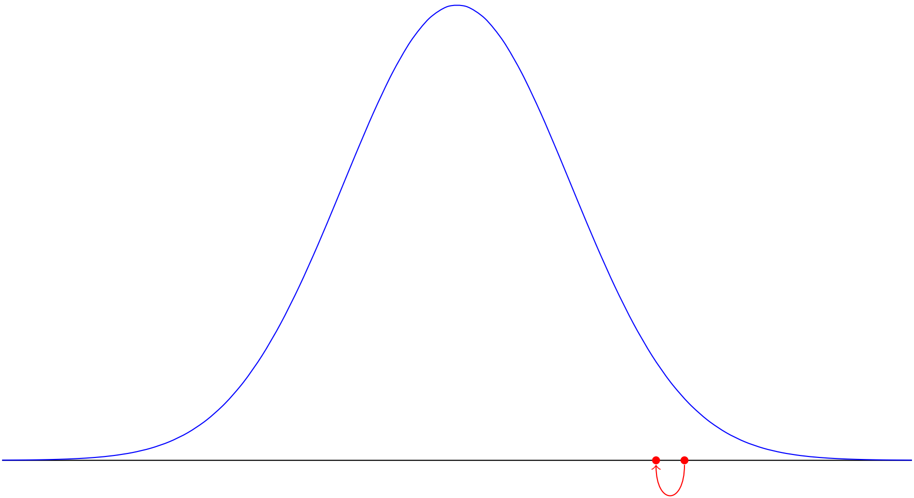


QUESO

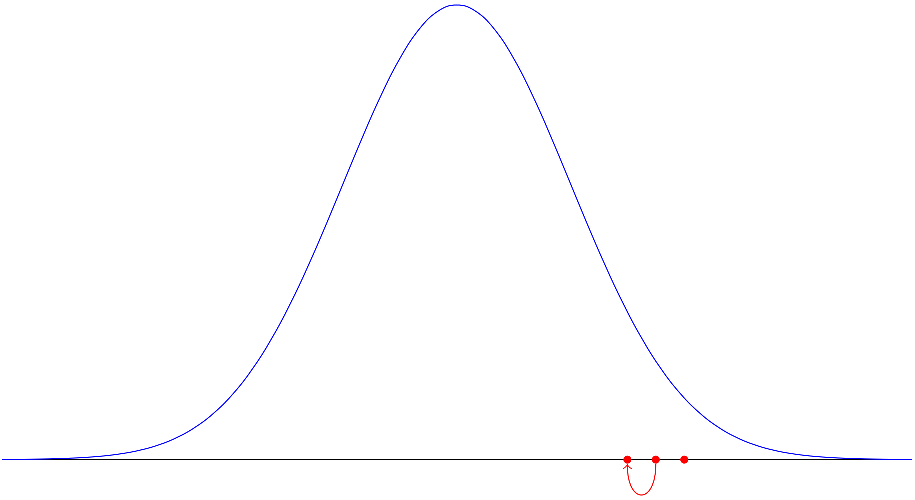
Nutshell: QUESO gives samples from $p(\theta|y)$ (called MCMC)

- Library for Quantifying Uncertainty in Estimation, Simulation and Optimisation
- Born in 2008 as part of PECOS PSAAP programme
- Provides robust and scalable sampling algorithms for UQ in computational models
- Open source
- C++
- MPI for communication
- Parallel chains, each chain can house several processes
- Dependencies are MPI, Boost and GSL. Other optional features exist
- http://libqueso.com

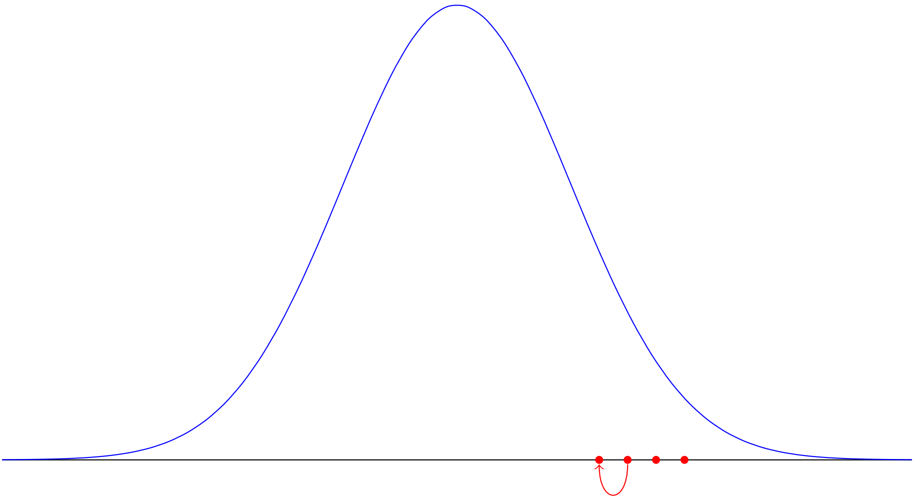
What does MCMC look like?



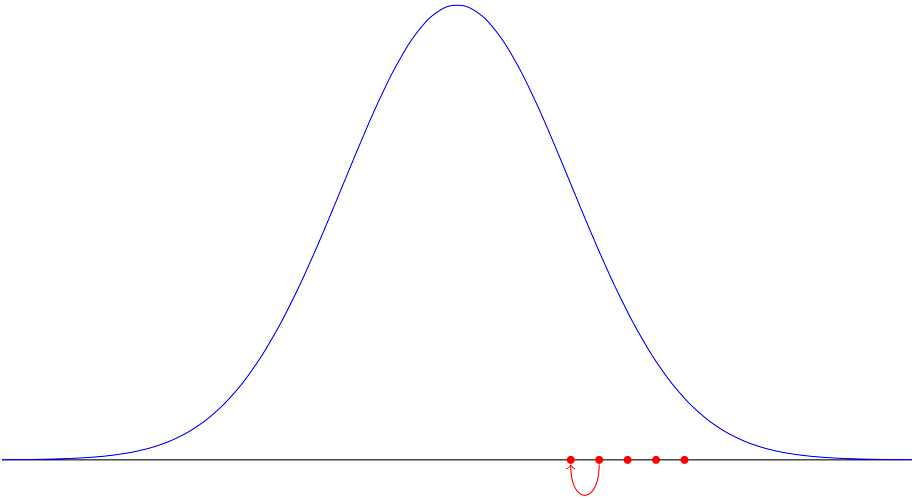
What does MCMC look like?



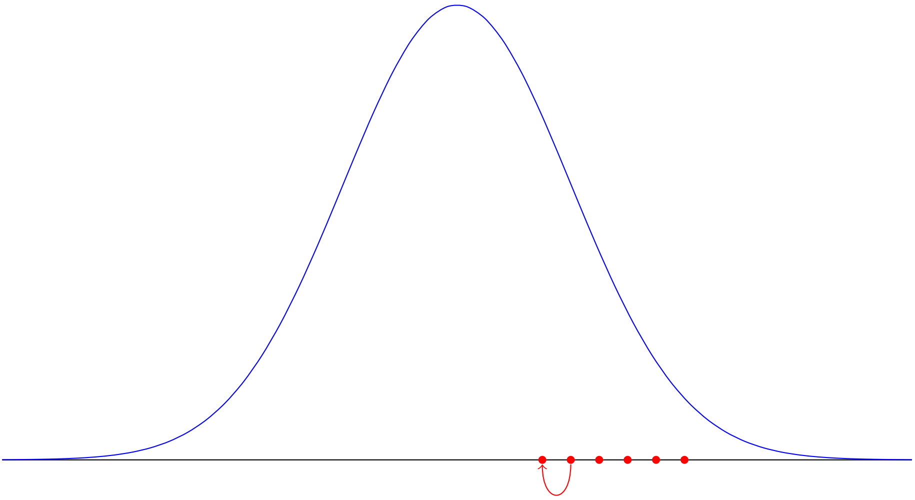
What does MCMC look like?



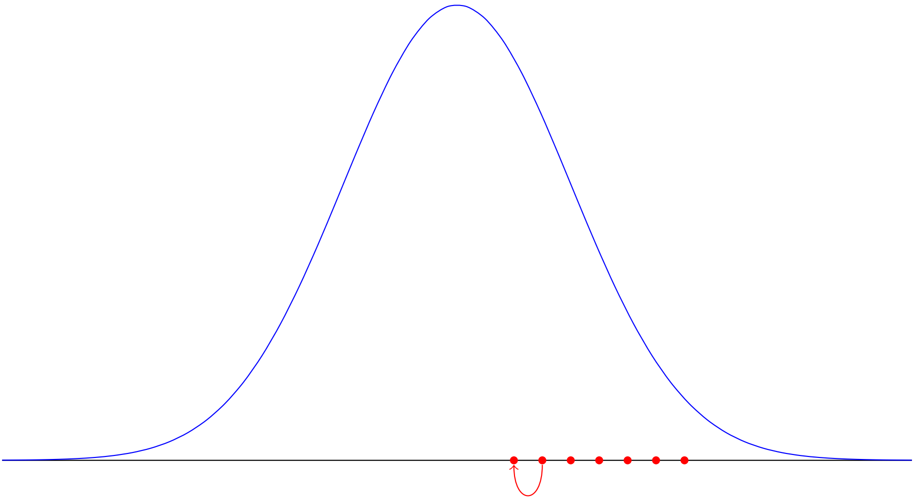
What does MCMC look like?



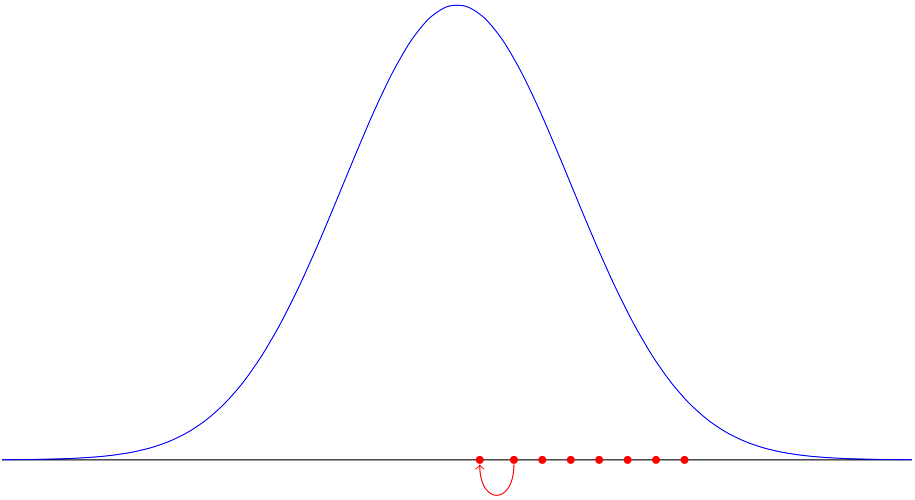
What does MCMC look like?



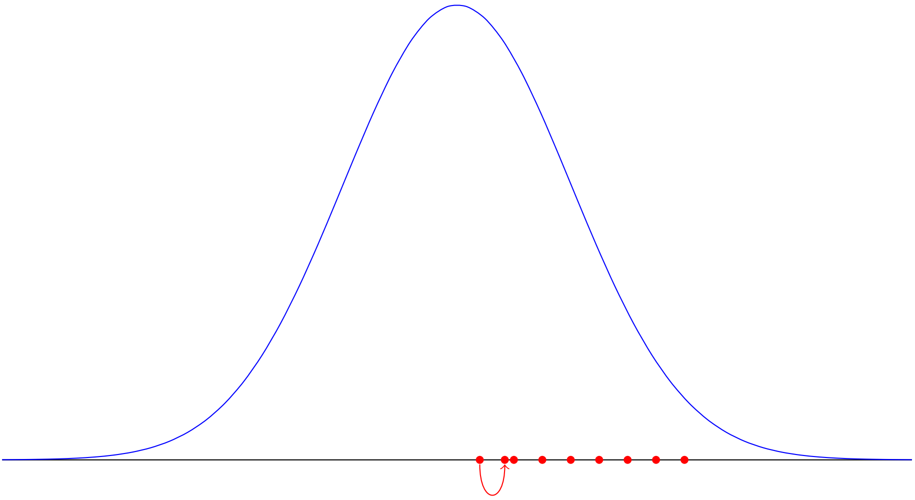
What does MCMC look like?



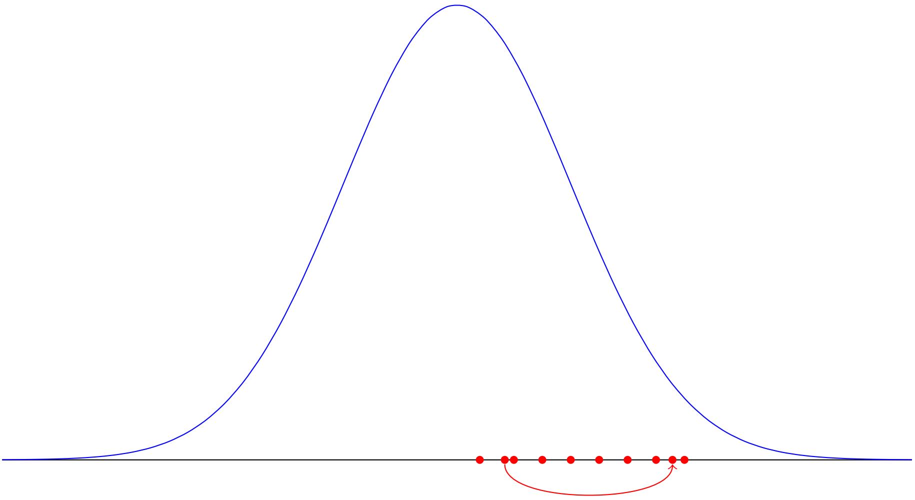
What does MCMC look like?



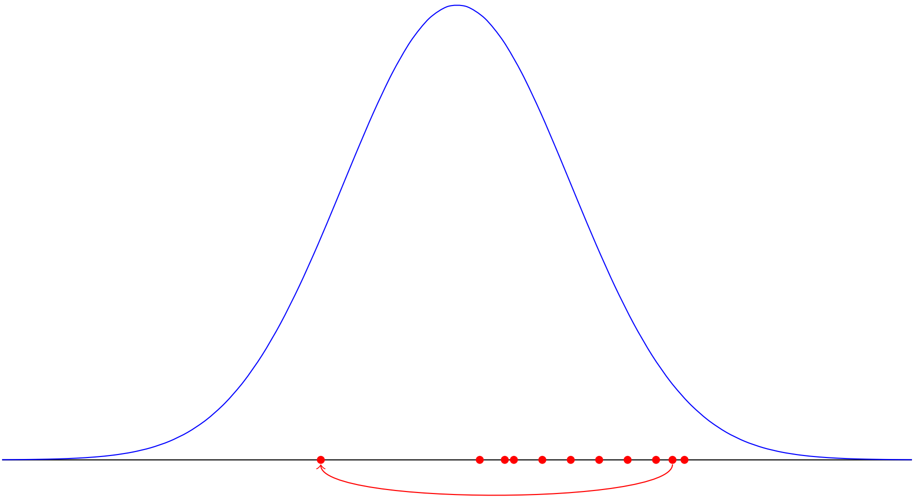
What does MCMC look like?



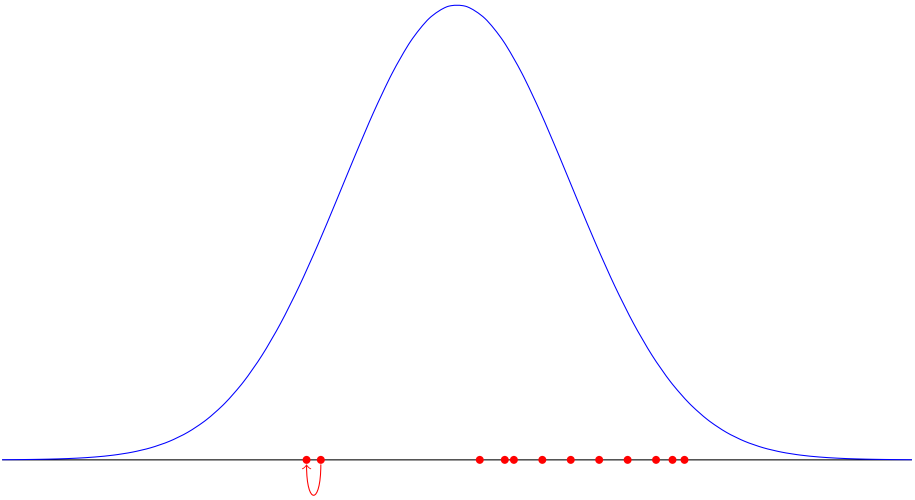
What does MCMC look like?



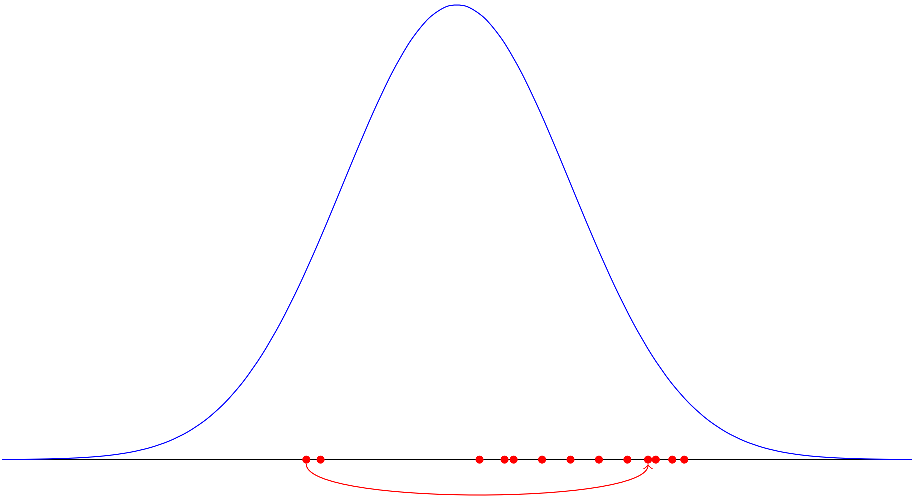
What does MCMC look like?



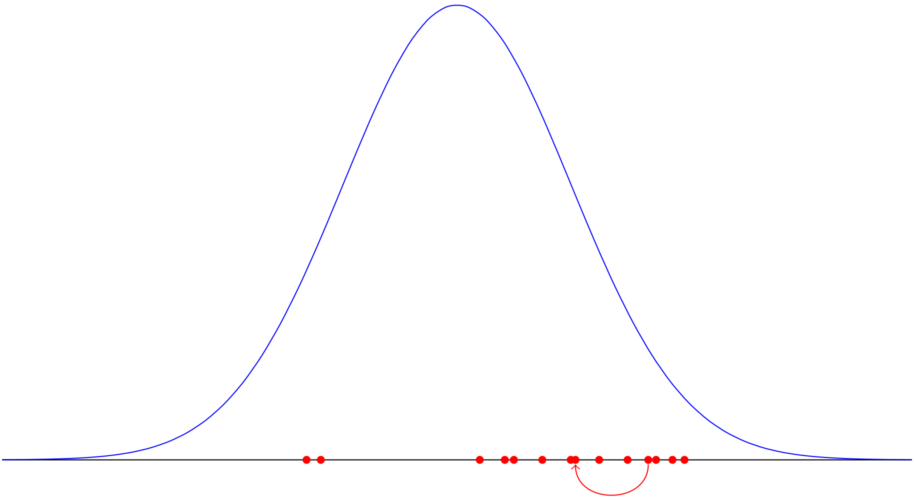
What does MCMC look like?



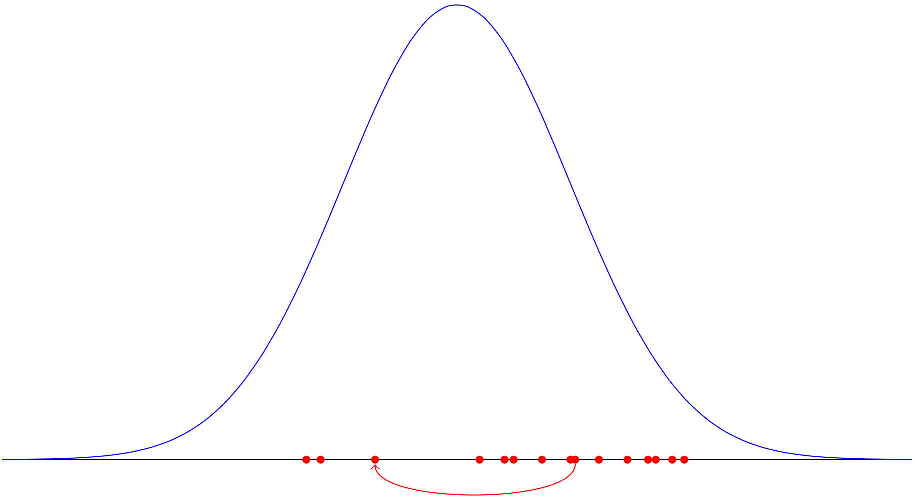
What does MCMC look like?



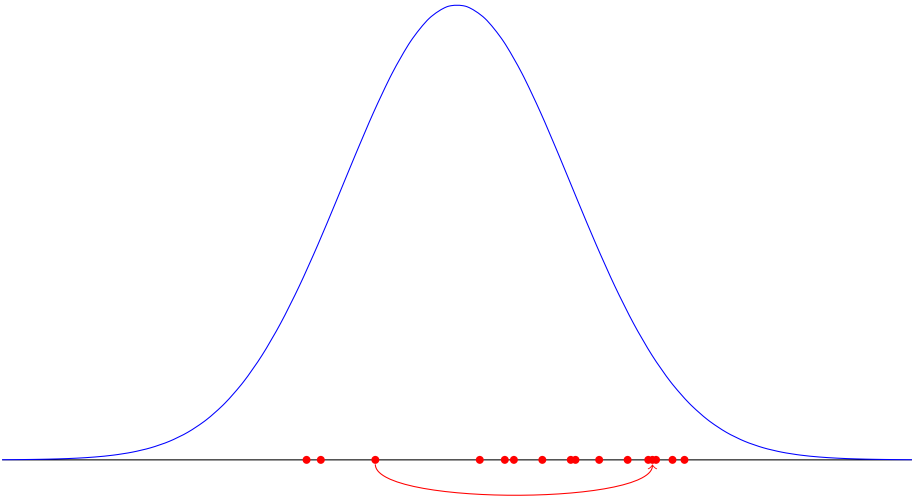
What does MCMC look like?



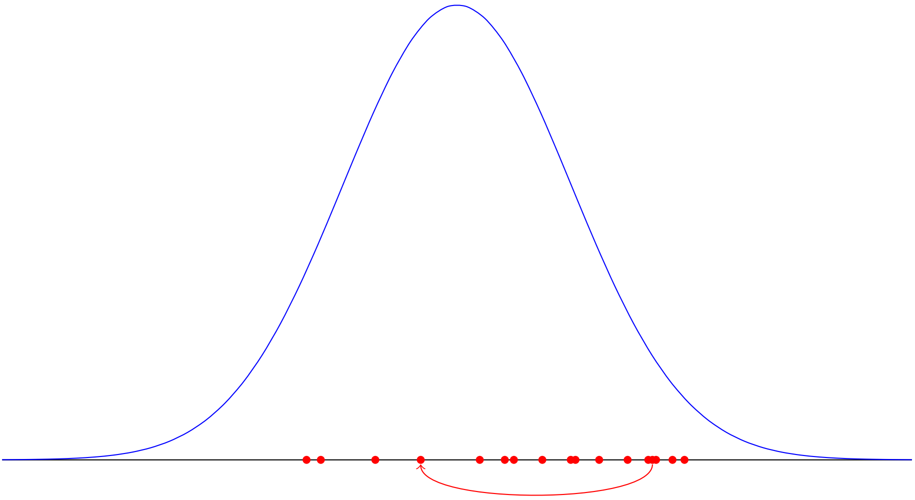
What does MCMC look like?



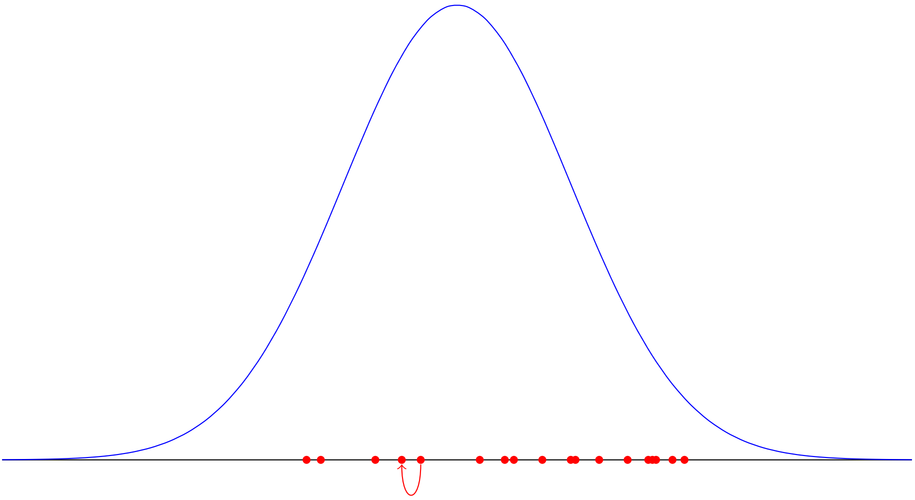
What does MCMC look like?



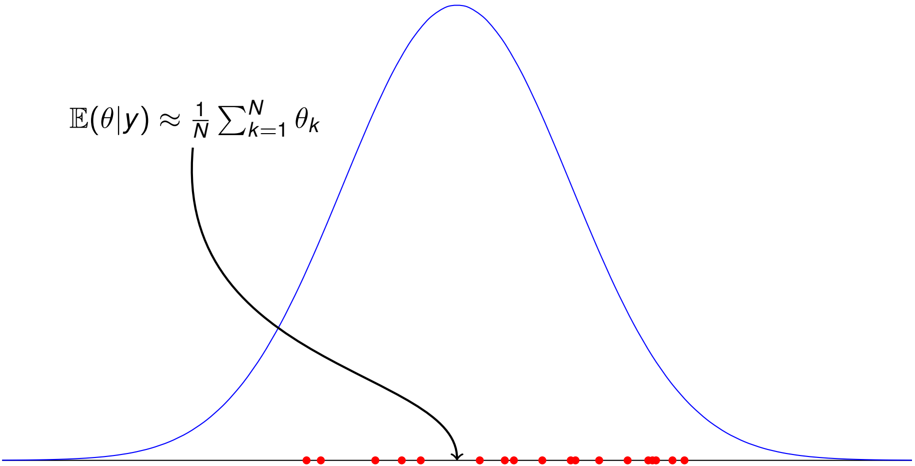
What does MCMC look like?



What does MCMC look like?



What does MCMC look like?



How to do MCMC? Sampling $p(\theta|y)$

- Idea: Construct $\{\theta_k\}_{k=1}^{\infty}$ cleverly such that $\{\theta_k\}_{k=1}^{\infty} \sim p(\theta|y)$
  1. Let θ_j be the 'current' state in the sequence and construct a *proposal*, $z \sim q(\theta_j, \cdot)$

How to do MCMC? Sampling $p(\theta|y)$

- Idea: Construct $\{\theta_k\}_{k=1}^{\infty}$ cleverly such that $\{\theta_k\}_{k=1}^{\infty} \sim p(\theta|y)$
  1. Let θ_j be the 'current' state in the sequence and construct a *proposal*, $z \sim q(\theta_j, \cdot)$
  2. Compute $\alpha(\theta_j, z) = 1 \wedge \frac{p(z|y)q(z,\theta_j)}{p(\theta_j|y)q(\theta_j,z)}$

How to do MCMC? Sampling $p(\theta|y)$

- Idea: Construct $\{\theta_k\}_{k=1}^{\infty}$ cleverly such that $\{\theta_k\}_{k=1}^{\infty} \sim p(\theta|y)$

1. Let θ_j be the 'current' state in the sequence and construct a *proposal*, $z \sim q(\theta_j, \cdot)$
2. Compute $\alpha(\theta_j, z) = 1 \wedge \frac{p(z|y)q(z,\theta_j)}{p(\theta_j|y)q(\theta_j,z)}$
3. Let

$$\theta_{j+1} = \begin{cases} \theta & \text{with probability } \alpha(\theta_j, z) \\ \theta_j & \text{with probability } 1 - \alpha(\theta_j, z) \end{cases}$$

How to do MCMC? Sampling $p(\theta|y)$

- Idea: Construct $\{\theta_k\}_{k=1}^{\infty}$ cleverly such that $\{\theta_k\}_{k=1}^{\infty} \sim p(\theta|y)$
  1. Let θ_j be the 'current' state in the sequence and construct a *proposal*, $z \sim q(\theta_j, \cdot)$
  2. Compute $\alpha(\theta_j, z) = 1 \wedge \frac{p(z|y)q(z,\theta_j)}{p(\theta_j|y)q(\theta_j,z)}$
  3. Let

$$\theta_{j+1} = \begin{cases} \theta & \text{with probability } \alpha(\theta_j, z) \\ \theta_j & \text{with probability } 1 - \alpha(\theta_j, z) \end{cases}$$

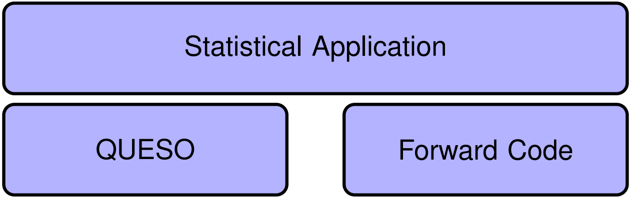
- We can take θ_1 to be a draw from $p(\theta)$

Why use QUESO?

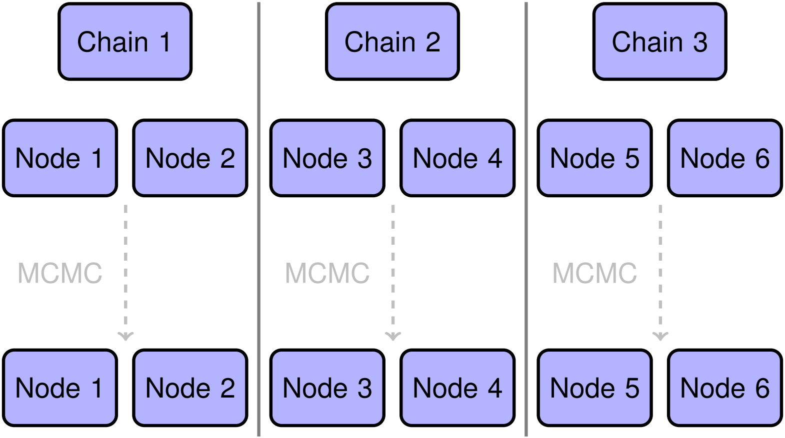
Other solutions are available, e.g. R, PyMC, emcee, MICA, Stan, MUQ.

QUESO solves the same problem, but:

- Has been designed to be used with large forward problems
- Has been used successfully with 5000+ cores
- Leverages parallel MCMC algorithms
- Supports for finite **and** infinite dimensional problems



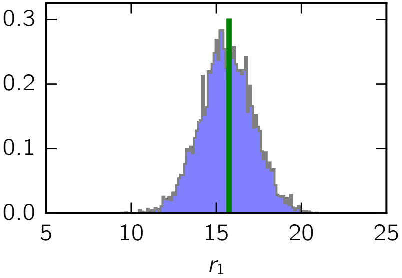
Why use QUESO?

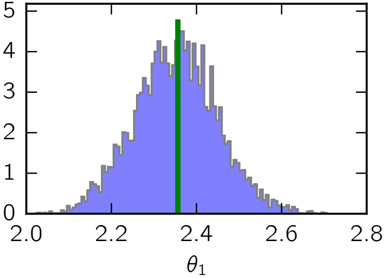


Results

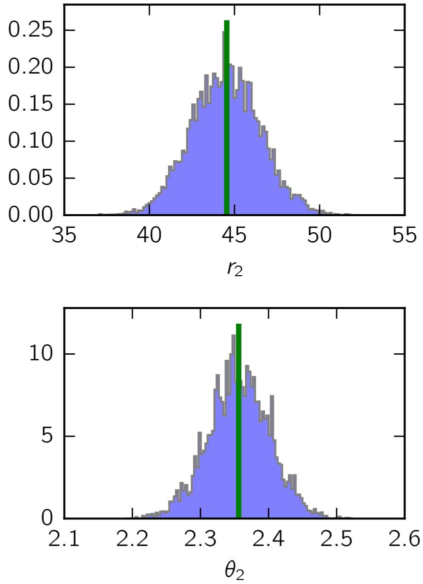
- $c_0(x) = A \exp\left(-\frac{1}{2\tau^2} \|x - 0.5\|\right)$
- $\alpha = 1.5$
- $\sigma = 0.01$
- $J = 8, K = 1 \Rightarrow y \in \mathbb{R}^8$
- `QUESO` DRAM algorithm
  - http://libqueso.com

Results: $p(r_1, \theta_1 | y)$

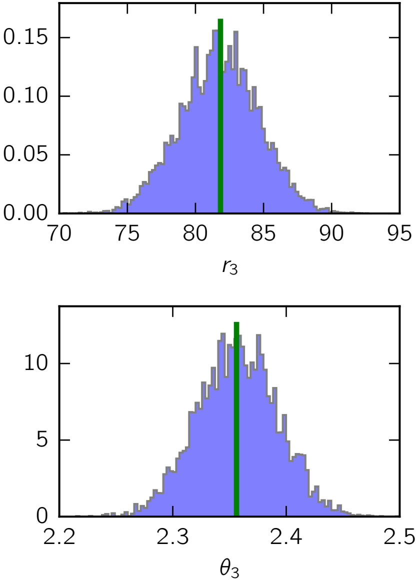




Results: $p(r_2, \theta_2 | y)$



Results: $p(r_3, \theta_3 | y)$



Summary

- Shown how to solve a Bayesian calibration problem.
- Shown what the solution to this calibration problem looks like.
- Described the (open source) software that we use.
- Can infer a differential operator. This is new.
- Framework arises from not knowing the truth.
- Future work:
  - Application to a problem where the inadequacy (operator) does not capture the true inadequacy
  - May still involve operator inference, but the operator is random
  - Are there opportunities to advance the algorithmic component for these types of problems?
  - Will these algorithms be HPC-friendly?

- We're hiring!
  - Post-doctoral research fellows
  - `http://pecos.ices.utexas.edu/job-postings/`