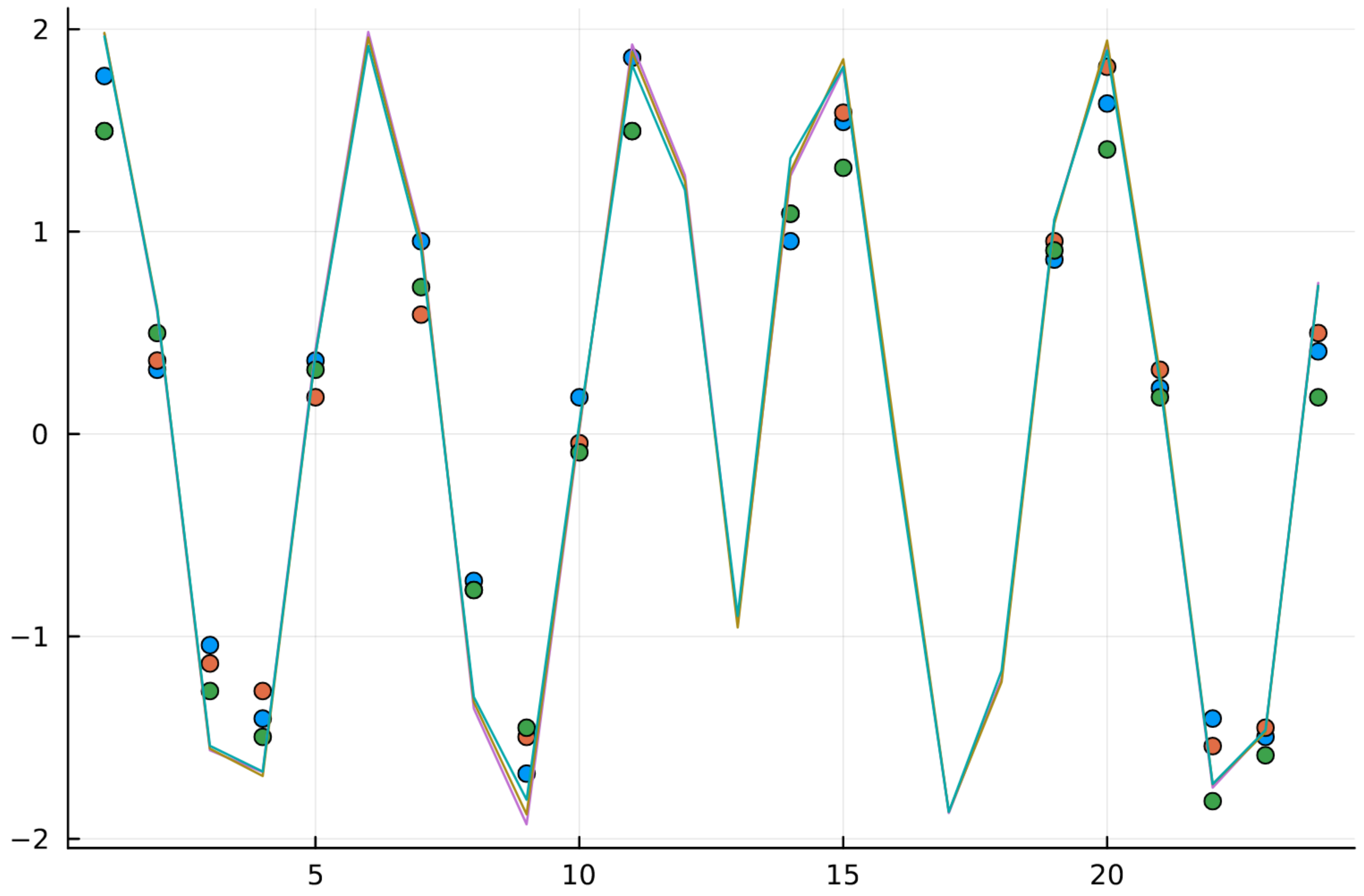
Uncertainty quantification and stochastic optimal control: Applications to booster beam steering

Nathan Urban
nurban@bnl.gov
Applied Mathematics, Computing & Data Sciences
Brookhaven National Laboratory

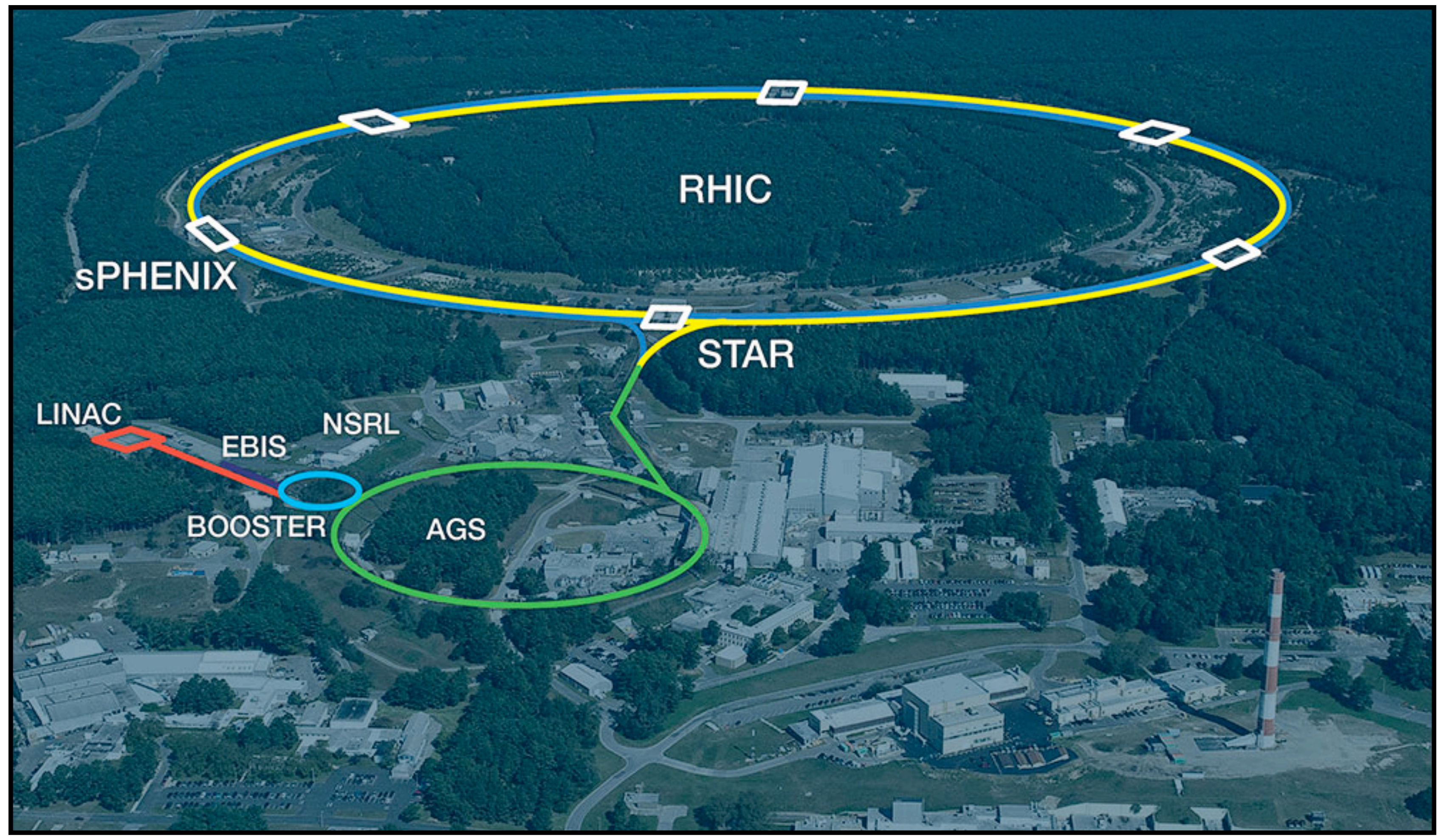
Accelerator control

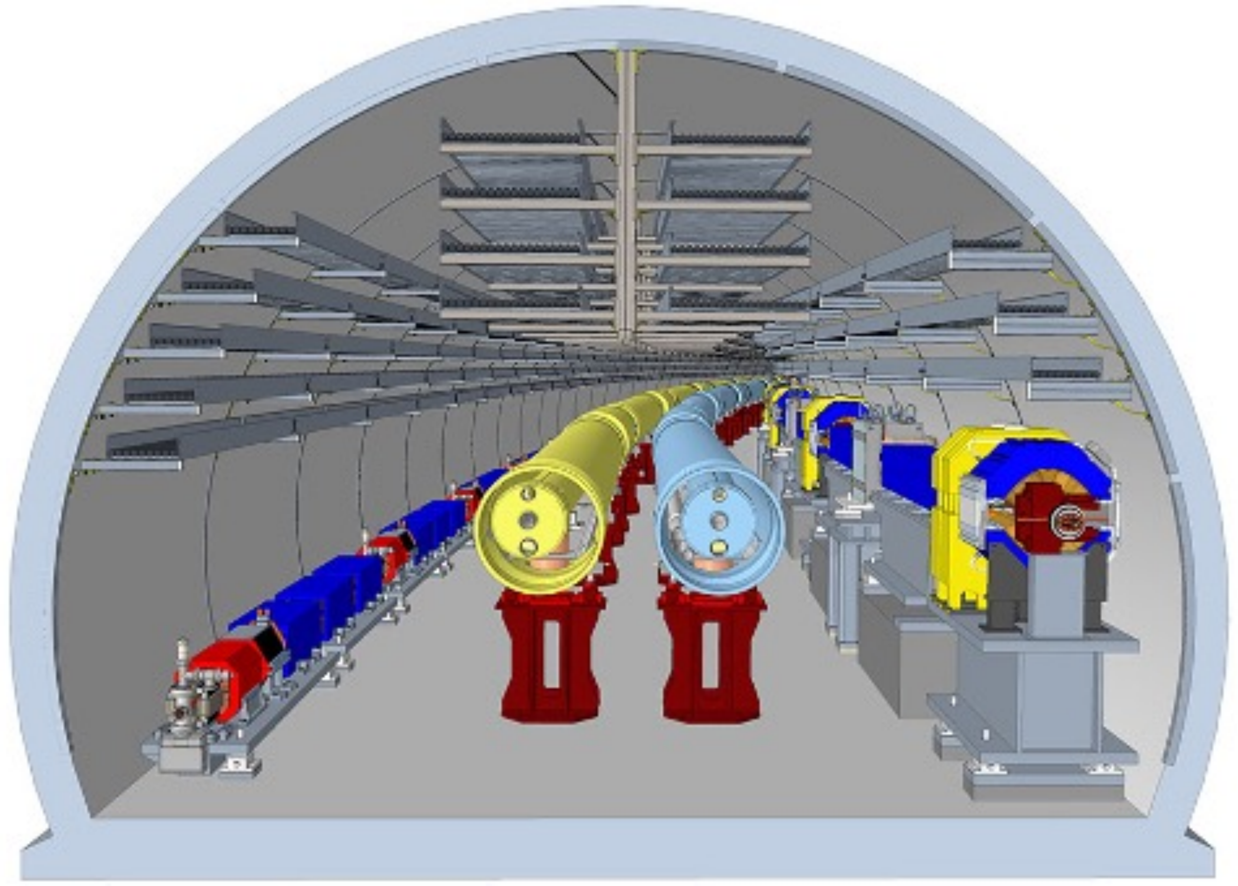
- For this talk: use Bmad model to predict beam position in response to operator inputs
  - Can control other quantities (polarization, emittance, luminosity, "figure of merit", …)
- Actual beam position measured (with error) at 24 BPMs
- Bmad can be used in an optimizer to find inputs that better <u>control the beam</u>
  - If Bmad is an accurate "twin" of the real machine
  - Model accuracy depends on assumed, but unknown characteristics of the machine



Uncertainty in accelerator control

- <u>Objective:</u> Steer the beam (or control other beam properties)
- <u>Problem:</u> Imperfect knowledge of the relationship between system inputs (currents) and outputs (beam position)
    - Magnet misalignments
    - Transfer function between current and magnetization
    - Current set points not identical to realized currents in system
- Imperfect modeling can lead to *incorrect* control policy, but we never have *perfect* knowledge





Parameter estimation (tuning)

- **Controls** c: known inputs that the operator specifies (currents, …)
- **Parameters** θ: fixed but unknown system properties (misalignments, current biases, …)
- **Model** $m(c;\theta)$: response of the system to its controls, assuming parameters are known
  - e.g., predicted beam position due to currents, if we knew all machine characteristics
  - Here we use Bmad as a "digital twin"
- **Measurements** $y(c)$: observed system response to the control

- Estimate parameters by fitting model to measurements, e.g. by least squares:

$$\hat{\theta} = \text{arg min}_\theta \sum_i (y_i - m_i(c;\theta))^2$$

Parameter estimation (inference)

- In **parameter fitting**, the goal is to find the best-fitting set of parameters

$$\hat{\theta}$$

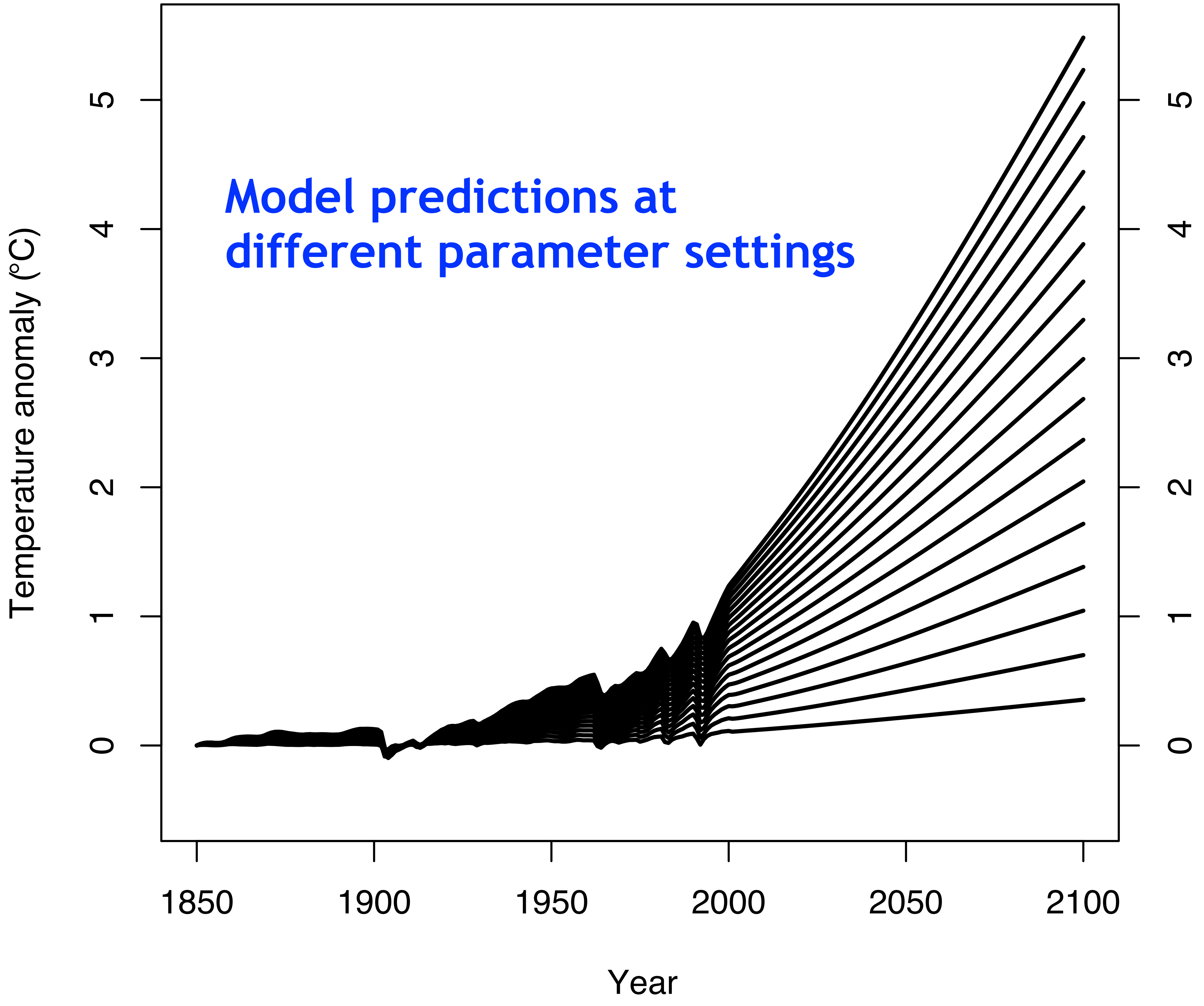
- In Bayesian **uncertainty quantification** (UQ), the goal is to estimate a probability distribution over the unknown parameters, not just a single point estimate (best fit).
  - Posterior distribution (probability of unknown parameters, conditional on measurements):

$$p(\theta \quad y)$$

- When do you want to go to the trouble of UQ?
  - May be *many* “best fits”, with different implications for predicted behavior
  - (in pure science) To put error bars on predictions (e.g,. compare theory and experiment)
  - (in control) Nonlinear response / non-Gaussian errors mean that *best fit parameters* don’t correspond to *controller with best average performance*
  - (in control) We might want to know the expected reliability of a control policy

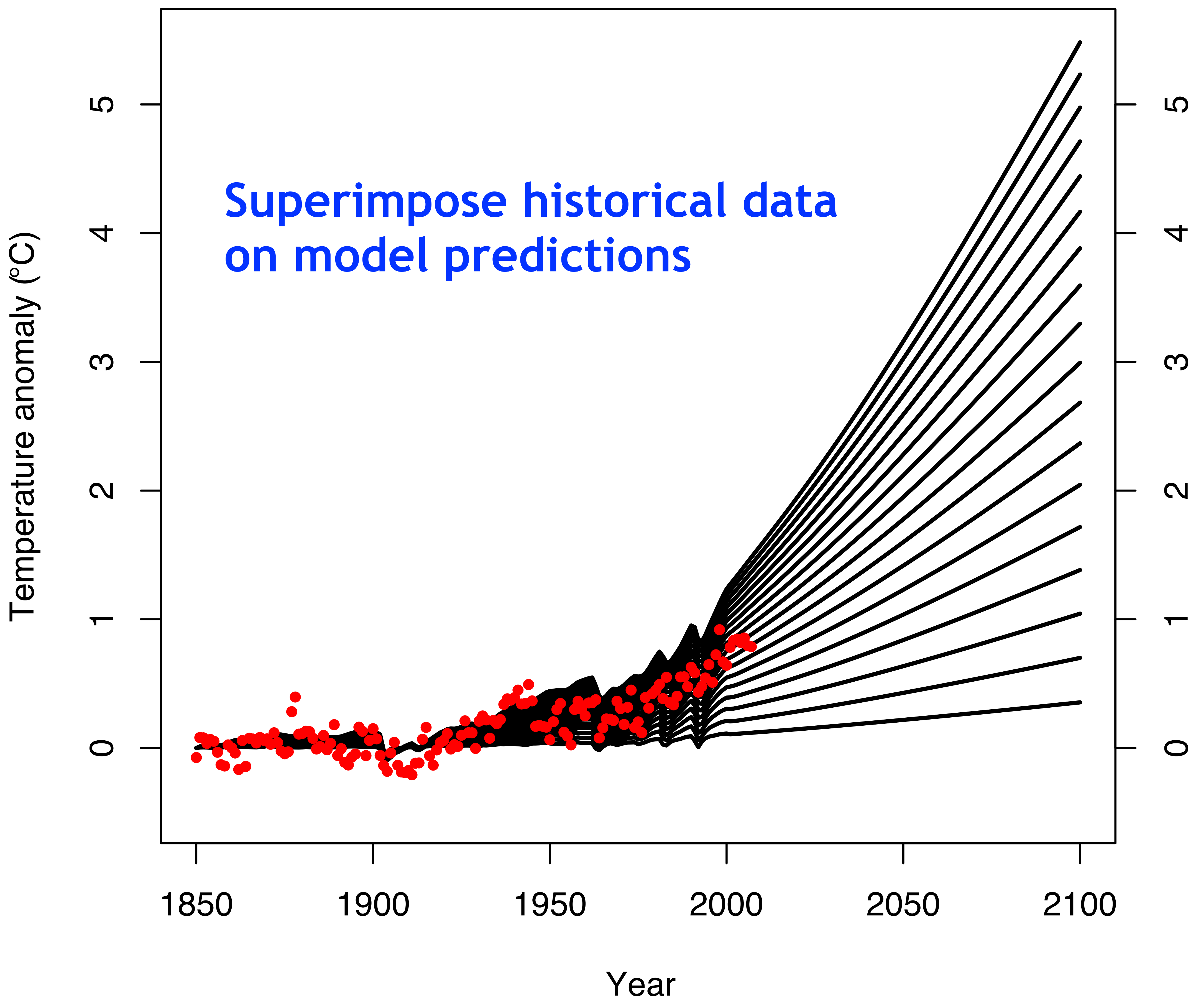
Probabilistically fitting a model to data

- Example of a 3-parameter model from climate science
- Could tune these parameters to data
- But rather than a point estimate, we can assign each parameter value a probability weight
  - Weight given by “goodness of fit”
- It is (probabilistic, nonlinear) **regression**



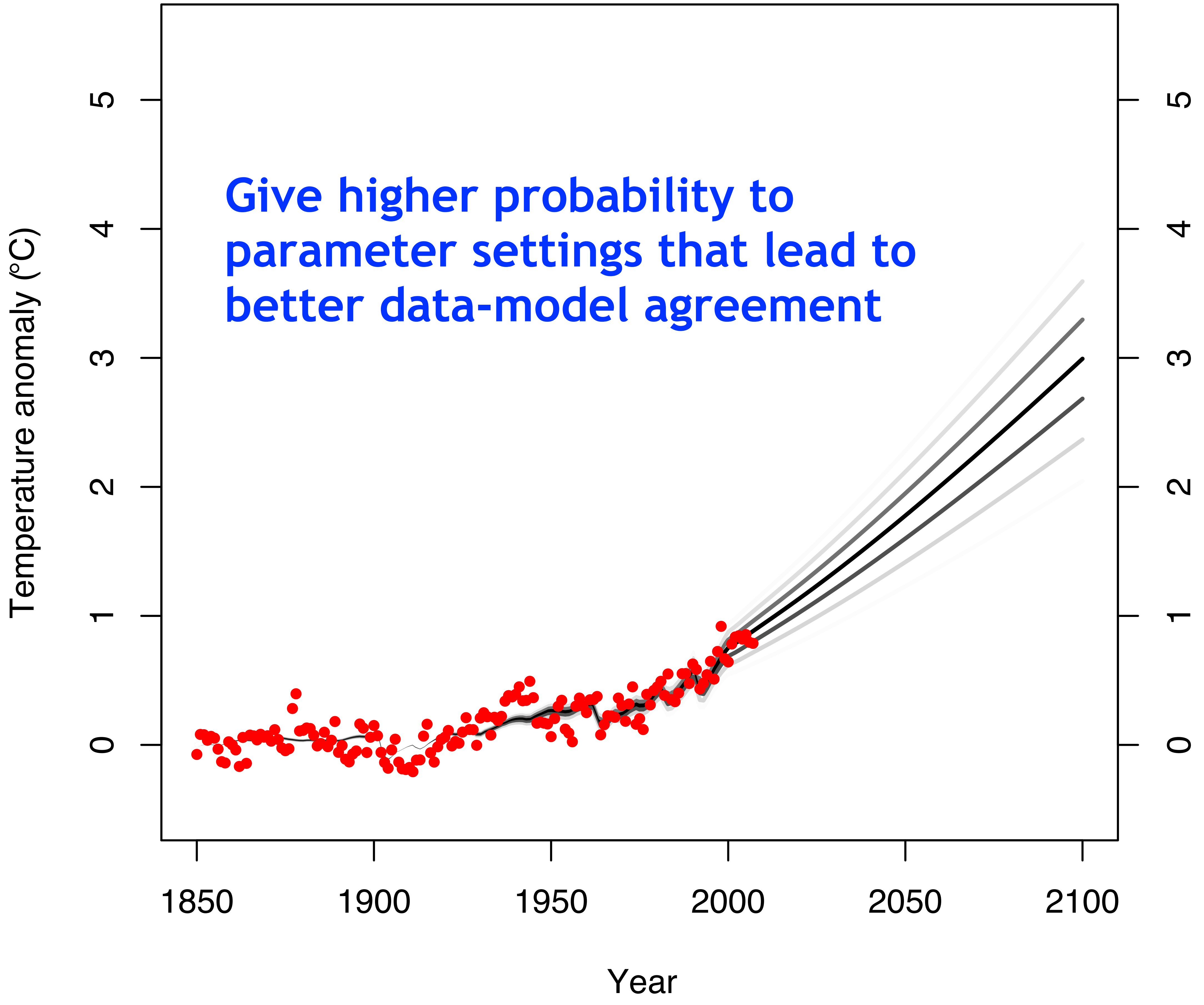
Probabilistically fitting a model to data

- Example of a 3-parameter model from climate science
- Could tune these parameters to data
- But rather than a point estimate, we can assign each parameter value a probability weight
  - Weight given by "goodness of fit"
- It is (probabilistic, nonlinear) **regression**



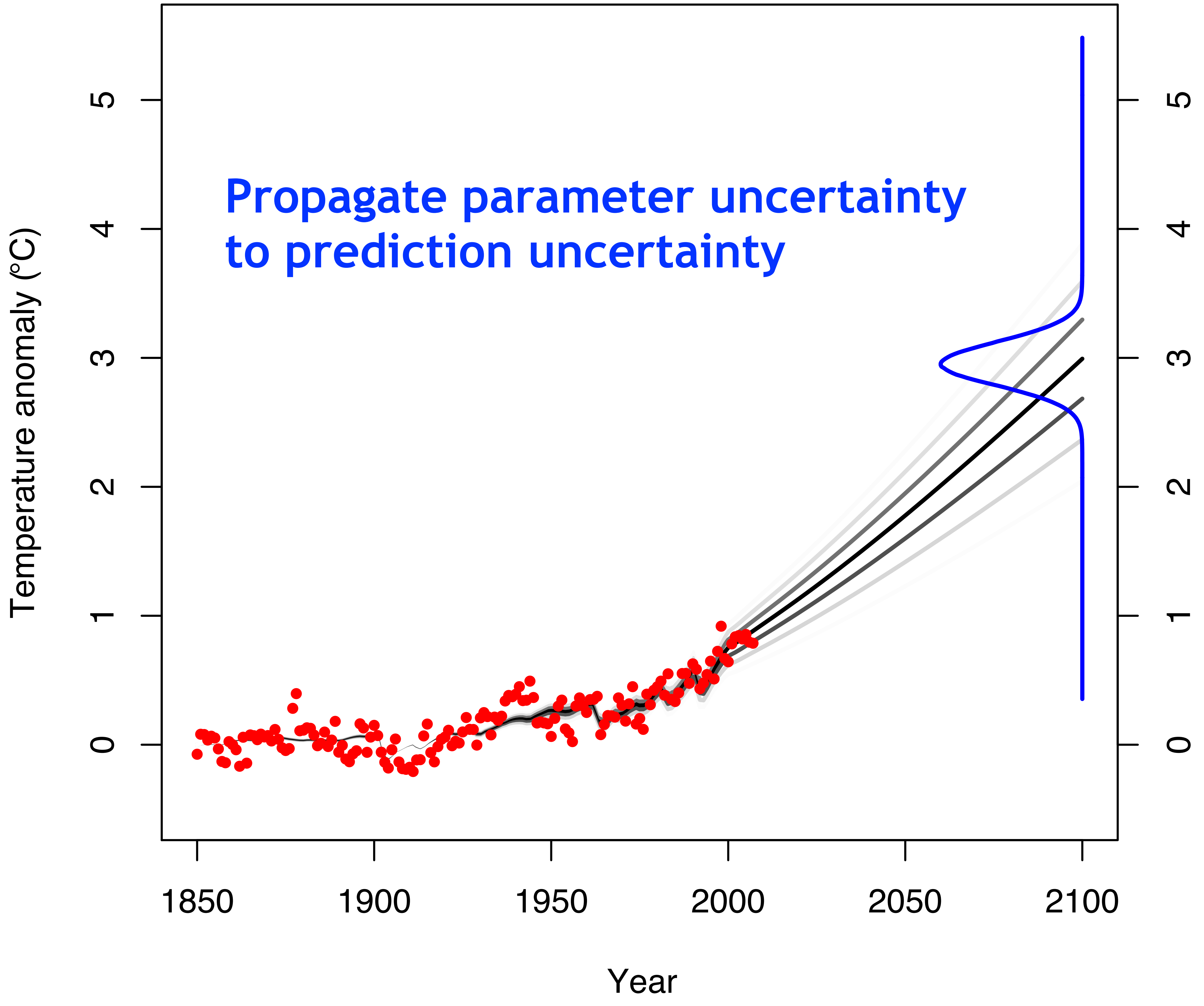
Probabilistically fitting a model to data

- Example of a 3-parameter model from climate science
- Could tune these parameters to data
- But rather than a point estimate, we can assign each parameter value a probability weight
  - Weight given by “goodness of fit”
- It is (probabilistic, nonlinear) **regression**



Probabilistically fitting a model to data

- Example of a 3-parameter model from climate science
- Could tune these parameters to data
- But rather than a point estimate, we can assign each parameter value a probability weight
  - Weight given by “goodness of fit”
- It is (probabilistic, nonlinear) **regression**



Posterior distribution: p(parameters|data)

Bayesian inference (probabilistic parameter estimation)

- Goal: infer parameter probability density functions (PDFs) from data
  - *Conditional* inference: infer parameter uncertainties from known data

<u>Bayes theorem:</u> $p(\text{parameters}|\text{data}) = p(\text{data}|\text{parameters})\ p(\text{parameters})\ /\ p(\text{data})$

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

To infer <u>posterior</u> PDF, need to know <u>likelihood function</u> (data-generating distribution) and <u>prior</u> distribution (beliefs about parameters before seeing the data).

Bayesian uncertainty quantifies “ignorance” about the true parameter values.

Prior distribution: *p*(parameters)

- What you believe about the parameters before you've seen the data
  - Use outside information (physical predictions, other data sources)
  - Priors must be independent of conditioning data (no double-counting)
  - Can use posterior inferred from other data as prior (sequential Bayesian update)

- Elicit booster prior uncertainties from operators
  - trim current errors $\approx\pm10^{-3}$ (1-σ)
  - magnet misalignments informed from previous surveys
  - transfer function coefficient ranges harder to elicit (not directly measured)

Likelihood function: *p*(data|parameters)

Assume data is distributed randomly (additively) around an accelerator model (e.g. Bmad):

Measurements(*BPM location i*) = Model(*control*; *parameters*) + Noise

$$y_i = m(c;\theta) + \varepsilon$$

Assume noise process is noise process (ε) is normal (independent and identically distributed, or *iid*), zero mean: $\varepsilon \sim N(0, \sigma^2)$

$$y_i \sim N(\mu = m_i(c;\theta), \sigma^2)$$

(Likelihood: one observation)

$$p(y_i \ \ \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(y_i - m_i(c;\theta))^2}{\sigma^2}\right]$$

(Likelihood: all observations)

$$p(y \ \ \theta) = \Pi_i \, p(y_i \ \ \theta) = \frac{1}{\left(\Pi_i \sqrt{2\pi\sigma_i^2}\right)} \exp\left[-\frac{1}{2}\frac{\sum_i (y_i - m_i(c;\theta))^2}{\sigma^2}\right]$$

Likelihood function: *p*(data|parameters)

Note: for an *iid* normal likelihood model, the *maximum likelihood estimate* (MLE) for θ is the same as a *least squares* or *minimum* χ^2 fit.

(Likelihood: all observations)

$$p(y \quad \theta) = \Pi_i \, p(y_i \quad \theta) = \frac{1}{\left(\Pi_i \sqrt{2\pi\sigma_i^2}\right)} \exp\left[-\frac{1}{2}\frac{\sum_i (y_i - m_i(c;\theta))^2}{\sigma^2}\right] \propto \exp(-\chi^2/2)$$

Assume noise process is noise process (ε) is normal (independent and identically distributed, or *iid*), zero mean: $\varepsilon \sim N(0, \sigma^2)$

$$y_i \sim N(\mu = m_i(c;\theta), \sigma^2)$$

Posterior distribution: *p*(parameters|data)

The posterior is proportional to the product of the likelihood and prior (which we will assume is independent for each parameter).

$$p(\theta \quad y) \propto p(y \quad \theta)\, p(\theta) = \frac{1}{\left(\prod_i \sqrt{2\pi\sigma_i^2}\right)} \exp\left[-\frac{1}{2}\frac{\sum_{i=1}^{N}(y_i - m_i(c;\theta))^2}{\sigma_i^2}\right] \times \prod_{k=1}^{K} p(\theta_k)$$

The log posterior is like a “regularized” least squares fit. If the priors are assumed normal around some typical mean, $\theta_k \sim N(\bar{\theta}_k, \nu_k^2)$, then the “maximum a posteriori” (MAP) estimate arises from minimizing a least squares term with an additional “penalty” term on the parameters.

$$-\log p(\theta \quad y) \propto \sum_{i=1}^{N} \frac{(y_i - m_i(c;\theta))^2}{\sigma^2} + \sum_{k=1}^{K} \frac{(\theta_k - \bar{\theta}_k)^2}{\nu^2} + const$$

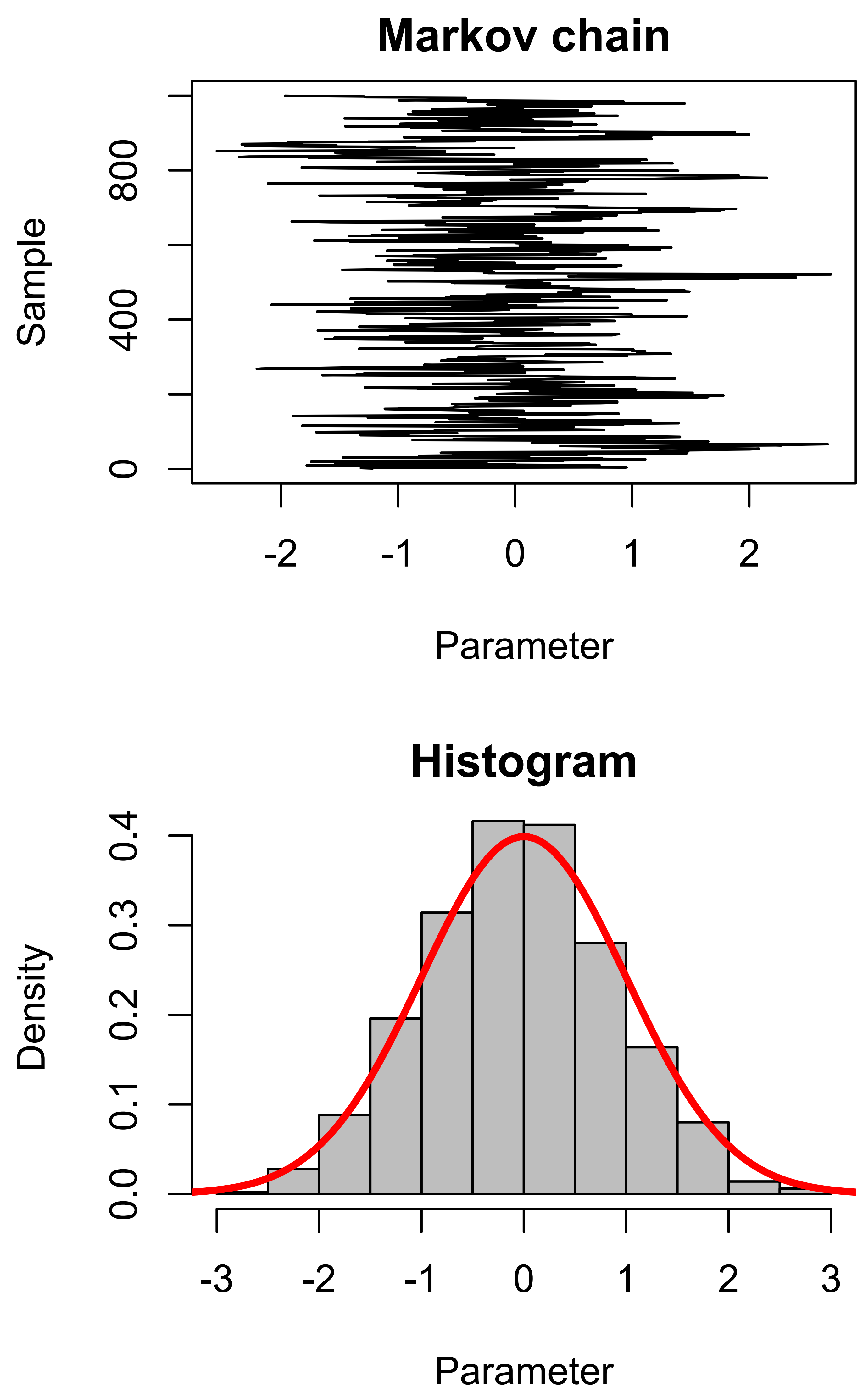
Posterior distribution: *p*(parameters|data)

- *However*: These relationships are just to connect to some familiar concepts.
- In UQ, we usually are not interested in point estimates.
  - (and if we do make a point estimate, it’s usually the posterior mean, not MAP)
- Our real goal is *uncertianty*, which means the full posterior distribution
  - Its mean, variance, and all higher moments

$$p(\theta \quad y) \propto p(y \quad \theta)\, p(\theta) = \frac{1}{\left(\prod_i \sqrt{2\pi\sigma_i^2}\right)} \exp\left[-\frac{1}{2}\frac{\sum_{i=1}^{N}(y_i - m_i(c;\theta))^2}{\sigma_i^2}\right] \times \Pi_{k=1}^{K}\, p(\theta_k)$$

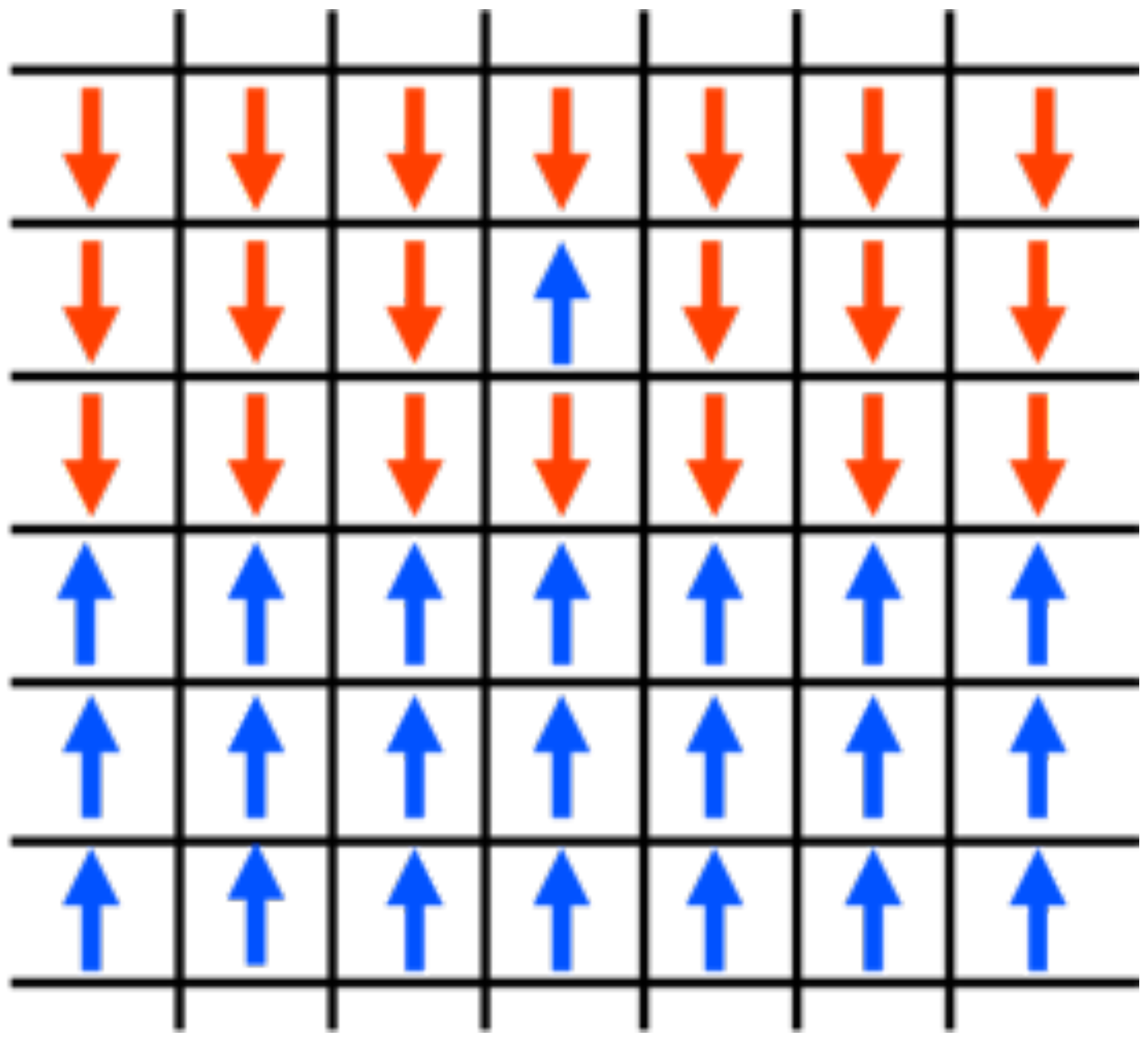
Markov chain Monte Carlo (MCMC) sampling

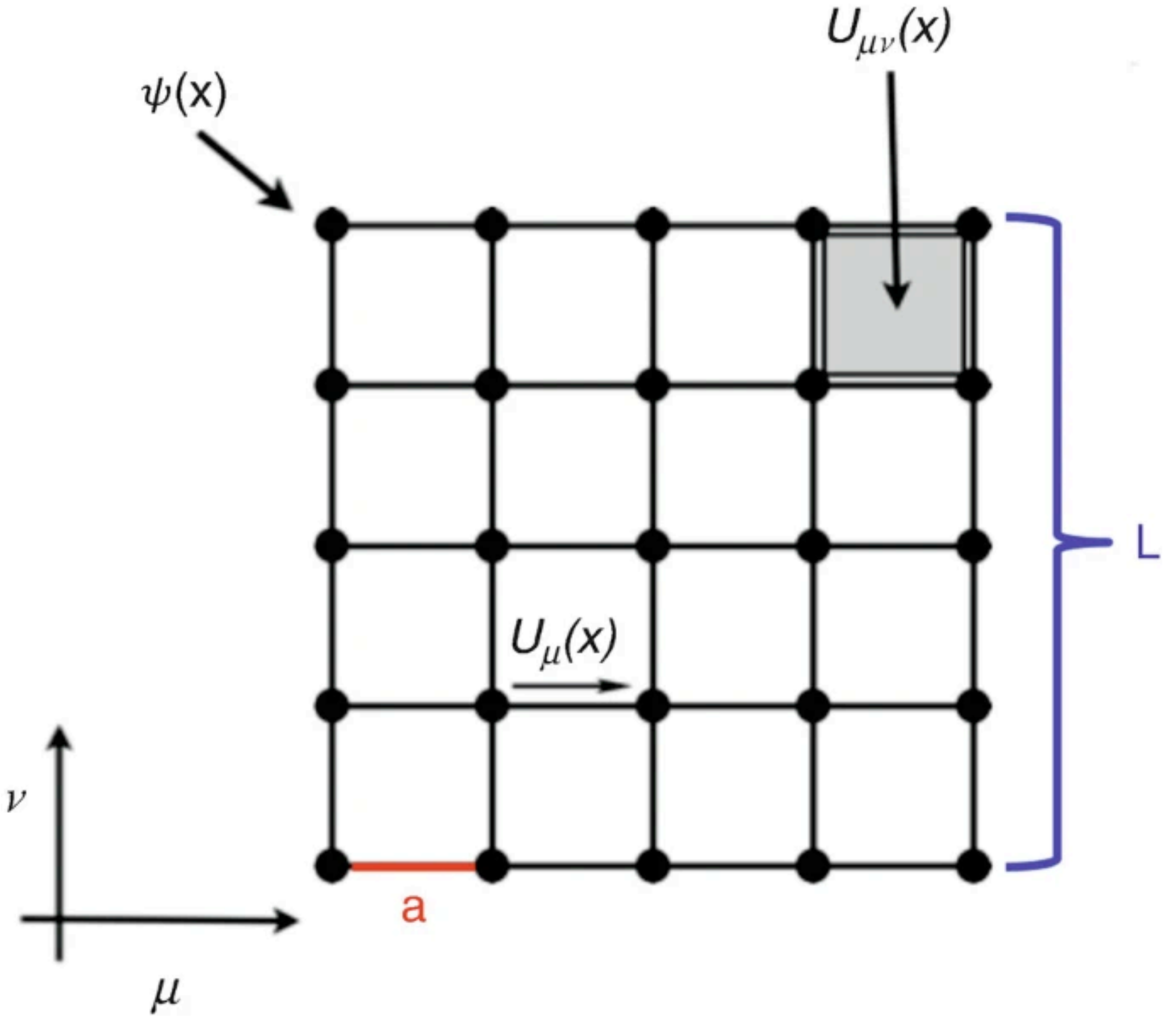
- We want to calculate the posterior distribution. In high dimensions, Monte Carlo sampling works best.
  - sampling converges like $1/\sqrt{N}$, where *N* is # of samples
- How to sample from an arbitrary distribution?
- Approach: <u>importance-biased random walk</u>
  - spend more time sampling high-probability regions
  - (note: samples from a random walk are not independent)



Physics note: MCMC

- Sampling from a probability distribution $p(x)$ is directly analogous to statistical mechanics
  - Sample Boltzmann distribution $p(x) \propto e^{-\beta E(x)}$
  - -log $p(x)$ is analogous to *potential energy*
- Or lattice gauge theory
  - $p(x) \propto e^{-S[x]}$
  - -log $p(x)$ is analogous to the *action*
- Advanced Bayesian inference uses *hybrid Monte Carlo* (HMC), just like lattice QCD
  - Requires calculating gradient of $p(x)$
  - Which for us means the gradient of the model output (e.g., Bmad beam position) w.r.t. the parameters
  - <u>*Differentiable Bmad* would be very helpful</u>





Metropolis MCMC algorithm

LANL

- Let the target distribution $\pi(\theta)$ be the posterior, $p(\theta|y)$
- Construct a random walk as follows:

1. Start at point θ
2. Propose moving to a new point θ' randomly, according to some easy to sample symmetric distribution $t(\theta'|\theta)$ (e.g., a Gaussian perturbation)
3. If this moves us to a higher probability point, $\pi(\theta') > \pi(\theta)$, accept the move to θ'
4. If this moves us to a lower probability point, accept randomly with probability $\pi(\theta')/\pi(\theta)$; else reject and stay at the same point θ
5. Either way, record the point you end up at to construct the Markov chain
6. Repeat

Code for Bayesian regression

julia

```julia
function metropolis(lpdf, num_iter, x₀, step)
    D = length(x₀)
    chain = zeros(num_iter, D)
    chain[1,:] = x₀
    x, lp = x₀, lpdf(x₀)
    num_accept = 0

    for i = 2:num_iter
        x′ = x + step .* randn(D) # proposal
        lp′ = lpdf(x′)

        if log(rand()) < lp′ - lp # Metropolis
            x, lp = x′, lp′
            num_accept = num_accept + 1
        end

        chain[i,:] = x
    end

    return (chain, num_accept/num_iter)
end
```

```julia
function model(p)
    λ,d,α,T₀ = p
    Δt = 31557600. # year [s]
    C = 4184000 * d # heat capacity/area [J/K/m^2]
    F = forcing_non_aerosol + α*forcing_aerosol
    T = zero(F)
    for i in 1:length(F)-1
        T[i+1] = T[i] + (F[i] - λ*T[i])/C * Δt
    end
    return T .+ T₀
end
```

```julia
function log_posterior(p)
    λ,d,α,T₀ = p
    log_post = -Inf

    if λ > 0 && d > 0 && α > 0 # parameters in range
        F2xCO₂ = 4.0 # forcing for doubled CO₂ [W/m²]
        lpri_λ = logpdf(LogNormal(log(3), log(2)/2), F2xCO₂/λ)
          + log(F2xCO₂/λ^2) # ECS prior + Jacobian (ECS = F2xCO₂/λ)
        lpri_d = logpdf(Normal(100, 25), d)
        lpri_α = logpdf(LogNormal(log(1), log(1.5)/2), α)
        lpri_T₀ = 0
        log_pri = lpri_λ + lpri_d + lpri_α + lpri_T₀ # prior

        σ = 0.1 # observational noise standard deviation [K]
        r = temp_obs - model(p)[midx] # data-model residual
        log_lik = sum(logpdf.(Normal(0,σ), r)) # likelihood
        log_post = log_lik + log_pri # posterior
    end

    return log_post
end
```

Optimizing control inputs

- **Control** *c*: currents or other inputs that the operator can specify
- **Model** *m*(*c*): the modeled system response to inputs (e.g., beam position)
- **Objective**: a metric of system performance (e.g., a loss function) to optimize
  - $\mathscr{L}(m(c)) = \sum_i (\bar{z}_i - m_i(c))^2$ (deviation of beam position from target position at BPMs)
  - (e.g., $\bar{z}_i = 0$)

- Find control that optimizes objective:

$$c^\star = \text{arg min}_c\, \mathscr{L}(m(c))$$

- Solve using standard optimization algorithms (quasi-Newton, gradient descent, …)

Stochastic optimization for control inputs

- **Control** c: inputs that the operator can specify
- **Parameters** θ: unknown system characteristics
    - Assume we have inferred a distribution $p(\theta)$ representing parameter uncertainty (e.g. a posterior $p(\theta|y)$)
- **Model** $m(c;\theta)$: the modeled system response to inputs (e.g., beam position)
- **Objective**: a metric of system performance (e.g., a loss function) to optimize
    - $\mathscr{L}(m(c;\theta\ \ y)) = \sum_i (\bar{z}_i - m_i(c;\theta))^2$ (deviation of beam position from target position at BPMs)
- Stochastic control aims to be *robust to uncertainties* in quantities we can't estimate perfectly
- Find control that optimizes *expected* objective (average over Monte Carlo parameter samples $\{\theta_j\}$):

$$c^\star = \text{arg min}_c \, \mathbb{E}_{\theta\ y}[\mathscr{L}(m(c;\theta))]$$

$$\approx \frac{1}{J} \sum_{j=1}^{J} \sum_{i=1}^{N} (\bar{z}_i - m_i(c;\theta_j))^2$$

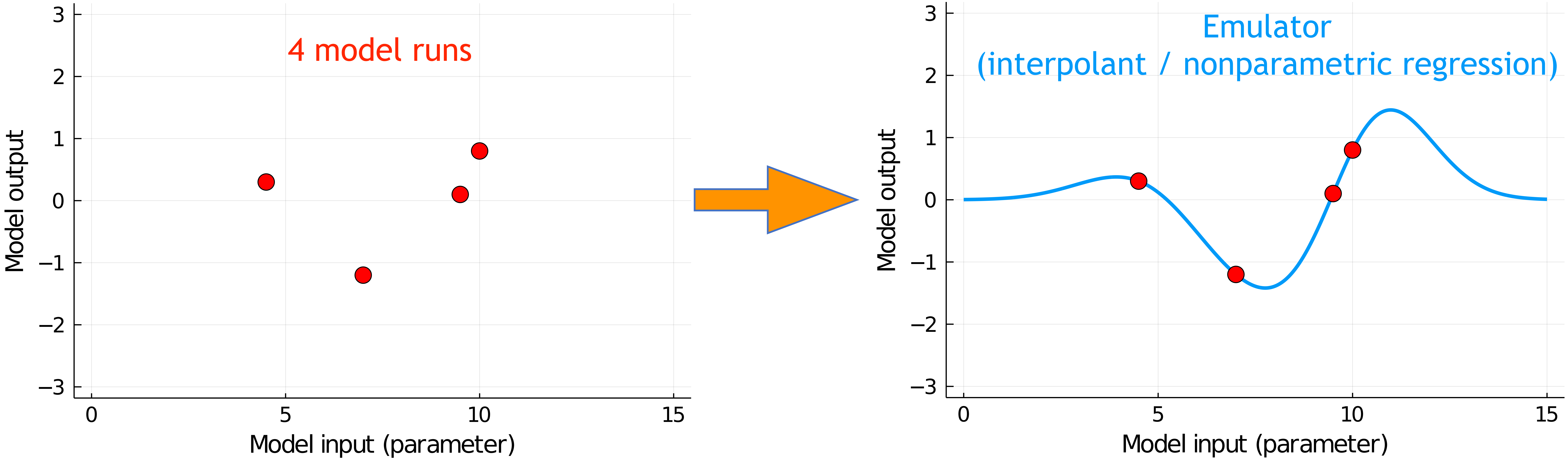
- Solve with a *stochastic* optimizer (designed to handle noisy objective functions)

On optimal control methods

- There are many optimization methods floating around
  - Bayesian optimization, gradient descent, quasi-Newton methods, …
- There are many ways to formulate beam control as an optimization problem
  - Nonlinear loss minimization, expected utility maximization (with chance constraints), robust optimization/control, classical control theory, reinforcement learning
- Probably a digression to discuss pros/cons in this talk, but we should discuss in the project
- The methods discussed here are adapted for this setting:
  - There is a physical system model, which is much cheaper than real experiments
  - We can solve control policies offline using the physical model (digital twin)
  - The model is imperfect, but imperfections are learnable via data-model comparisons
  - There are many variables to control; maybe many uncertain system parameters
  - Decisions are one-off / non-sequential (if sequential, can extend to RL-like approaches)

Model emulation

- We can only afford a limited number of Bmad simulations; hard to embed in Monte Carlo sampler where many evaluations are required
- Can we estimate "what the model would have predicted at a new parameter setting" from an ensemble of training simulation output, without actually running the model?
- "Response surface" emulation: *interpolation* to the rescue
  - Gaussian processes (as in Bayesian optimization), neural networks, other regression approaches

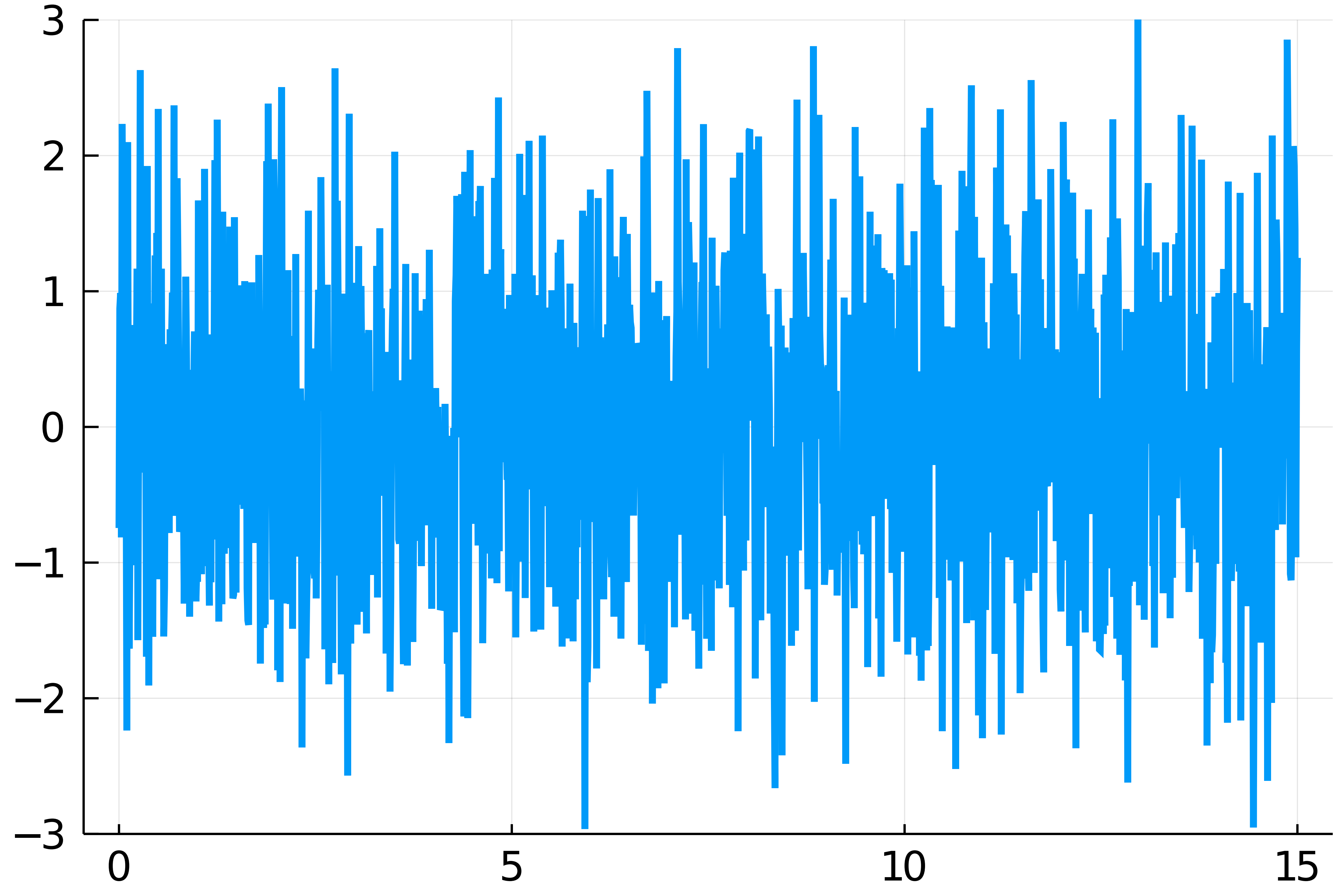


Gaussian process regression as emulation

- A Gaussian processes is a probability distribution on a space of *functions*
- Can be used for *probabilistic* interpolation / regression

- Draw, say, 1000 Gaussian random samples and plot them over “space”:

$Y_i \sim N(0,1)$



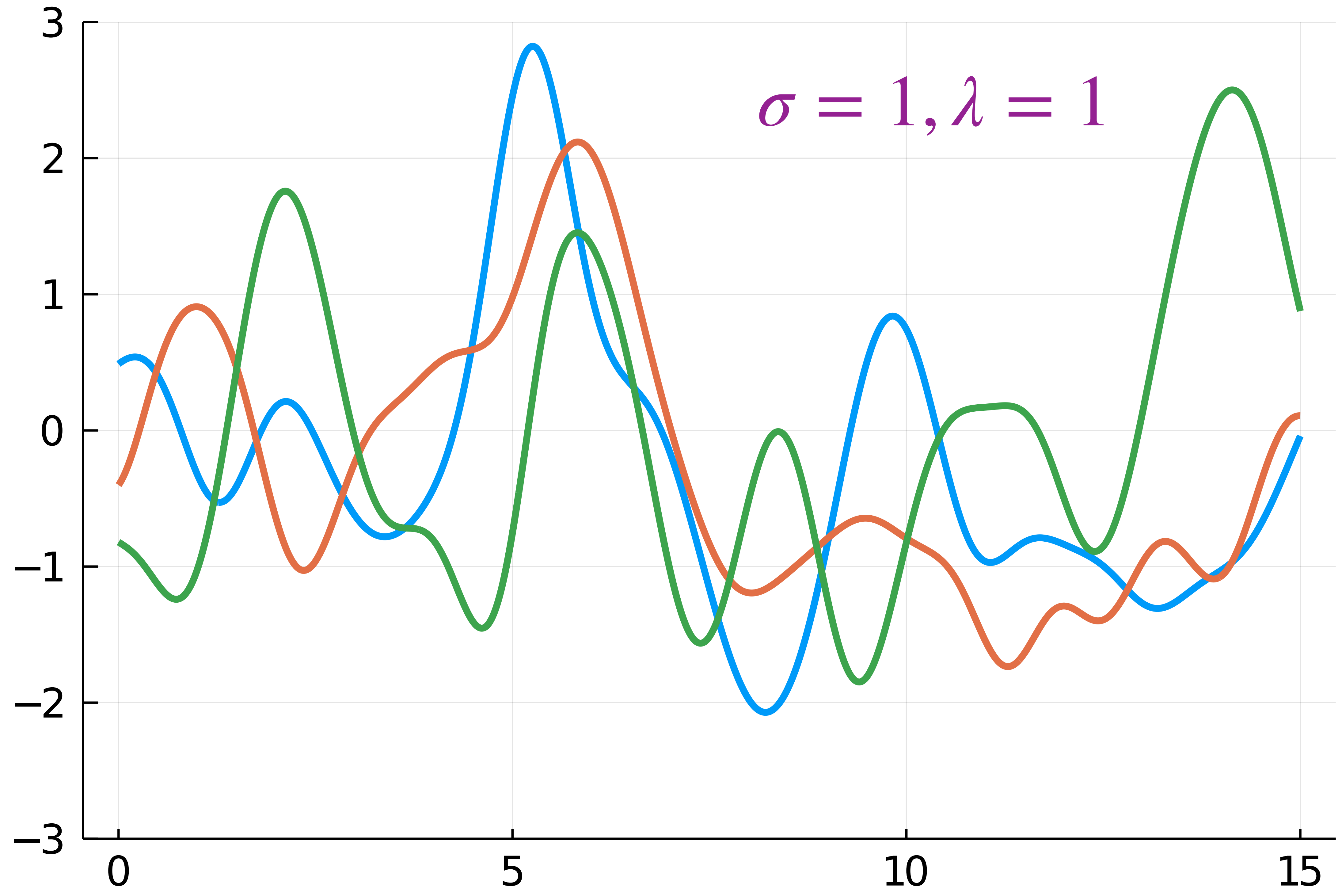
Gaussian process regression as emulation

- A Gaussian processes is a probability distribution on a space of *functions*
- Can be used for *probabilistic* interpolation / regression

- Draw 1000 random variables, but *correlated with each other*; here are 3 draws:

$$Y \sim N(0,\Sigma), \qquad \Sigma_{ij} = \mathrm{Cov}(Y_i, Y_j)$$

$$\mathrm{Cov}(Y_i, Y_j) = \sigma^2 \exp\left[-\left(\frac{X_i - X_j}{\lambda}\right)^2\right]$$



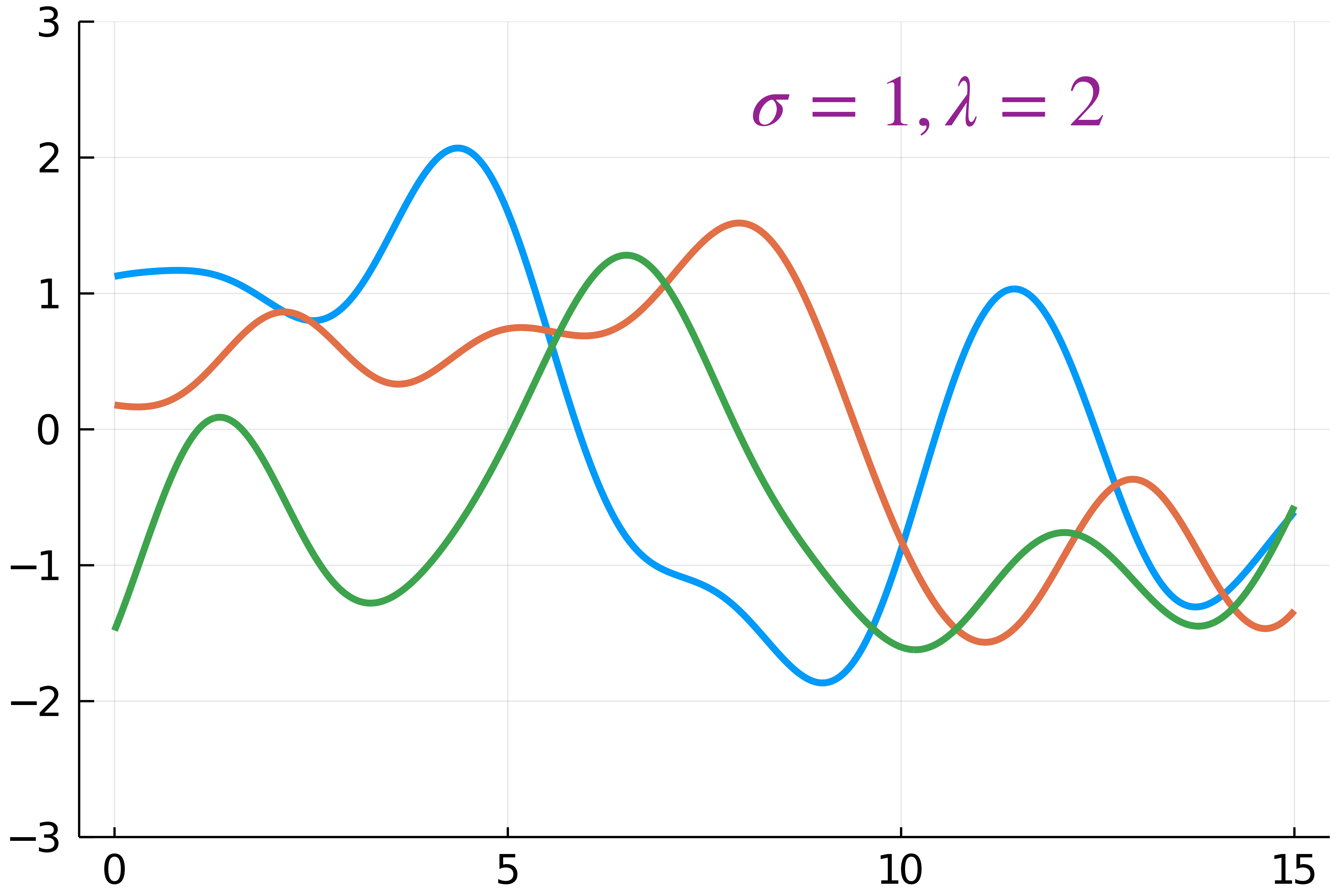
Gaussian process regression as emulation

- A Gaussian processes is a probability distribution on a space of *functions*
- Can be used for *probabilistic* interpolation / regression

- Draw 1000 random variables, but *correlated with each other*; here are 3 draws:

$$Y \sim N(0, \Sigma), \qquad \Sigma_{ij} = \mathrm{Cov}(Y_i, Y_j)$$

$$\mathrm{Cov}(Y_i, Y_j) = \sigma^2 \exp\left[- \left(\frac{X_i - X_j}{\lambda} \right)^2 \right]$$



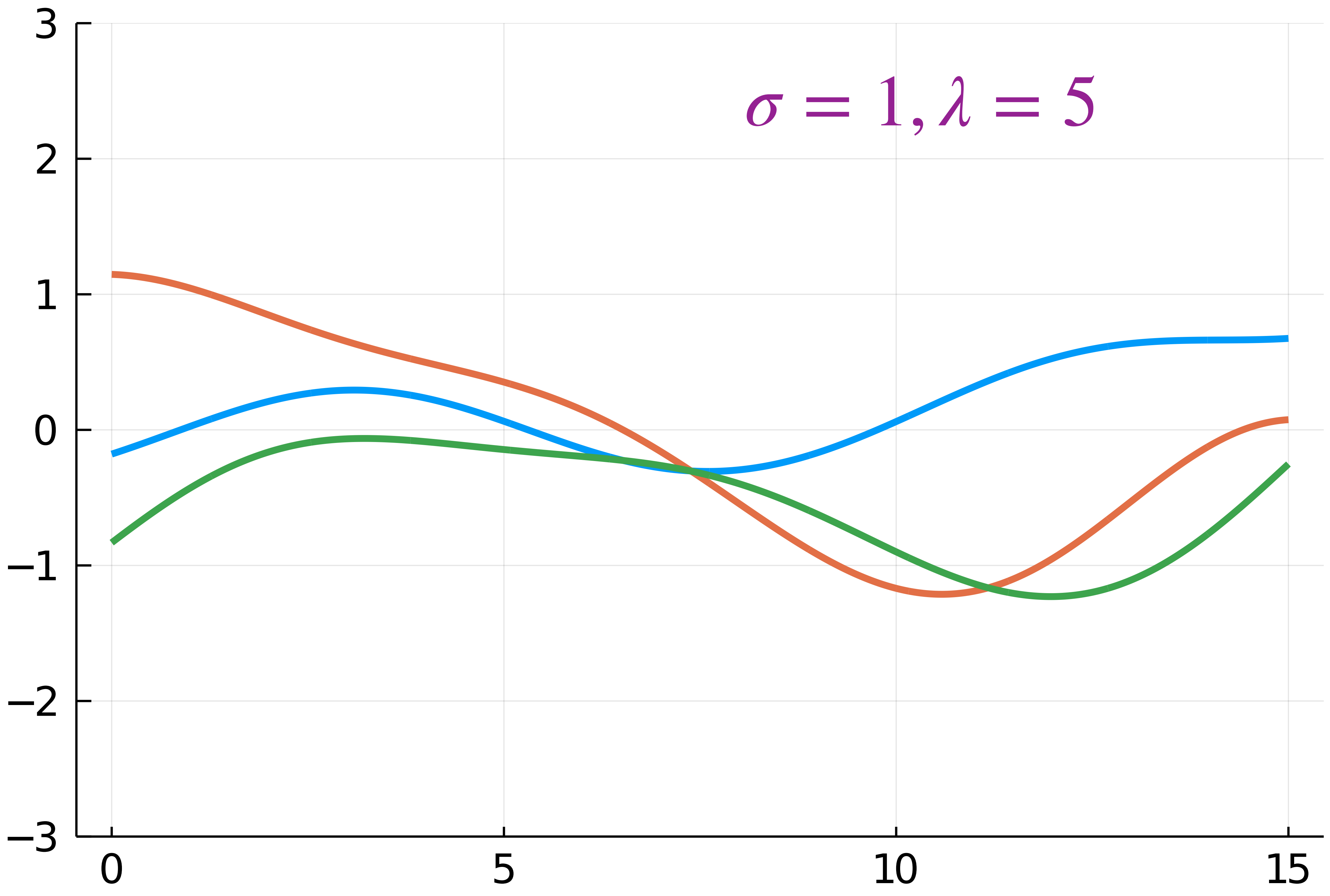
Gaussian process regression as emulation

- A Gaussian processes is a probability distribution on a space of *functions*
- Can be used for *probabilistic* interpolation / regression

- Draw 1000 random variables, but *correlated with each other*; here are 3 draws:

$$Y \sim N(0, \Sigma), \qquad \Sigma_{ij} = \mathrm{Cov}(Y_i, Y_j)$$

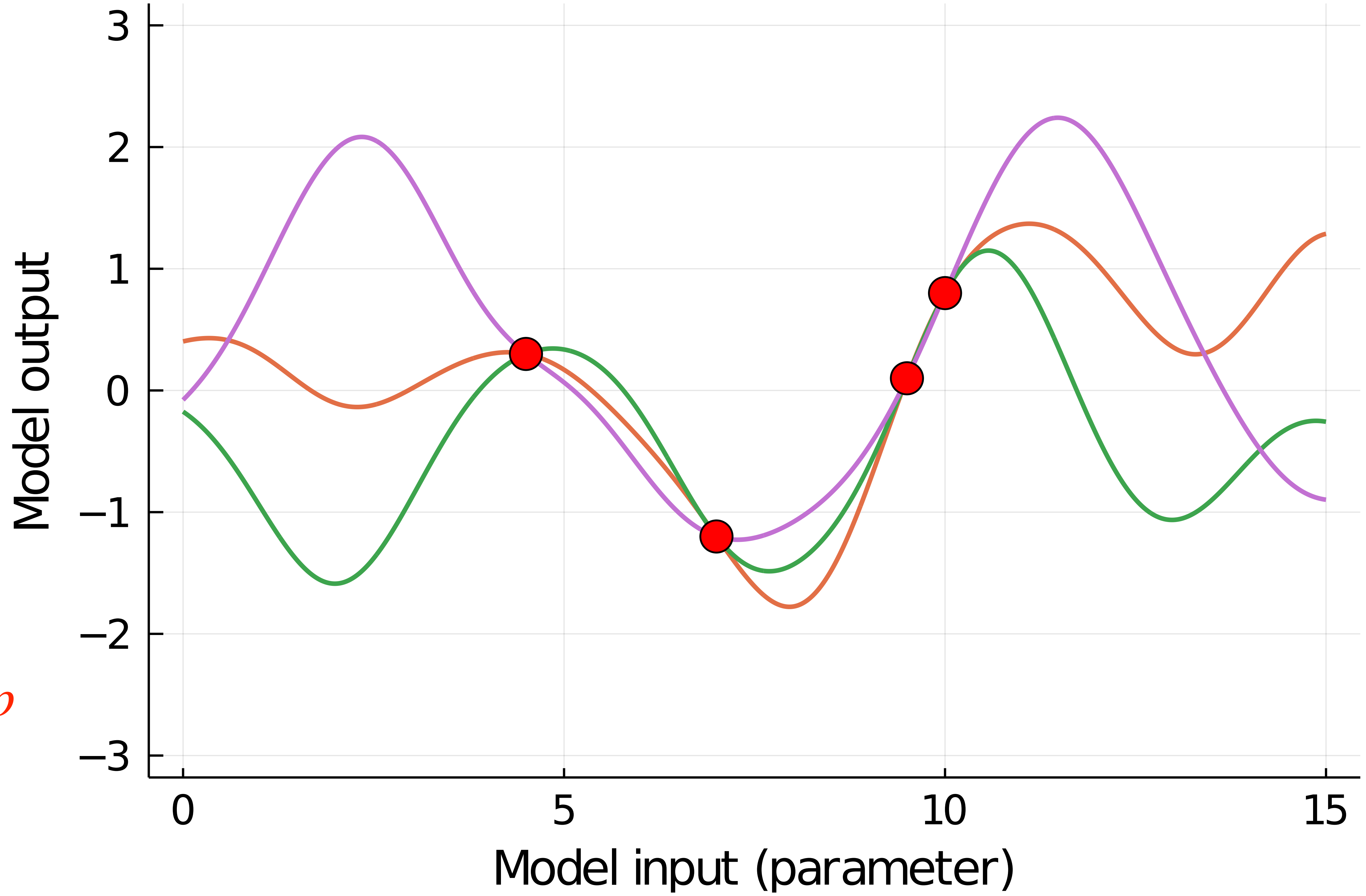
$$\mathrm{Cov}(Y_i, Y_j) = \sigma^2 \exp\left[-\left(\frac{X_i - X_j}{\lambda} \right)^2 \right]$$

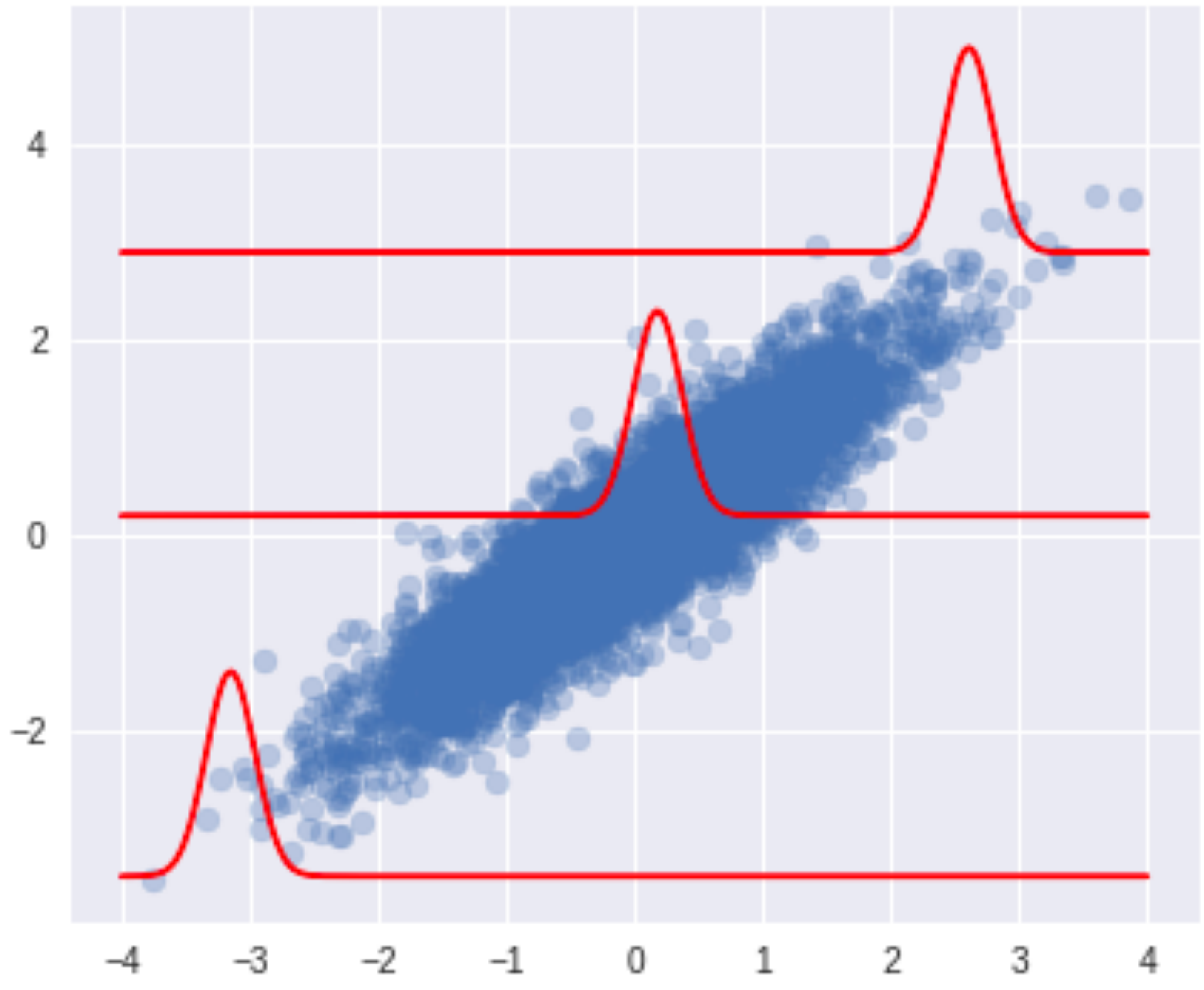


Gaussian process regression as emulation

- We have seen that we can draw *random vectors* that have smooth behavior by imposing a correlation over space (nearer points are more correlated)
- A Gaussian process is the continuum limit of this idea to *random functions*
- We can be Bayesian, and *condition* on "observed" data to get a *posterior*:

$$Y \sim N(\mu^*, \Sigma^*)$$
$$\mu^* = \Sigma_{pt}\Sigma_{tt}^{-1}y_t$$
$$\Sigma^* = \Sigma_{pp} - \Sigma_{pt}\Sigma_{tt}^{-1}\Sigma_{tp}$$



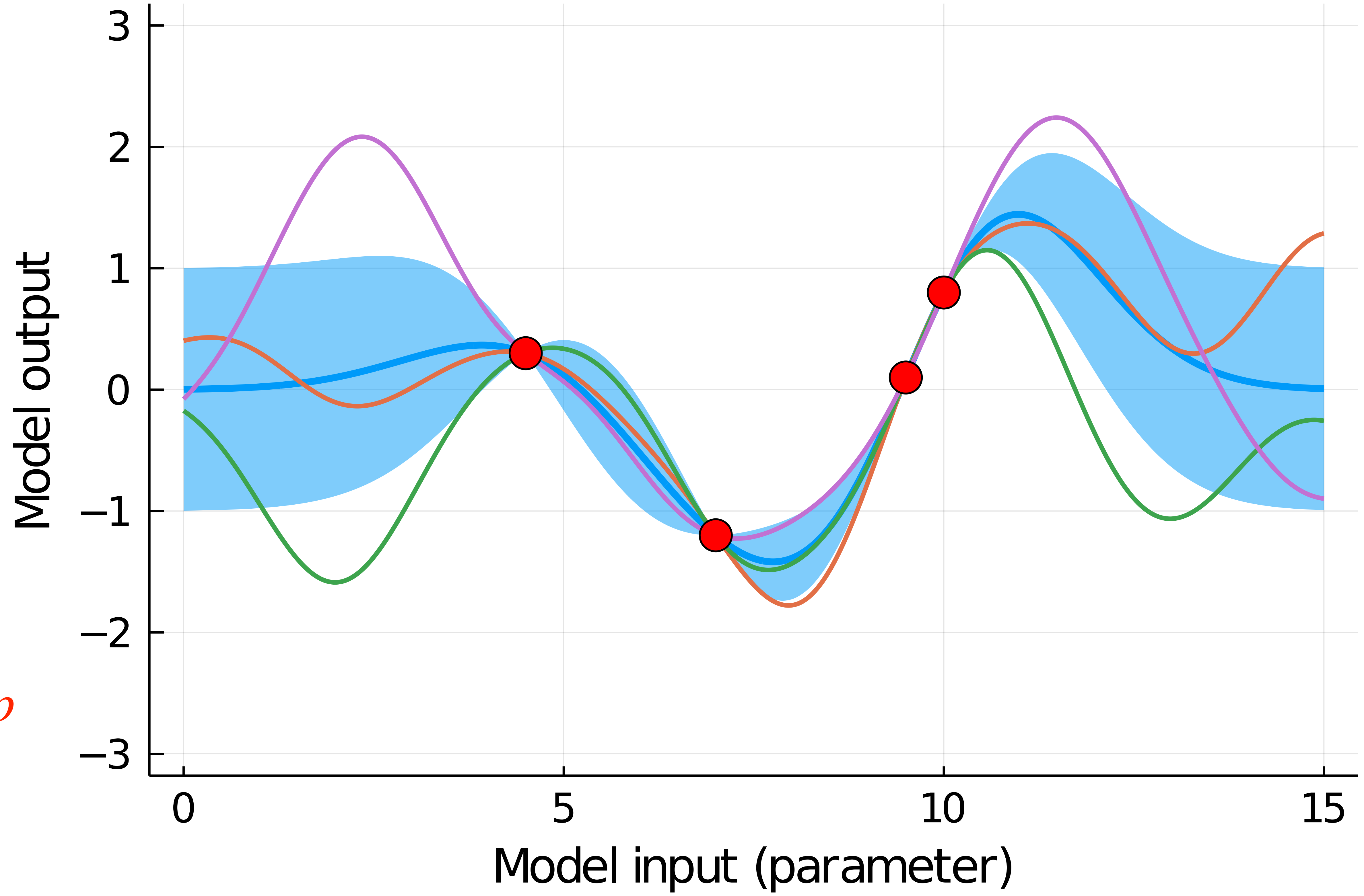


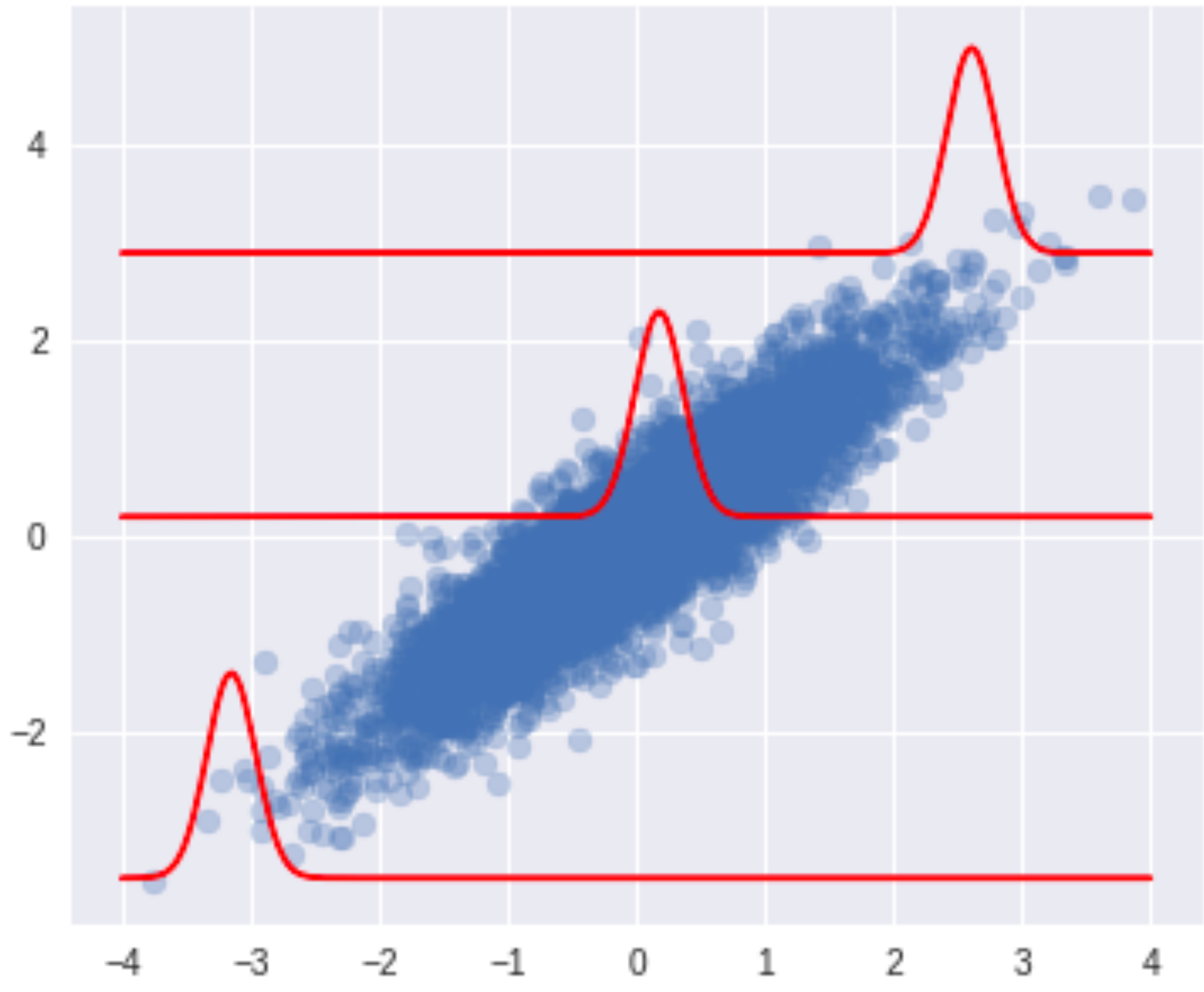
CrossValidated (2019)

Gaussian process regression as emulation

- We have seen that we can draw *random vectors* that have smooth behavior by imposing a correlation over space (nearer points are more correlated)
- A Gaussian process is the continuum limit of this idea to *random functions*
- We can be Bayesian, and *condition* on “observed” data to get a *posterior*:

$$Y \sim N(\mu^*, \Sigma^*)$$
$$\mu^* = \Sigma_{pt}\Sigma_{tt}^{-1}y_t$$
$$\Sigma^* = \Sigma_{pp} - \Sigma_{pt}\Sigma_{tt}^{-1}\Sigma_{tp}$$





CrossValidated (2019)

Errors in variables

- We have assumed that the controls (e.g., currents) are perfectly known, because we set them
- But what if the true control is unknown (currents fluctuate randomly, or there is a persistent but unknown bias between set point and realized current)?
  - The model has noisy inputs in addition to noisy outputs
- We can treat the “true” controls as *parameters* to infer (“latent variables”)
  - Probability model for set current as random perturbation of true current: $\tilde{c}_d \sim N(c_d, \varsigma_d^2)$
  - Find joint posterior for parameters and true currents $p(\theta, c \mid y, \tilde{c})$

$$p(\theta, c \mid y, \tilde{c}) \propto p(y \mid \theta)\, p(c \mid \tilde{c})\, p(\theta)\, p(c)$$

$$\propto \exp\left[-\frac{1}{2}\frac{\sum_{i=1}^{N}(y_i - m_i(c;\theta))^2}{\sigma_i^2}\right] \times \prod_{k=1}^{K}\frac{(\theta_k - \bar{\theta}_k)^2}{\nu_i^2} \times \prod_{d=1}^{D}\frac{(\tilde{c}_d - c_d)^2}{\varsigma_d^2}$$

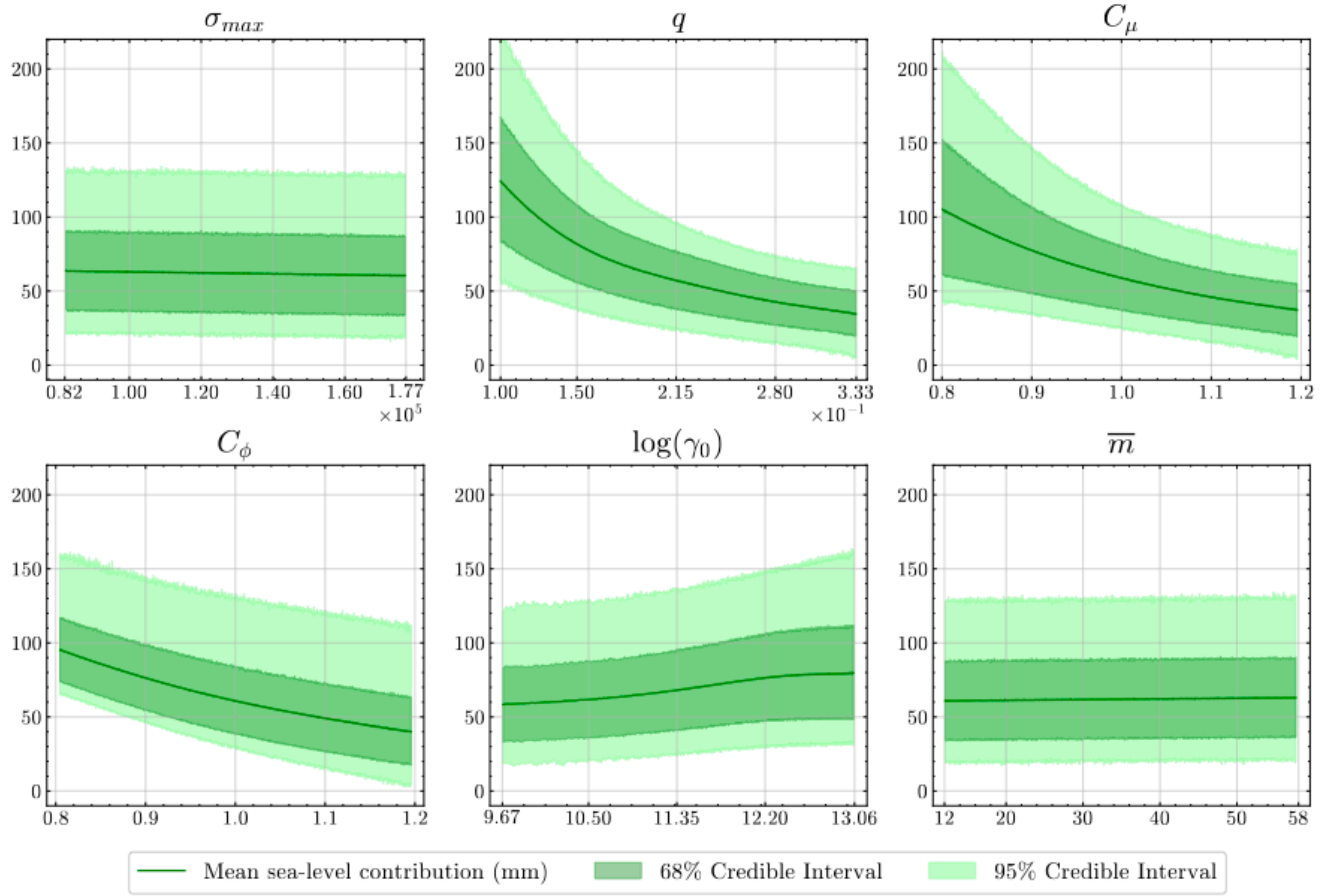
Obtain parameter posterior by integrating out (“marginalizing over”) latent variables: $p(\theta \mid y, \tilde{c}) = \int p(\theta, c \mid y, \tilde{c})\, dc$

Do any of these uncertainties matter?

- So far we've been proceeding under the assumption that we know which parameters are responsible for beam positioning, or Bmad model misfit
  - We just have to quantify their effects
- What if we don't know what matters?
  - Magnet misalignments, transfer function, trim currents
- Can we go through a list of suspects, and identify or quantify their importance?
  - In terms of influence on model prediction, or data-model misfit
- Characterizing the response of outputs to inputs is known as **sensitivity analysis**
- Traditional approach: "one-at-a-time" (OAT) parameter scan
  - Pick a parameter, change its value over a range (fixing all other parameters at nominal)
  - Doesn't pick up any interactions between parameters
  - Can be sample-inefficient (most of the time you aren't learning about most parameters)
  - Be aware of overconfidence: exploring parameters and stopping when one shows an effect

Accounting for uncertainty in sensitivity analysis

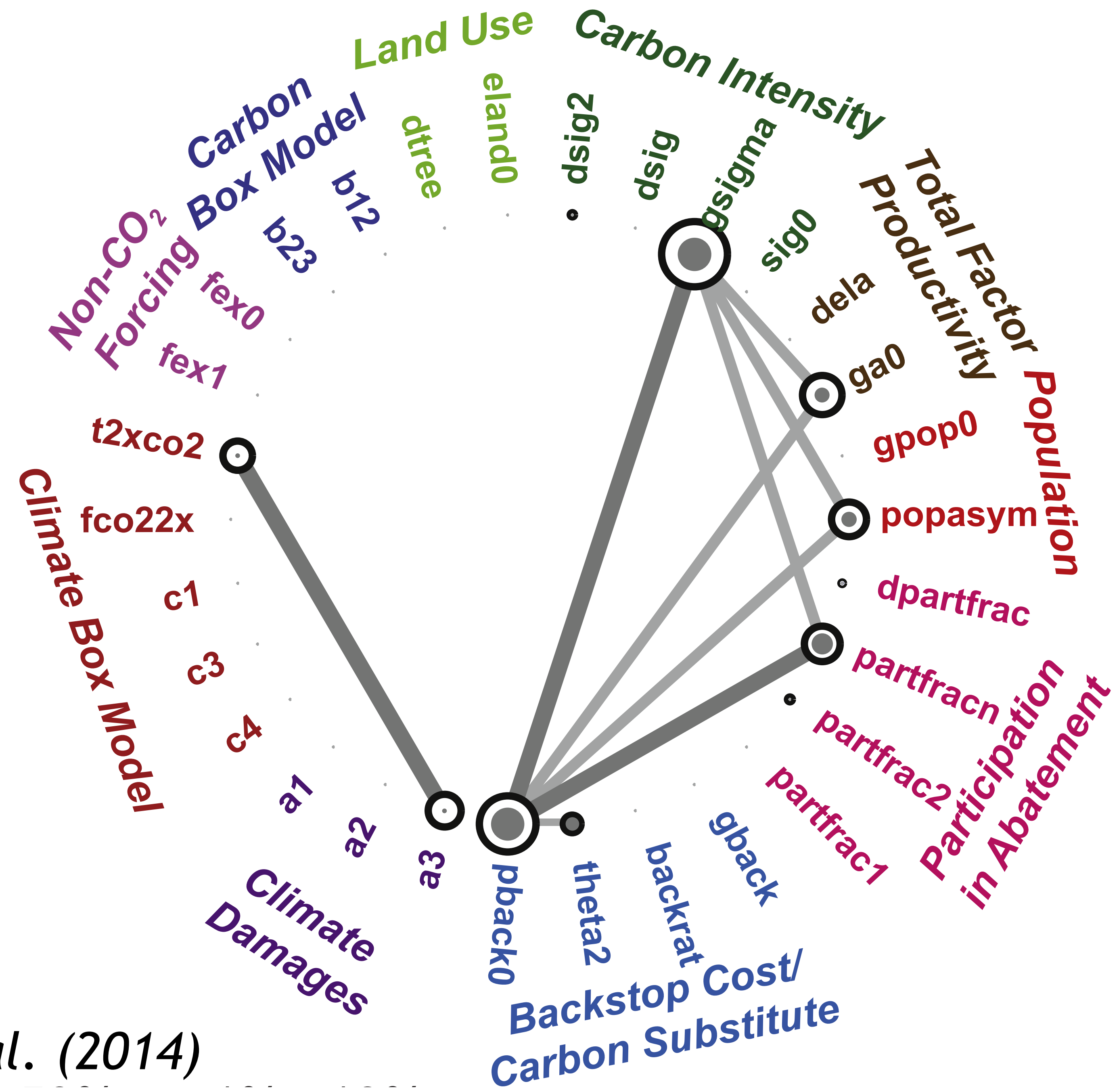
- OAT: change one parameter, holding all others fixed
- Alternative: change one parameter, *sampling randomly* over all other parameters (given a distribution)
  - Accounts for uncertainty in the response of one parameter, due to variability in other parameters



Jantre et al. (2024)

Variance-based global sensitivity analysis (GSA)

- Sobol' decomposition: **Analysis-of-variance (ANOVA) to construct a model's "uncertainty budget"**
  - Requires user to specify a probability distribution over uncertain inputs
- **How much of the output uncertainty can be attributed to the uncertainty in a particular input?**
  - Or, how much could we reduce output uncertainty if we learned the true value of an input?
- **How much does an input contribute directly, and indirectly through correlations with other inputs?**
  - Quantifies importance of (2-way, 3-way, …) interactions between input variables
- **Contrast with "one-at-a-time" parameter scans**
  - Don't identify contributions to output uncertainty, or detect interactions
- **Specific advantages when GSA is coupled with an emulator:**
  - Fast, closed-form analytic solutions for sensitivity metrics
  - Change assumptions about input uncertainties without new simulations



Butler et al. (2014)

Global sensitivity analysis, quantitatively

- How much would we reduce uncertainty in output Y, if we learned the value of the ith input, X_i?
  - Difficulty: we don't know the true value of X_i
- Uncertainty in output due to uncertainty in all inputs = $\mathbf{Var}(Y)$
- Uncertainty in output, after learning the true value x of input X_i = $\mathbf{Var}_{\sim i}(Y|X_i=x)$
- Expected output uncertainty after learning true input, averaged over input uncertainty = $\mathbf{E}_i(\mathbf{Var}_{\sim i}(Y|X_i))$
- Expected reduction in uncertainty after learning input i = $\mathbf{Var}(Y) - \mathbf{E}_i(\mathbf{Var}_{\sim i}(Y|X_i))$
  - Also equal to $\mathbf{Var}_i(\mathbf{E}_{\sim i}(Y|X_i))$, via law of total variance
- Normalizing by the output variance gives the **first-order sensitivity index**, $S_i = \mathbf{Var}_i(\mathbf{E}_{\sim i}(Y|X_i)) / \mathbf{Var}(Y)$
- Nested expectations calculated by sampling, or (sometimes) analytically with an emulator of $Y(X)$

- We can define similar indices for *interactions* between pairs of variables, S_{ij}
- The sum of first-order and interaction sensitivities is the **total sensitivity index**, $T_i = \mathbf{E}_{\sim i}(\mathbf{Var}_i(Y|X_{\sim i})) / \mathbf{Var}(Y)$
- A large first-order sensitivity means it would be valuable to reduce uncertainty in that variable
- A small total sensitivity means that variable's uncertainty is negligible (it does not influence output uncertainty either directly, or indirectly through its interactions with other variables)

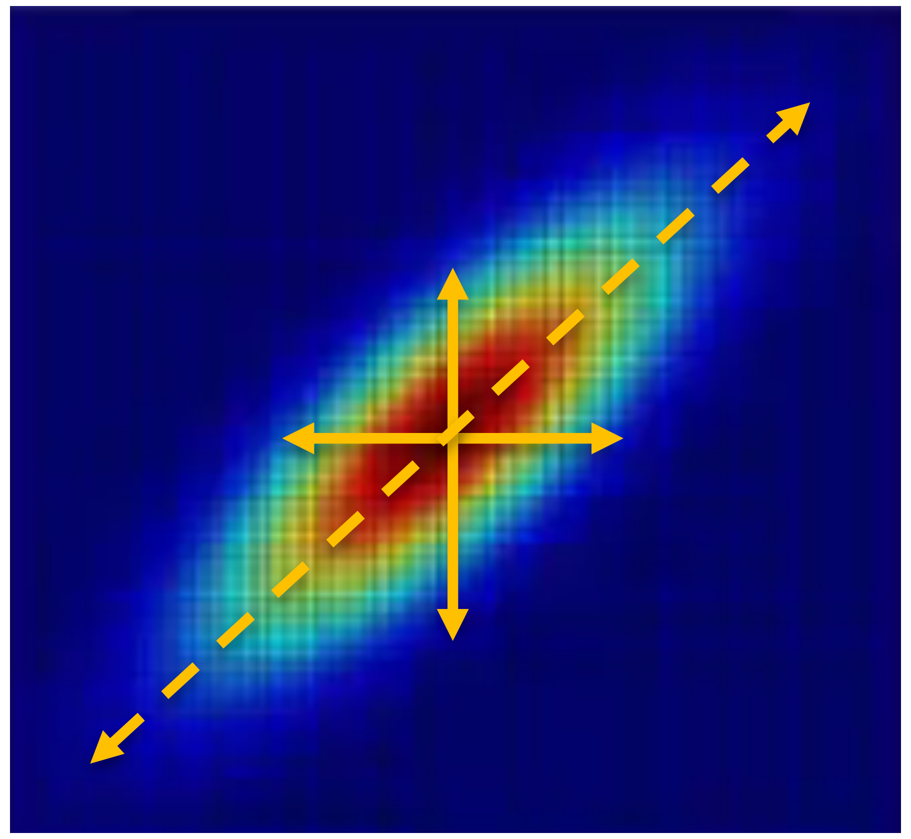
Code for global sensitivity analysis

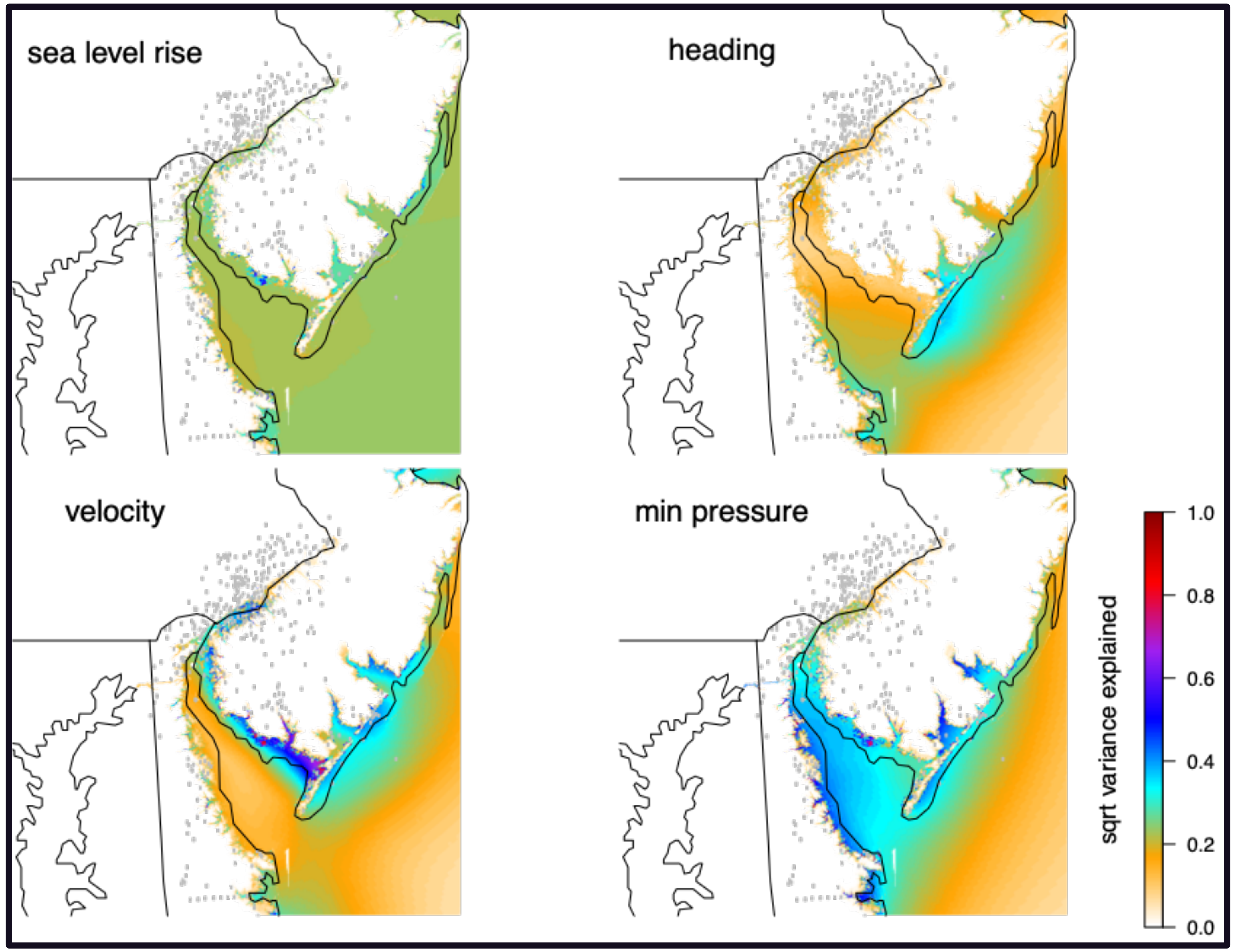
```julia
# conditional draw on xᵢ
randᵢ(d, i, xᵢ) = [j==i ? xᵢ : rand(d[j]) for j=1:length(d)]
# conditional draw on x₋ᵢ
rand!ᵢ(d, i, x!ᵢ) = [j==i ? rand(d[i]) : x!ᵢ[j] for j=1:length(d)]

# Sobol' first-order sensitivity index
S(m, d, i, N) = var(mean(m(randᵢ(d,i,xᵢ)) for k=1:N) for xᵢ in rand(d[i],N))
              / var(m(rand.(d)) for j=1:N^2)

# Sobol' total sensitivity index
T(m, d, i, N) = mean(var(m(rand!ᵢ(d,i,x!ᵢ)) for k=1:N) for x!ᵢ in (randᵢ(d,i,NaN) for j=1:N))
              / var(m(rand.(d)) for j=1:N^2)
```

julia

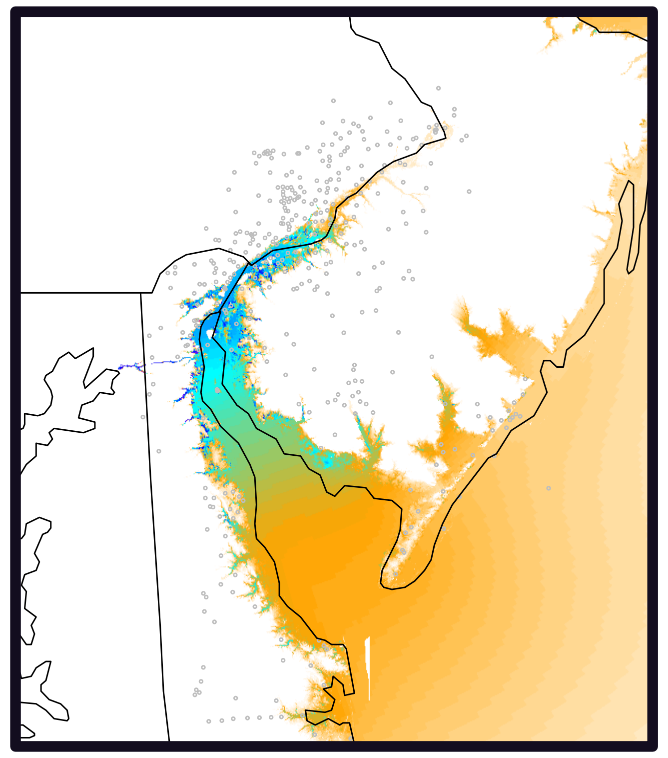
Global sensitivity analysis example

- Sensitivity of flooding to sea level rise and hurricane direction, speed, and intensity
- This does not mean these two inputs are correlated with each other (though they can be)
- Rather, nonlinear variations in the output may occur when two variables change together
- These effects would be invisible if the inputs were varied one-at-a-time





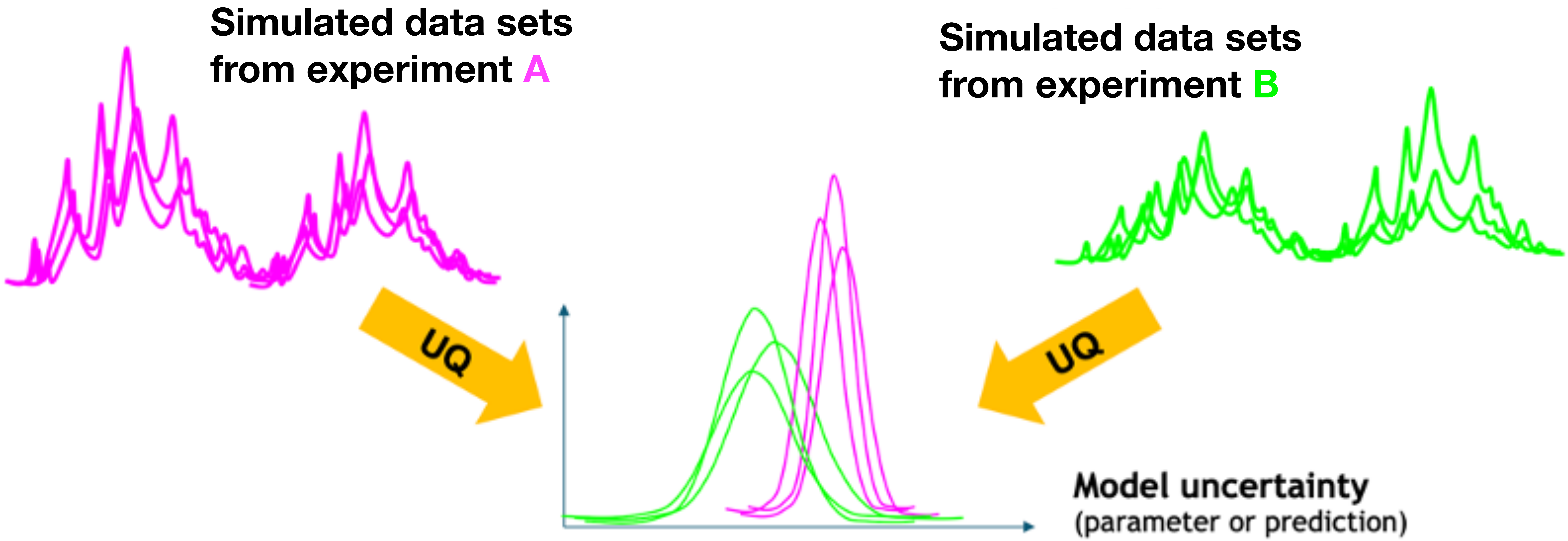
heading × velocity



Francom et al.
LA-UR-19-27244

Optimal experimental design

- Which experiments would give us the information we need to help us control the beam?
- Choose experiments whose data would reduce uncertainties the most?
  - Or rather, most reduce the objective to the stochastic optimal control problem



Optimal experimental design: Mathematics

- Uncertainty about parameter distribution $p(\theta)$ given by *entropy* $H[\theta] = \mathbb{E}_\theta[\log p(\theta)]$
- What experiment d would most reduce the entropy (maximize *information gain*)
  - Possible experimental outcomes are random, with probability distribution $p(y \quad \theta, d)$
  - Observing an outcome y gives a new distribution $p(\theta \quad y)$ with entropy $H[\theta \quad y]$.
  - We want to maximize information gain (entropy reduction) $H[\theta] - H[\theta \quad y]$
- The problem is, we don't know which outcome y we will measure
- Choose d to maximize *expected* information gain (EIG), averaged over possible outcomes
  - $EIG = \mathbb{E}_{y \quad \theta,d}\Big[H[\theta] - H[\theta \quad y]\Big]$

What next?

- We need to identify controls (and their ranges) that matter to the beam position
  - More expert elicitation, sensitivity analysis / parameter screening, …
- Perform UQ
  - Are results Gaussian? Correlated? May inform approximations we make in the future
- Stochastic optimization
  - Minimize expected loss via BFGS, gradient descent, BO, …
- Optimal experimental design
- How important are Bmad structural errors (biases, missing physics, …?)
  - Keep adding things to Bmad? Some other approach
- Sequential / realtime decision making?
  - Amortized myopic optimization (precompute policy: optimal solution conditional on state)
  - Reinforcement learning (accounting for future decisions in present actions)
    - RL with UQ: all state variables become *belief states* (infinite-dimensional distributions)