Probabilistic Models for PDEs with Random Coefficients

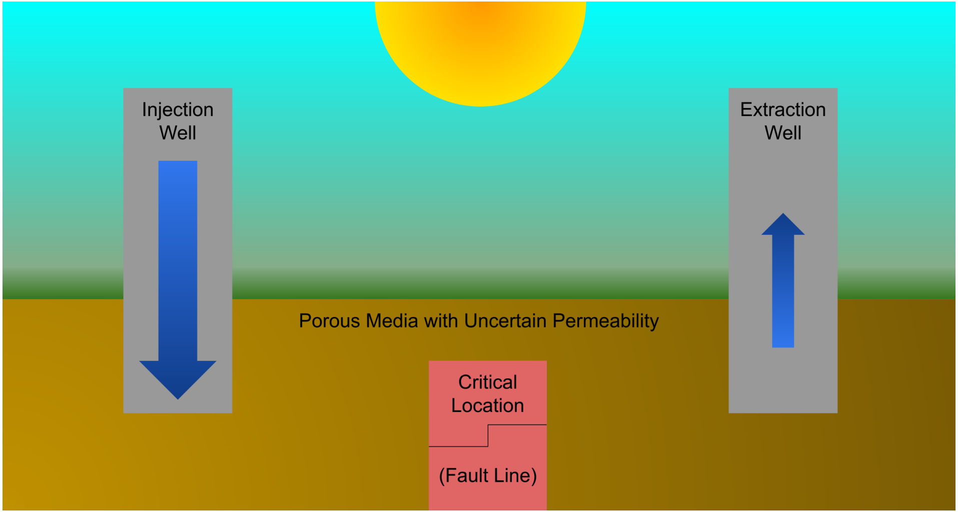
Aleksei G. Sorokin[1,2] Aleksandra Pachalieva[1] Dan O'Malley[1] Mac Hyman[1]
Fred J. Hickernell[2] Nick Hengartner[1]

[1]Los Alamos National Laboratory.
[2]Illinois Institute of Technology, Department of Applied Mathematics.

LA-UR 23-28111

Control Extraction Rate to Manage Pressure at Critical Location

Control Extraction Rate to Manage Pressure at Critical Location



Random Permiability in Porous Media Induces Random Pressure

[1] Aleksandra Pachalieva et al. "Physics-informed machine learning with differentiable programming for heterogeneous underground reservoir pressure management". In: *Scientific Reports* 12.1 (2022), p. 18734.

Random Permiability in Porous Media Induces Random Pressure

Darcy's equation $\nabla.(G(x).\nabla p(E,G,x)) = f(E,x)$ models pressure in porous media

- $G(x)$ a **random** *Gaussian permeability field*, known only by statistical properties
- $p(x)$ the **random** *pressure solution*

[1] Aleksandra Pachalieva et al. "Physics-informed machine learning with differentiable programming for heterogeneous underground reservoir pressure management". In: *Scientific Reports* 12.1 (2022), p. 18734.

Random Permiability in Porous Media Induces Random Pressure

Darcy's equation $\nabla.(G(x).\nabla p(E,G,x)) = f(E,x)$ models pressure in porous media

- $G(x)$ a **random** *Gaussian permeability field*, known only by statistical properties
- $p(x)$ the **random** *pressure solution*

Following Pachalieva et al.[1], we model the *flow rate* as

$$f(x,E) = \begin{cases} I, & x = x_{\text{injection}} \\ -E, & x = x_{\text{extraction}} \\ 0, & \text{else} \end{cases}.$$

- I, the **fixed** *injection rate*
- E, the **variable** *extraction rate*

[1] Aleksandra Pachalieva et al. "Physics-informed machine learning with differentiable programming for heterogeneous underground reservoir pressure management". In: *Scientific Reports* 12.1 (2022), p. 18734.

Choose Extraction Rate E to Rarely Overpressurizes a Critical Location

Choose Extraction Rate E to Rarely Overpressurizes a Critical Location

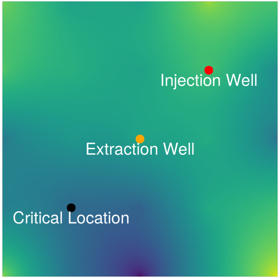
- Given critical location x_{critical} e.g. fault line
- Given *pressure threshold* $\bar{p}$ at the critical location
- $p^c(E, G) := p(E, G, x_{\text{critical}})$, the *critical pressure*

Want **online** choice of smallest E which gives high *confidence*

$$P(p^c(E,G) \leq \bar{p})$$

in keeping low enough pressure at the critical location

Birds-Eye View



Choose Extraction Rate E to Rarely Overpressurizes a Critical Location

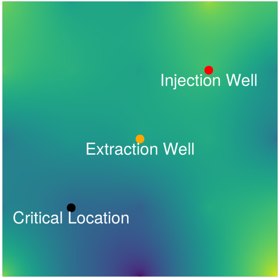
- Given critical location x_{critical} e.g. fault line
- Given *pressure threshold* $\bar{p}$ at the critical location
- $p^c(E, G) := p(E, G, x_{\text{critical}})$, the *critical pressure*

Want **online** choice of smallest E which gives high *confidence*

$$P(p^c(E, G) \leq \bar{p})$$

in keeping low enough pressure at the critical location

Birds-Eye View



Challenges

- Must approximate $P(p^c(E, G) \leq \bar{p})$ for many different E values
- Approximating $P(p^c(E, G) \leq \bar{p})$ requires many samples of $G(x)$ e.g. over 10,000

Represent $G(x)$ as an Infinite Sum, Approximate $G(x)$ by a Finite Sum

Represent $G(x)$ as an Infinite Sum, Approximate $G(x)$ by a Finite Sum

Karhunen-Loève series expansion of permeability field $G(x)$

$$G(x) = \sum_{j=1}^{\infty} \sqrt{\lambda_j} Z_j \varphi_j(x)$$

$$Z_1, Z_2, \ldots \overset{\text{IID}}{\sim} \mathcal{N}(0,1)$$

Represent $G(x)$ as an Infinite Sum, Approximate $G(x)$ by a Finite Sum

Karhunen-Loève series expansion of permeability field $G(x)$

$$G(x) = \sum_{j=1}^{\infty} \sqrt{\lambda_j} Z_j \varphi_j(x)$$

$$Z_1, Z_2, \ldots \overset{\text{IID}}{\sim} \mathcal{N}(0,1)$$

Since $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots$, we approximate $G(x)$ by a sum of s terms

$$G(x) \approx \sum_{j=1}^{s} \sqrt{\lambda_j} Z_j \varphi_j(x)$$

$$\boldsymbol{Z} = (Z_1, \ldots, Z_s)^T \sim \mathcal{N}(0, I_s)$$

Numerical Solution $p^c_{s,d}(E, \boldsymbol{Z})$ Approximates True Solution $p^c(E, G)$

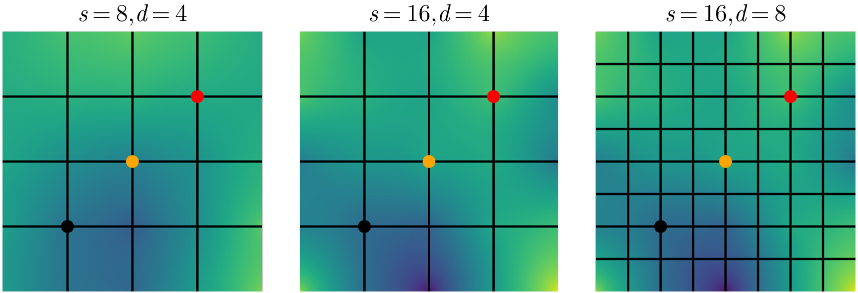
Numerical Solution $p^c_{s,d}(E, \boldsymbol{Z})$ Approximates True Solution $p^c(E, G)$

s, **KL Dimension:** $G(x) \approx \sum_{j=1}^{s} \sqrt{\lambda}_j Z_j \varphi_j(x)$ and $\boldsymbol{Z} = (Z_1, \ldots, Z_s)^T \sim \mathcal{N}(0, I_s)$

d, **Discretization Dimension:** Mesh width $1/d$ in each physical dimension

Numerical Solution $p^c_{s,d}(E,\boldsymbol{Z})$ Approximates True Solution $p^c(E,G)$

s, **KL Dimension:** $G(x) \approx \sum_{j=1}^{s} \sqrt{\lambda_j} Z_j \varphi_j(x)$ and $\boldsymbol{Z} = (Z_1, \dots, Z_s)^T \sim \mathcal{N}(0, I_s)$
d, **Discretization Dimension:** Mesh width $1/d$ in each physical dimension



Numerical solution uses DPFEHM: `https://github.com/OrchardLANL/DPFEHM.jl`

Gaussian Processes as Probabilistic Surrogates for p_c

Gaussian Processes as Probabilistic Surrogates for p_c

$$\text{Assume } p^c_{s,d} = p^c + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

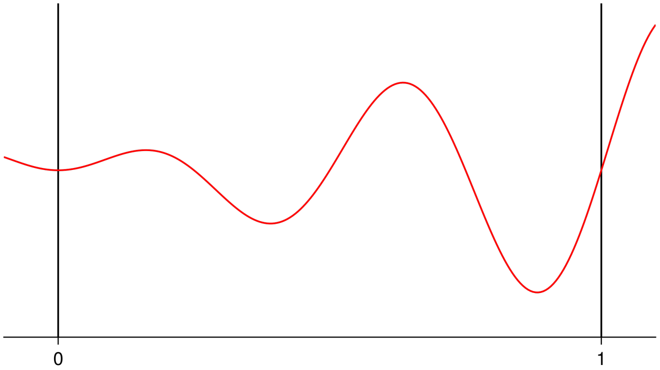
$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

Gaussian Processes as Probabilistic Surrogates for p_c

$$\text{Assume } p^c_{s,d} = p^c + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

solution p^c, numerical solutions, approximate solution, uncertainty in solution

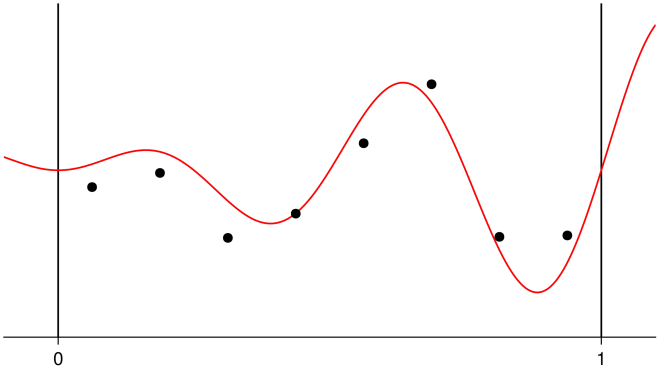


Gaussian Processes as Probabilistic Surrogates for p_c

$$\text{Assume } p^c_{s,d} = p^c + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

solution p^c, numerical solutions, approximate solution, uncertainty in solution

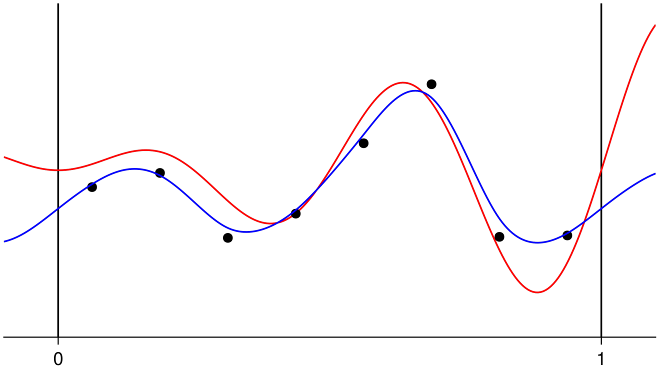


Gaussian Processes as Probabilistic Surrogates for p_c

$$\text{Assume } p^c_{s,d} = p^c + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

solution p^c, numerical solutions, approximate solution, uncertainty in solution

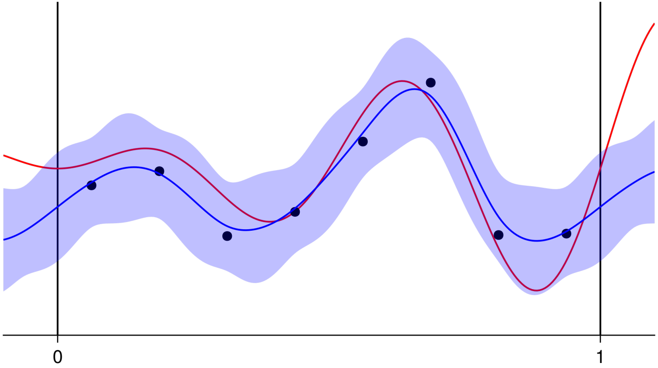


Gaussian Processes as Probabilistic Surrogates for p_c

$$\text{Assume } p^c_{s,d} = p^c + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

solution p^c, numerical solutions, approximate solution, uncertainty in solution



Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10{,}000$

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit
- Quasi-random nodes with matching kernels produces nice kernel matrices

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit
- Quasi-random nodes with matching kernels produces nice kernel matrices
  - Lattice sequence nodes + periodic shift invariant kernels $\rightarrow$ *circulant* matrices[2]

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit
- Quasi-random nodes with matching kernels produces nice kernel matrices
  - Lattice sequence nodes + periodic shift invariant kernels $\to$ *circulant* matrices[2]
  - Digital sequence nodes + digitally shift invariant kernels $\to$ *block Toeplitz* matrices[3]

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit
- Quasi-random nodes with matching kernels produces nice kernel matrices
  - Lattice sequence nodes + periodic shift invariant kernels $\rightarrow$ *circulant* matrices[2]
  - Digital sequence nodes + digitally shift invariant kernels $\rightarrow$ *block Toeplitz* matrices[3]
- Solving systems with these nice kernel matrices costs $\mathcal{O}(n \log n)$

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Structuring Nodes and Kernel Enables $\mathcal{O}(n \log n)$ Scaling for GPs

- Want GP fit to large number of samples nodes e.g. $n > 10,000$
- Classic GPs cost $\mathcal{O}(n^3)$ to fit
- Quasi-random nodes with matching kernels produces nice kernel matrices
  - Lattice sequence nodes + periodic shift invariant kernels $\to$ *circulant* matrices[2]
  - Digital sequence nodes + digitally shift invariant kernels $\to$ *block Toeplitz* matrices[3]
- Solving systems with these nice kernel matrices costs $\mathcal{O}(n \log n)$

$\therefore$ Can construct GPs with $\mathcal{O}(n \log n)$ cost when we have control over nodes details

[2] R. Jagadeeswaran and Fred J. Hickernell. "Fast automatic Bayesian cubature using lattice sampling". In: *Statistics and Computing* 29.6 (2019), pp. 1215–1229. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09895-9. URL: http://dx.doi.org/10.1007/s11222-019-09895-9.

[3] Rathinavel Jagadeeswaran and Fred J Hickernell. "Fast Automatic Bayesian Cubature Using Sobol Sampling". In: *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*. Springer, 2022, pp. 301–318.

Calibrate Gaussian Process Noise Variance $\zeta_{s,d}$ to Match Numerical Error

Calibrate Gaussian Process Noise Variance $\zeta_{s,d}$ to Match Numerical Error

- Approximate upper bound $\bar{\zeta}_{s,d}$ by tracking convergence as fidelity increases ▸ details

Calibrate Gaussian Process Noise Variance $\zeta_{s,d}$ to Match Numerical Error

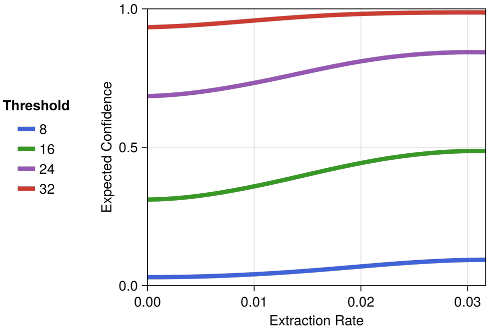
- Approximate upper bound $\bar{\zeta}_{s,d}$ by tracking convergence as fidelity increases details
- Decrease $\zeta_{s,d}$ starting at $\bar{\zeta}_{s,d}$ to optimize Gaussian process likelihood

Calibrate Gaussian Process Noise Variance $\zeta_{s,d}$ to Match Numerical Error

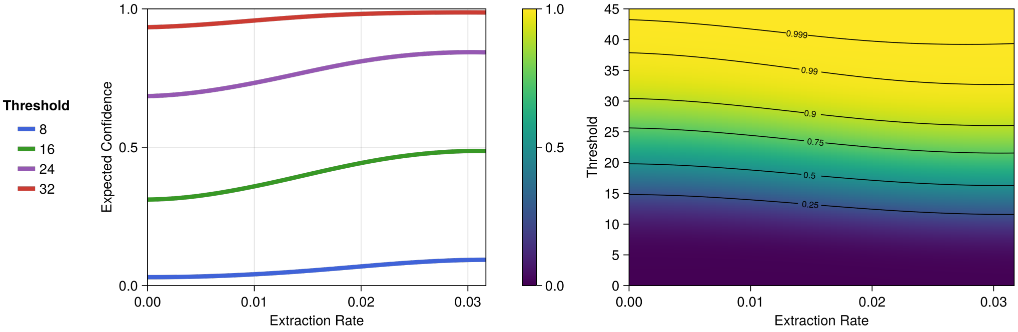
- Approximate upper bound $\bar{\zeta}_{s,d}$ by tracking convergence as fidelity increases ▸ details
- Decrease $\zeta_{s,d}$ starting at $\bar{\zeta}_{s,d}$ to optimize Gaussian process likelihood
- Hyperparameter optimization also costs $\mathcal{O}(n \log n)$ for specially structured GPs

Estimate $P(p^c(E,G) \leq \bar{p})$ Online with Gaussian Process Surrogate

Estimate $P(p^c(E,G) \leq \bar{p})$ Online with Gaussian Process Surrogate



Estimate $P(p^c(E, G) \leq \bar{p})$ Online with Gaussian Process Surrogate



We Fit a Probabilistic Model to a PDE with Random Coefficients

We Fit a Probabilistic Model to a PDE with Random Coefficients

- **Efficient:** Gaussian process construction in $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^3)$ cost

We Fit a Probabilistic Model to a PDE with Random Coefficients

- **Efficient:** Gaussian process construction in $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^3)$ cost
- **Error Aware:** Gaussian process noise calibrated to numerical error

We Fit a Probabilistic Model to a PDE with Random Coefficients

- **Efficient:** Gaussian process construction in $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^3)$ cost
- **Error Aware:** Gaussian process noise calibrated to numerical error
- **Transferable:** Methodology applicable to other PDEs with random coefficients

Don't Approximate Confidence $P(p^c(E,G) \leq \bar{p})$ by $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$

Don't Approximate Confidence $P(p^c(E,G) \leq \bar{p})$ by $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$

- Estimating $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$ for many E requires large number of $\boldsymbol{Z}$ samples

Don't Approximate Confidence $P(p^c(E,G) \leq \bar{p})$ by $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$

- Estimating $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$ for many E requires large number of $\boldsymbol{Z}$ samples
- Numerical solutions of $p^c_{s,d}(E,\boldsymbol{Z})$ are too expensive to compute online

Don't Approximate Confidence $P(p^c(E,G) \leq \bar{p})$ by $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$

- Estimating $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p})$ for many E requires large number of $\boldsymbol{Z}$ samples
- Numerical solutions of $p^c_{s,d}(E,\boldsymbol{Z})$ are too expensive to compute online
- $P(p^c_{s,d}(E,\boldsymbol{Z}) \leq \bar{p}) \neq P(p^c(E,G) \leq \bar{p})$ generally

Use Numerical Solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ to Build a Model for $p^c(E,G)$

Use Numerical Solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ to Build a Model for $p^c(E, G)$

- Can choose sampling nodes $(E_i, \boldsymbol{Z}_i)_{i=1}^n$ to explore the space well

Use Numerical Solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ to Build a Model for $p^c(E, G)$

- Can choose sampling nodes $(E_i, \boldsymbol{Z}_i)_{i=1}^n$ to explore the space well
- Numerical solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ can be computed offline

Use Numerical Solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ to Build a Model for $p^c(E, G)$

- Can choose sampling nodes $(E_i, \boldsymbol{Z}_i)_{i=1}^n$ to explore the space well
- Numerical solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ can be computed offline
- Surrogate model can be fit offline and evaluated quickly online

Use Numerical Solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ to Build a Model for $p^c(E, G)$

- Can choose sampling nodes $(E_i, \boldsymbol{Z}_i)_{i=1}^n$ to explore the space well
- Numerical solutions $\{p^c_{s,d}(E_i, \boldsymbol{Z}_i)\}_{i=1}^n$ can be computed offline
- Surrogate model can be fit offline and evaluated quickly online

$\therefore$ Inexpensive to estimate confidence $P(p^c(E,G) \leq \bar{p})$ online using surrogate

Noisy *Quasi*-Gaussian Process Surrogate

Assume $p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$

numerical solution $=$ solution $+$ Gaussian noise

Noisy *Quasi*-Gaussian Process Surrogate

$$\text{Assume } p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

Gaussian Process Regression

1. Prior mean $\mathbb{E}[p^c(t)] = 0$ and covariance $k(t,t') := \mathrm{Cov}[p^c(t), p^c(t')] = k(t,t')$.

Noisy *Quasi*-Gaussian Process Surrogate

Assume $p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$

numerical solution $=$ solution $+$ Gaussian noise

Gaussian Process Regression

1. Prior mean $\mathbb{E}[p^c(t)] = 0$ and covariance $k(t,t') := \mathrm{Cov}[p^c(t), p^c(t')] = k(t,t')$.
2. Observe $Y = \left(p^c_{s,d}(t_i)\right)_{i=1}^{n}$

Noisy *Quasi*-Gaussian Process Surrogate

Assume $p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$

numerical solution = solution + Gaussian noise

Gaussian Process Regression

1. Prior mean $\mathbb{E}[p^c(t)] = 0$ and covariance $k(t,t') := \mathrm{Cov}[p^c(t), p^c(t')] = k(t,t')$.
2. Observe $Y = \left(p^c_{s,d}(t_i)\right)_{i=1}^n$
3. Posterior mean $m_n(t) = \mathbb{E}[p^c(t)|Y]$ and covariance $k_n(t,t') = \mathrm{Cov}[p(t), p(t')|Y]$.

$$m_n(t) = K_T(t)\,[K_{TT} + \zeta I_n]^{-1}\,Y$$

costs $\mathcal{O}(n^3)$ where $K_T(t) = (k(t,t_i))_{i=1}^n\,, K_{TT} = (k(t_i,t_j))_{i,j=1}^n$.

Noisy *Quasi*-Gaussian Process Surrogate

$$\text{Assume } p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

Gaussian Process Regression

1. Prior mean $\mathbb{E}[p^c(t)] = 0$ and covariance $k(t,t') := \mathrm{Cov}[p^c(t), p^c(t')] = k(t,t')$.
2. Observe $Y = \left(p^c_{s,d}(t_i)\right)_{i=1}^n$
3. Posterior mean $m_n(t) = \mathbb{E}[p^c(t)|Y]$ and covariance $k_n(t,t') = \mathrm{Cov}[p(t), p(t')|Y]$.

$$m_n(t) = K_T(t) \left[K_{TT} + \zeta I_n\right]^{-1} Y$$

costs $\mathcal{O}(n^3)$ where $K_T(t) = (k(t,t_i))_{i=1}^n, K_{TT} = (k(t_i,t_j))_{i,j=1}^n$.

Noisy *Quasi*-Gaussian Process Surrogate

$$\text{Assume } p^c_{s,d}(t) = p^c(t) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \zeta_{s,d})$$

$$\text{numerical solution} = \text{solution} + \text{Gaussian noise}$$

Gaussian Process Regression

1. Prior mean $\mathbb{E}[p^c(t)] = 0$ and covariance $k(t,t') := \mathrm{Cov}[p^c(t), p^c(t')] = k(t,t')$.
2. Observe $Y = \left(p^c_{s,d}(t_i)\right)_{i=1}^n$
3. Posterior mean $m_n(t) = \mathbb{E}[p^c(t)|Y]$ and covariance $k_n(t,t') = \mathrm{Cov}[p(t), p(t')|Y]$.

$$m_n(t) = K_T(t)\,[K_{TT} + \zeta I_n]^{-1}\,Y$$

costs $\mathcal{O}(n^3)$ where $K_T(t) = (k(t,t_i))_{i=1}^n, K_{TT} = (k(t_i,t_j))_{i,j=1}^n$.

Quasi-Gaussian Process Regression

Choose special $(t_i)_{i=1}^n$ and k so $[K_{TT} + \zeta I_n]^{-1} Y$ costs $\mathcal{O}(n \log n)$.

Estimating Bound on Standard Deviation of GP Noise

$$\sqrt{\zeta_{s_T,d_T}} = \|p^c - p^c_{s_T,d_T}\| = \sqrt{\mathbb{E}\left[p^c(t) - p^c_{s_T,d_T}(t)\right]^2}, \quad \forall t$$

Rewrite as telescoping sum

$$\sqrt{\zeta_{s_T,d_T}} := \|p^c - p^c_{s_T,d_T}\| \le \sum_{j=T}^{\infty} \|\underbrace{p^c_{s_{j+1},d_j} - p^c_{s_j,d_j}}_{\Delta_{s_{j+1}}}\| + \sum_{j=T}^{\infty} \|\underbrace{p^c_{s_{j+1},d_{j+1}} - p^c_{s_{j+1},d_j}}_{\Delta_{d_{j+1}}}\|.$$

Models $\|\Delta_{s_j}\| = 2^{b_s} s_j^{a_s}$ and $\|\Delta_{d_j}\| = 2^{b_d} d_j^{a_d}$ with $s_j = v_s 2^j$ and $d_j = v_d 2^j$ imply

$$\sqrt{\zeta_{s_T,d_T}} \le \sum_{j=T+1}^{\infty} 2^{b_s} s_j^{a_s} + \sum_{j=T+1}^{\infty} 2^{b_d} d_j^{a_d} = 2^{b_s} v_s^{a_s} \frac{2^{(T+1)a_s}}{1-2^{a_s}} + 2^{b_d} v_d^{a_d} \frac{2^{(T+1)a_d}}{1-2^{a_d}} := \sqrt{\bar{\zeta}_{s_T,d_T}}.$$

Simple Linear Regression Convergence Models