UQ Workshop and UncertainSCI software

Jake Bergquist[1,3] Dana Brooks[4] Chantel Charlebois[1,3] Zexin Liu[2,3]
Rob MacLeod[1,3] Akil Narayan[2,3] Sumientra Rampersad[4]
Lindsay Rupp [1,3] Jess Tate[3] Dan White [3]

[1] Department of Biomedical Engineering
University of Utah

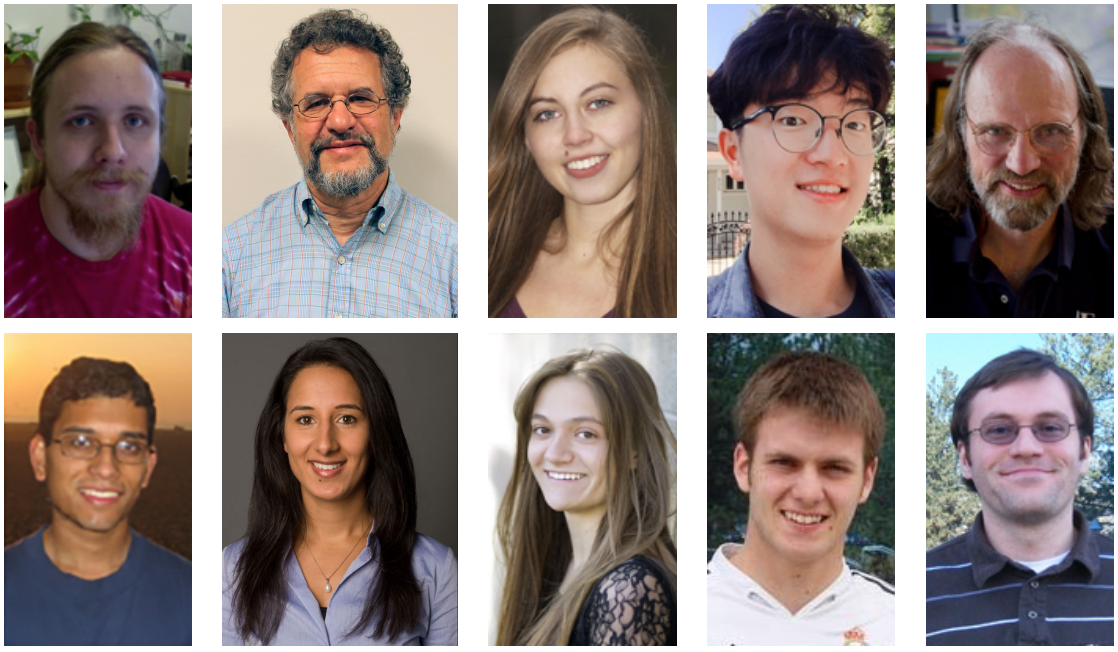
[2] Department of Mathematics
University of Utah

[3] Scientific Computing and Imaging (SCI) Institute,
University of Utah

[4] Electrical and Computer Engineering,
Northeastern University

June 25, 2021
FIMH 2021
Supported by NIH U24-EB029012

Workshop team



Software: `https://github.com/SCIInstitute/UncertainSCI/releases/tag/0.1.0-beta`

Github discussion: `https://github.com/SCIInstitute/UncertainSCI/discussions/82`

Discord discussion: `https://discord.com/invite/MGEVK6K5`

Workshop goals

This workshop has two parts that explore two complementary themes.

Modeling parametric uncertainty

- UQ goals and desiderata
- Parametric uncertainty
- Polynomial Chaos

UQ in practice with UncertainSCI

- UncertainSCI software
- Cardiac bioelectricity use cases and applications
- Neuromodulation examples

Workshop agenda

Workshop overview, all times MT:

9:00 – 9:30	Overview and UQ introduction	Akil Narayan
9:30 – 10:00	Mathematics of polynomial Chaos	Akil Narayan
10:00 – 10:30	UncertainSCI software	Jess Tate, Jake Bergquist
10:30 – 11:00	Break	
11:00 – 11:30	Cardiac bioelectricity use case	Jess Tate, Jake Bergquist
11:30 – 12:00	Neuromodulation use case	Sumientra Rampersad
12:00 – 12:30	Breakout sessions	

Simulation models

Computational simulations are subject to parametric uncertainties,

- conductivities
- heart location, geometry

and also model uncertainties,

- model misspecification
- simplified mathematical equations
- computational/discretization error

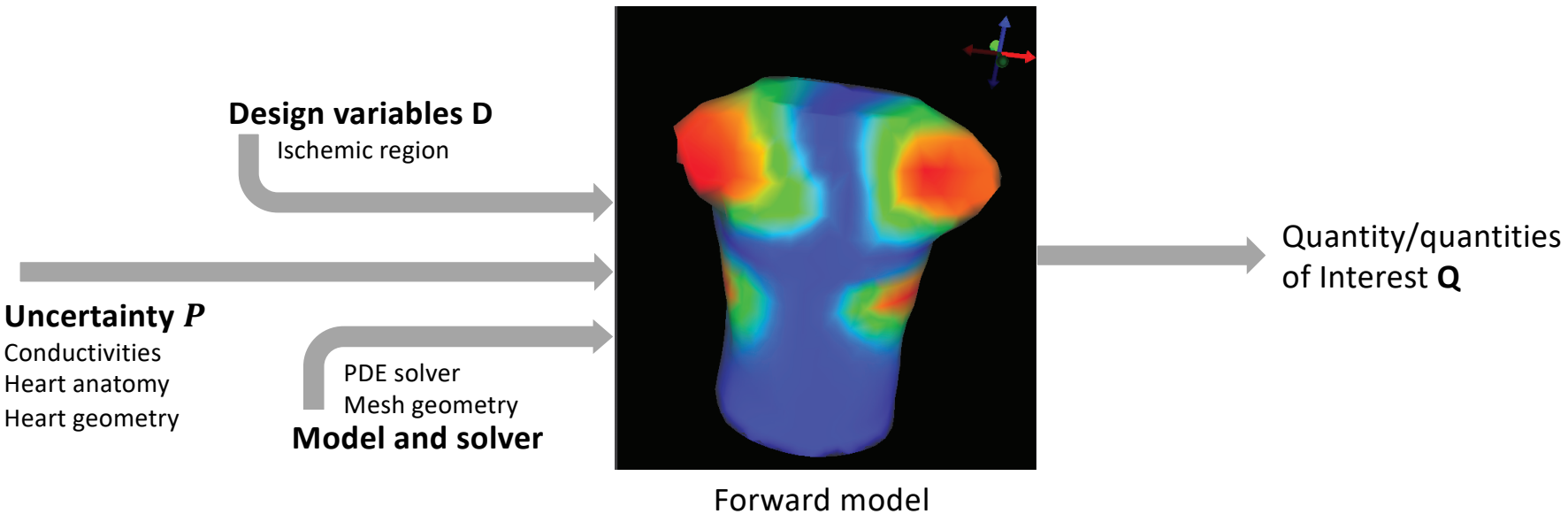
Parametric uncertainty can typically be modeled and interpreted meaningfully.

Model uncertainty: problem-specific and more nebulous

Uncertainty in models

Parametric uncertainty requires modeling

- probability densities for scalars
- parameterized geometry

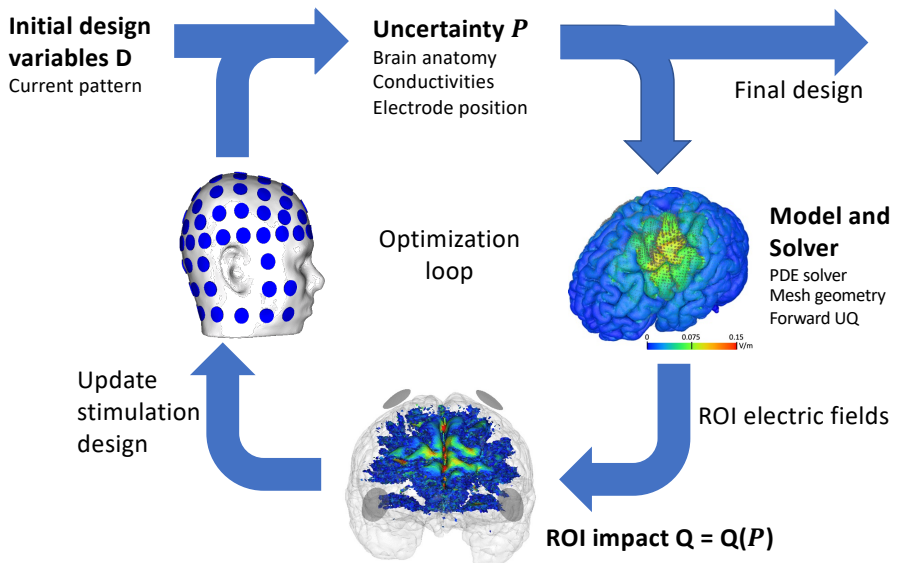


Output quantities of interest Q depend on parameterized uncertainty.

Uncertainty quantification

With a model of stochasticity, there can be several goals for UQ:

- Forward propagation of uncertainty
  - statistics of quantities of interest
  - sensitivity analysis
  - parameter screening or reduction
- Parameter estimation (typically with data)
  - inverse/inference problem built on forward simulations
  - identification of experimentally unobservable quantities
- Design and performance optimization
  - outer-loop optimization on design variables
  - computation of designs that are robust to uncertainty



UQ setup

There are 3 ingredients required to set up any of these UQ problems:

- Identification of parameters P
- Probabilistic modeling (specifying a distribution) for P
- Definition of an output quantity of interest

UQ setup

There are 3 ingredients required to set up any of these UQ problems:

- **Identification of parameters P**
- Probabilistic modeling (specifying a distribution) for P
- Definition of an output quantity of interest

What is uncertain in my model? How can I parameterize this uncertainty?

- Finite-dimensional parameters (bidomain conductivities)
- Stochastic fields (conductivity fields)
- Geometric uncertainty (cohort shape variability)

These can all be meaningfully modeled as a *finite-dimensional* parameter $P \in \mathbb{R}^d$.

Underparameterization (small d) can yield a poor model of uncertainty.

Overparameterization (large d) makes it difficult to explore uncertainty.

Cf. use cases later today!

UQ setup

There are 3 ingredients required to set up any of these UQ problems:

- Identification of parameters P
- **Probabilistic modeling (specifying a distribution) for P**
- Definition of an output quantity of interest

What kinds of values are reasonable for $P = (P_1, \ldots, P_d)$ to take?

Are some parameters coupled? Is P_j independent of P_k?

A quantifiable way to describe these considerations is through probabilistic modeling:
Let $w : \mathbb{R}^d \to [0, \infty)$ be a probability density function for P.
This in particular defines the range of values that P can take (the support of w).

A common assumption is that all parameters are independent. In this case,

$$w(p) = w_1(p_1) \cdots w_d(p_d), \qquad p \in \mathbb{R}^d.$$

This results in substantial simplification of algorithms.

UQ setup

There are 3 ingredients required to set up any of these UQ problems:

- Identification of parameters P
- Probabilistic modeling (specifying a distribution) for P
- **Definition of an output quantity of interest**

For each fixed parameter value a forward simulation yields an output quantity of interest:

$$P \xrightarrow{\text{Forward simulation}} u(P) \xrightarrow{\text{Restriction, averaging, etc}} Q(u(P))$$

For exmaple, $u(P)$ can be the output of a(n expensive!) PDE forward model for bioelectric propagation.

Q represents a summarized output (e.g., localized epicardial potential)

In forward UQ analysis, we seek to understand the map $P \mapsto Q(u(P))$.

Surrogates and emulators

A popular technique for accelerating forward UQ analysis: emulators.

$$Q(P) \approx Q_N(P)$$

Q_N is a trained computational emulator that is efficient and ideally accurate.

Surrogates and emulators

A popular technique for accelerating forward UQ analysis: emulators.

$$Q(P) \approx Q_N(P)$$

Q_N is a trained computational emulator that is efficient and ideally accurate.

There are two (frequently) overlapping strategies:

- *linear* methods: simple, direct, well-understood accuracy

$$Q_N(P) = \sum_{j=1}^{N} \widehat{q}_j \phi_j(P),$$

  where ϕ_j are prescribed functions.
  - Stochastic finite element methods
  - some Polynomial chaos (PC) methods
- *nonlinear* methods: more expressive, but also more "finicky" and opaque
  - other Polynomial chaos methods
  - Gaussian processes
  - Neural networks

In UncertainSCI we use linear PC emulators.

Forward UQ analysis

After an emulator is built, UQ analysis is an efficient post-processing step.

The following can be efficiently approximated componentwise for Q_N:

- Median, quantiles, confidence intervals
- Statistics (mean, variance, etc.)
- Partial variances: let T denote a subset of $\{1, \ldots, d\}$
  - *Global* variance: $\mathrm{var}_T(Q_N) = \mathrm{var}\left(\mathbb{E}[Q_N(P) \mid P_T]\right)$
    Measures the variance due to "genuine" interactions among variables in subset P_T.
  - *Total* variance: $\mathrm{var}_T^{\mathrm{tot}}(Q_N) = \sum_{U \subset T} \mathrm{var}_U(Q_N)$
    Measures the variance due to variable subset P_T.

Forward UQ analysis

After an emulator is built, UQ analysis is an efficient post-processing step.

The following can be efficiently approximated componentwise for Q_N:

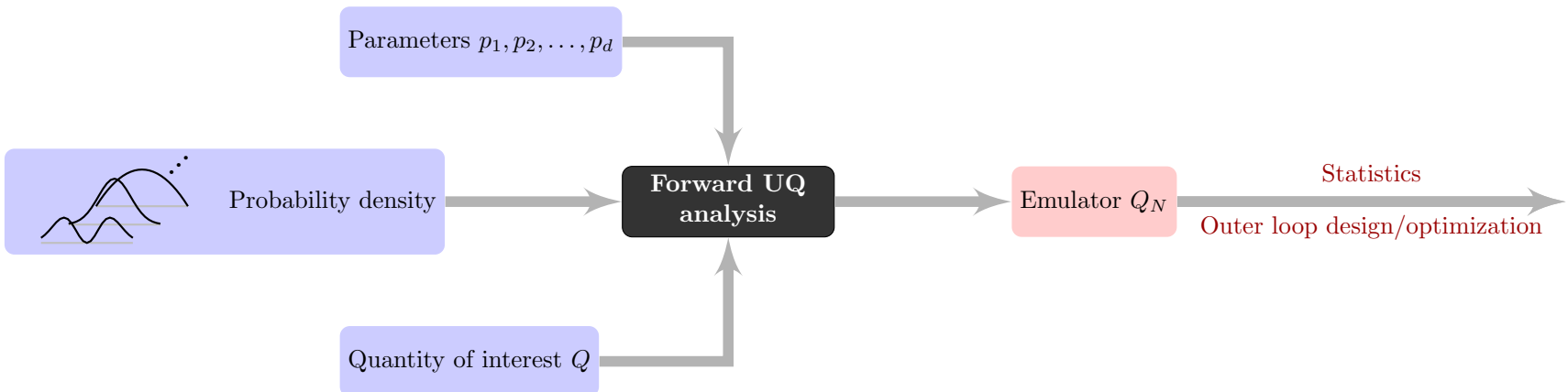
- Median, quantiles, confidence intervals
- Statistics (mean, variance, etc.)
- Partial variances: let T denote a subset of $\{1, \ldots, d\}$
  - *Global* variance: $\mathrm{var}_T(Q_N) = \mathrm{var}\left(\mathbb{E}[Q_N(P) \mid P_T]\right)$
    Measures the variance due to "genuine" interactions among variables in subset P_T.
  - *Total* variance: $\mathrm{var}_T^{\mathrm{tot}}(Q_N) = \sum_{U \subset T} \mathrm{var}_U(Q_N)$
    Measures the variance due to variable subset P_T.
- Sensitivities
  - *Global* sensitivities: $S_T = \frac{\mathrm{var}_T(Q_N)}{\mathrm{var}(Q_N)} \leqslant 1$
    Measures the relative importance of "genuine" interactions in variable subset P_T.
  - *Total* sensitivities: $S_T^{\mathrm{tot}} = \frac{\mathrm{var}_T^{\mathrm{tot}}(Q_N)}{\mathrm{var}(Q_N)} \leqslant 1$
    Measures the relative importance of variable subset P_T.

Note: these are approximations since $Q_N \approx Q$.

Summary

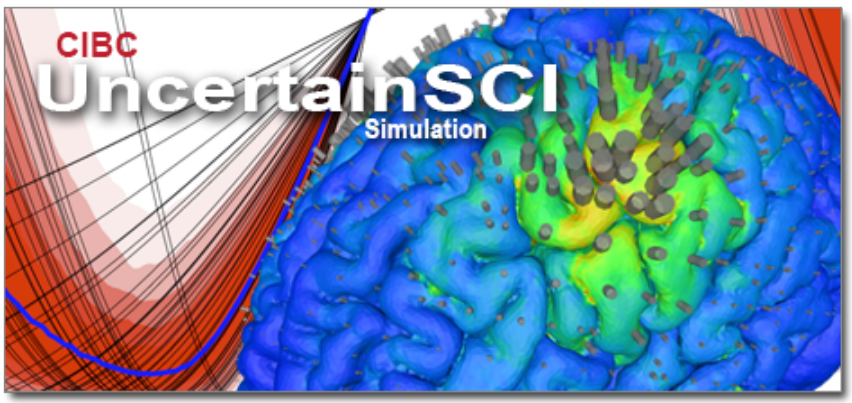
To model forward uncertainty with emulators, we require

- identification of a d-dimensional random parameter P
- modeling of likely values of P through a density w
- definition of a forward simulation output, a quantity of interest $Q(P)$



UncertainSCI

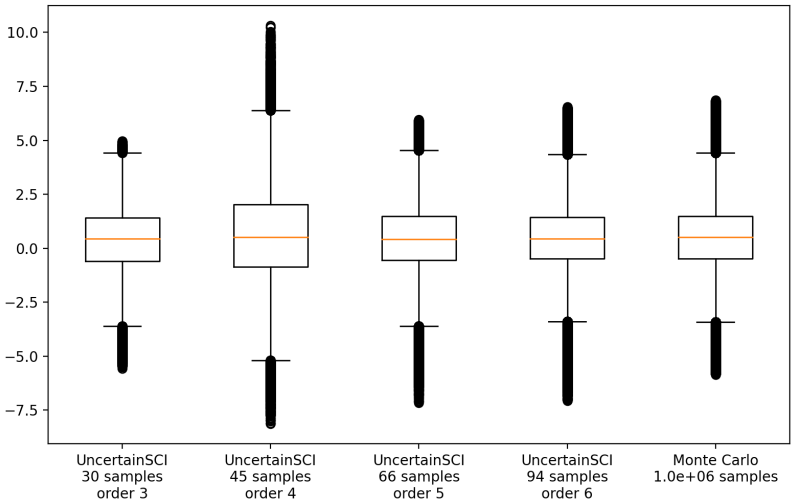
The software package we have built and use: UncertainSCI

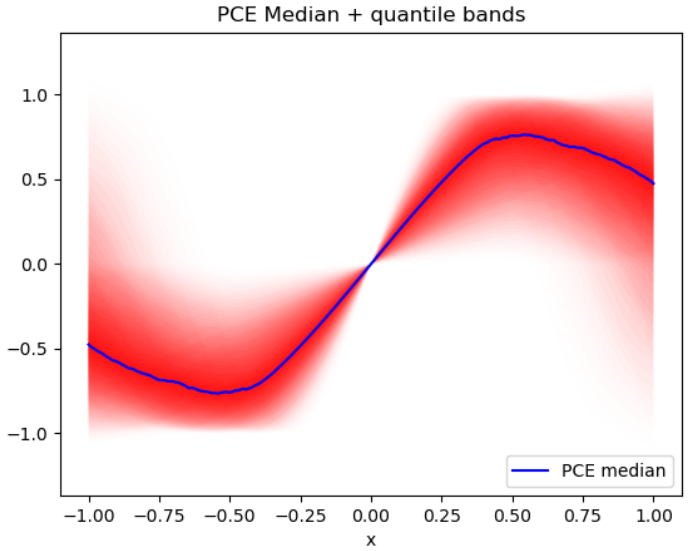


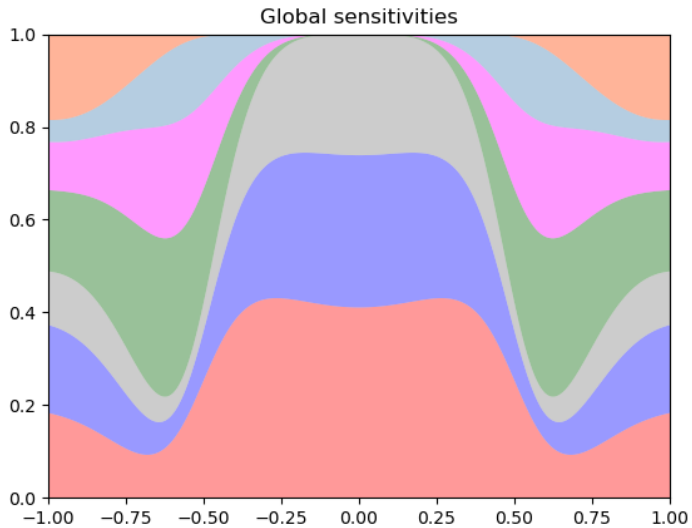
- open-source Python software
- forward UQ analysis
- polynomial Chaos-based

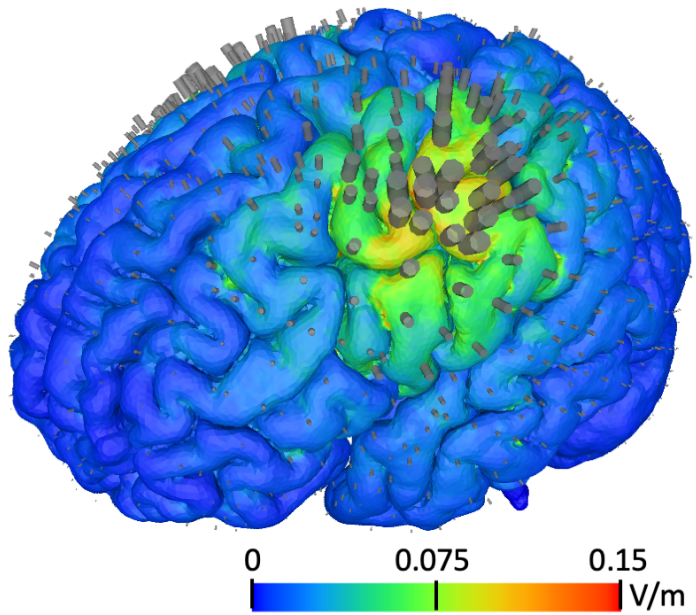
`https://www.sci.utah.edu/sci-software/simulation/uncertainsci.html`
(`http://bit.ly/uncertainsci`)

UncertainSCI capabilities









Take 5

Polynomial chaos (PC)

Recall:

- $P \in \mathbb{R}^d$ is a random variable with probability density w
- $Q(P)$ is a quantity of interest from a forward simulation
- $Q_N(P)$ is an emulator

PC approaches construct the emulator

$$Q(P) \approx Q_N(P) := \sum_{j=1}^{N} \widehat{q}_j \phi_j(P),$$

Polynomial chaos (PC)

Recall:

- $P \in \mathbb{R}^d$ is a random variable with probability density w
- $Q(P)$ is a quantity of interest from a forward simulation
- $Q_N(P)$ is an emulator

PC approaches construct the emulator

$$Q(P) \approx Q_N(P) := \sum_{j=1}^{N} \widehat{q}_j \phi_j(P),$$

The functions ϕ_j are multivariate polynomials spanning a particular space.

Polynomial chaos (PC)

Recall:

- $P \in \mathbb{R}^d$ is a random variable with probability density w
- $Q(P)$ is a quantity of interest from a forward simulation
- $Q_N(P)$ is an emulator

PC approaches construct the emulator

$$Q(P) \approx Q_N(P) := \sum_{j=1}^{N} \widehat{q}_j \phi_j(P),$$

The functions ϕ_j are multivariate polynomials spanning a particular space.

The coefficients $\widehat{q}_j$ are learned by training:

- Intrusive methods: Compute $\widehat{q}_j$ by "opening up", and possibly manipulating, the forward solver Q
- Non-intrusive methods: Compute $\widehat{q}_j$ using black-box data $\{(p_m, Q(p_m))\}_{m=1}^{M}$

We will focus on the non-intrusive case.

Polynomial spaces

The multivariate polynomials ϕ_j are a basis for a dimension-N polynomial subspace.

The choice of polynomial space identifies Q_N's capacity, complexity, and expressivity.

Large N increases model capacity, but makes training more expensive

For independent parameters, the polynomial basis functions take the form[1],

$$\phi_j(p) = \prod_{q=1}^{d} p_q^{\lambda_j^{(q)}}, \qquad \lambda_j = \left(\lambda_j^{(1)}, \lambda_j^{(2)}, \ldots, \lambda_j^{(d)}\right) \in \mathbb{N}_0^d, \qquad \Lambda = \{\lambda_j\}_{j=1}^{N}.$$

We denote the polynomial space defined by Λ as $V(\Lambda)$.

[1] For numerical stability we actually use orthonormal polynomials, not monomials.

Polynomial spaces

The multivariate polynomials ϕ_j are a basis for a dimension-N polynomial subspace.

The choice of polynomial space identifies Q_N's capacity, complexity, and expressivity.

Large N increases model capacity, but makes training more expensive

For independent parameters, the polynomial basis functions take the form[1],

$$\phi_j(p) = \prod_{q=1}^{d} p_q^{\lambda_j^{(q)}}, \qquad \lambda_j = \left(\lambda_j^{(1)}, \lambda_j^{(2)}, \ldots, \lambda_j^{(d)}\right) \in \mathbb{N}_0^d, \qquad \Lambda = \{\lambda_j\}_{j=1}^{N}.$$

We denote the polynomial space defined by Λ as $V(\Lambda)$.

Polynomial index sets Λ are identified by

- an *order* parameter k (similar to polynomial degree)
  - Large k allows elements λ of Λ to be "large" in magnitude
  - Large k can make N large due to interaction terms
- a prescription of how much parameters can interact
  - More interaction allows mixed terms $p_1^{\lambda^{(1)}} p_2^{\lambda^{(2)}}$ for "large" λ
  - More interaction terms: more model capacity, more training needed

[1] For numerical stability we actually use orthonormal polynomials, not monomials.

Polynomial spaces, cont.

Some $d = 2$ examples of order-k interactions between p_1 and p_2:

- "Hyperbolic cross" spaces Λ_{HC}: suppression of interactions

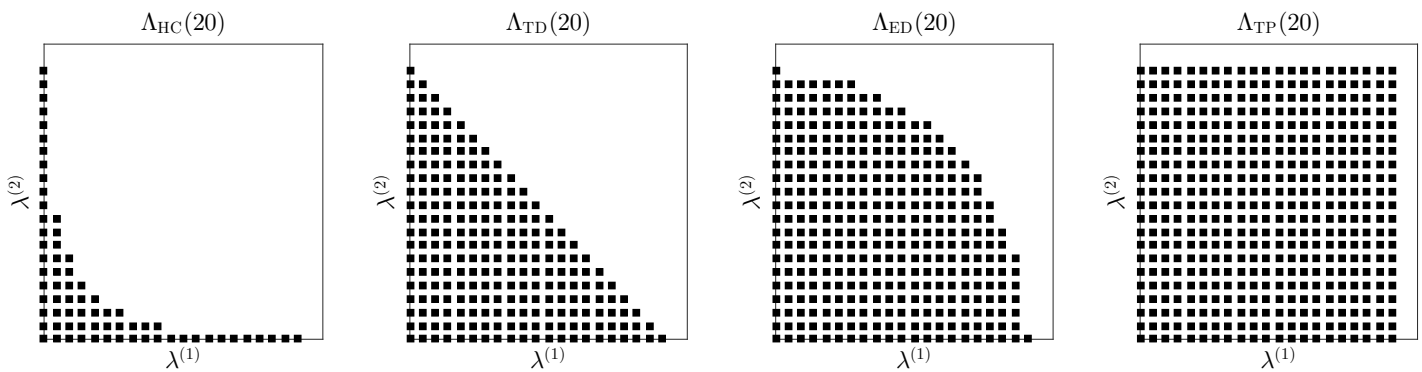
$$\phi_j(p) = p_1^{\lambda_j^{(1)}} p_2^{\lambda_j^{(2)}}, \qquad \log\left(\lambda_j^{(1)}\lambda_j^{(2)}\right) \leqslant \log(k+1)$$

- "Total degree" spaces Λ_{TD}: quite a few interactions

$$\phi_j(p) = p_1^{\lambda_j^{(1)}} p_2^{\lambda_j^{(2)}}, \qquad \lambda_j^{(1)} + \lambda_j^{(2)} \leqslant k.$$

- "Tensor product" spaces Λ_{TP}: lots of interactions

$$\phi_j(p) = p_1^{\lambda_j^{(1)}} p_2^{\lambda_j^{(2)}}, \qquad \lambda_j^{(1)} \leqslant k \text{ and } \lambda_j^{(2)} \leqslant k.$$



Polynomial spaces, cont.

Interactions can substantially increase model capacity $\rightarrow$ curse of dimensionality.

Index set sizes N for increasing dimension:

$(\boldsymbol{k} = \mathbf{2})$	Λ_{TP}	Λ_{TD}	Λ_{HC}
$d = 1$	3	3	3
$d = 2$	9	6	3
$d = 5$	243	21	4
$d = 8$	6,561	45	9
$d = 15$	14,348,907	136	16

Balancing richness of interactions with computational feasibility is a bit of an art.

Polynomial spaces, cont.

Interactions can substantially increase model capacity $\rightarrow$ curse of dimensionality.

Index set sizes N for increasing dimension:

$(\boldsymbol{k} = \mathbf{2})$	Λ_{TP}	Λ_{TD}	Λ_{HC}
$d = 1$	3	3	3
$d = 2$	9	6	3
$d = 5$	243	21	4
$d = 8$	6,561	45	9
$d = 15$	14,348,907	136	16

$(\boldsymbol{k} = \mathbf{7})$	Λ_{TP}	Λ_{TD}	Λ_{HC}
$d = 1$	8	8	8
$d = 2$	64	36	18
$d = 5$	32,768	792	91
$d = 8$	16,777,216	6435	245
$d = 15$	35,184,372,088,832	170,544	1071

Balancing richness of interactions with computational feasibility is a bit of an art.

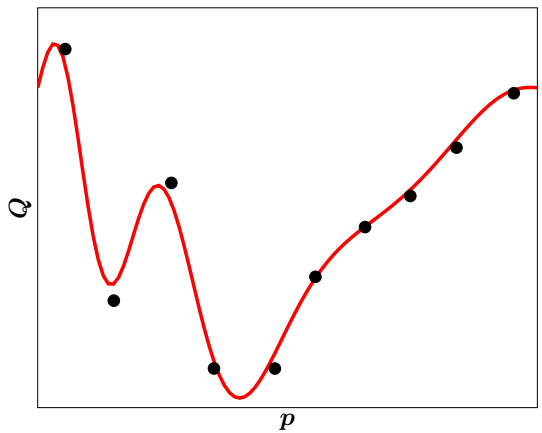
Training with least squares

We use non-intrusive PC construction with least squares: Enforce

$$Q(p_m) \approx Q_N(p_m) \qquad \longrightarrow \qquad Q(p_m) \approx \sum_{j=1}^{N} \widehat{q}_j \phi_j(p_m),$$

Given data $Q(p_m)$, we solve

$$\min_{\widehat{q}_j} \sum_{m=1}^{M} \left(Q(p_m) - Q_N(p_m)\right)^2 .$$



This is a polynomial fitting problem, though not necessarily a standard one.

$M \geqslant N$ is necessary to ensure a unique least squares solution.

Given Λ, how are the samples p_m chosen?

What does UncertainSCI do?

Once polynomial space / index set is chosen:

$$Q(P) \approx Q_N(P) := \sum_{j=1}^{N} \widehat{q}_j \phi_j(P) \in V(\Lambda),$$

UncertainSCI computes coefficients $\widehat{q}_j$ by

- solving a weighted least squares problem
- using data from a weighted D-optimal design that is optimized by induced measure sampling

What does UncertainSCI do?

Once polynomial space / index set is chosen:

$$Q(P) \approx Q_N(P) := \sum_{j=1}^{N} \widehat{q}_j \phi_j(P) \in V(\Lambda),$$

UncertainSCI computes coefficients $\widehat{q}_j$ by

- solving a weighted least squares problem
- using data from a weighted D-optimal design that is optimized by induced measure sampling

Construct Q_N by solving

$$\min_{Q_N \in V(\Lambda)} \sum_{m=1}^{M} \left(Q(p_m) - Q_N(p_m)\right)^2 .$$

How are samples p_m chosen? Ideally we want sampling to

- work in high dimensions with $M \sim N$
- not require independent parameters P

A simple idea is to use random ("Monte Carlo") sampling from the density w of P:

$$p_m \overset{\text{iid}}{\sim} w$$

How well does random sampling work?

Least squares: N unknowns, M data samples. Approximation with $M \sim N$ is optimal.

Near-optimal approximation can be achieved:

Theorem

Fix the distribution of P and Λ. There is a constant $C = C(\Lambda, w)$ such that if $M = C\ K\ N \log N$ samples are taken for any $K > 1$, then

$$\mathbb{E}_{P,p_m}\left[Q_N - Q\right]^2 \lesssim \epsilon_\Lambda(Q) + M^{-K/2}, \qquad \epsilon_\Lambda(Q) := \inf_{R \in V(\Lambda)} \mathbb{E}_P\left[R(P) - Q(P)\right]^2.$$

The quantity $\epsilon_\Lambda(Q)$ is the best possible emulator from the polynomial space defined by Λ.

How well does random sampling work?

Least squares: N unknowns, M data samples. Approximation with $M \sim N$ is optimal.

Near-optimal approximation can be achieved:

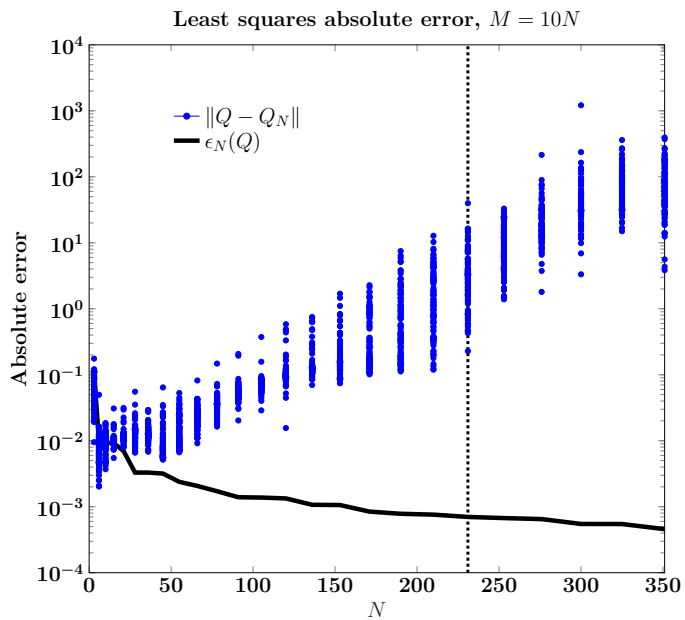
Theorem

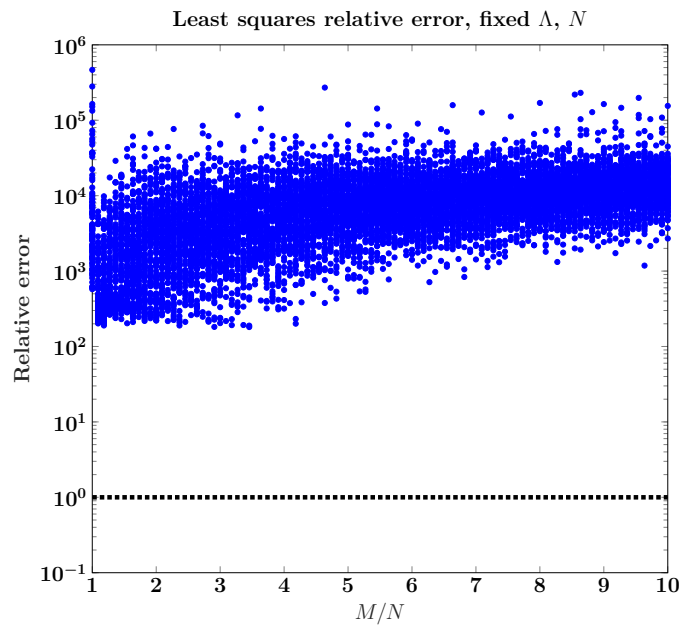
Fix the distribution of P and Λ. There is a constant $C = C(\Lambda, w)$ such that if $M = C\ K\ N \log N$ samples are taken for any $K > 1$, then

$$\mathbb{E}_{P,p_m}\left[Q_N - Q\right]^2 \lesssim \epsilon_\Lambda(Q) + M^{-K/2}, \qquad \epsilon_\Lambda(Q) := \inf_{R \in V(\Lambda)} \mathbb{E}_P\left[R(P) - Q(P)\right]^2.$$

The quantity $\epsilon_\Lambda(Q)$ is the best possible emulator from the polynomial space defined by Λ.

The problem: $C(\Lambda, w)$ can be huge, and it's easy to construct such an example:





The induced measure

The problem can be rectified by using weighted least squares + importance sampling:

Fixing (w, Λ), the *induced measure* for this pair corresponds to a density ρ given by,

$$\rho(p) := w(p) \sup_{R \in V(\Lambda)\setminus\{0\}} \frac{R^2(p)}{N\ \mathbb{E}_P R^2(P)}$$

The density ρ depends on w and Λ.

The induced measure

The problem can be rectified by using weighted least squares + importance sampling:

Fixing (w, Λ), the *induced measure* for this pair corresponds to a density ρ given by,

$$\rho(p) := w(p) \sup_{R \in V(\Lambda)\setminus\{0\}} \frac{R^2(p)}{N \, \mathbb{E}_P R^2(P)}$$

The density ρ depends on w and Λ.

We now perform weighted least squares: Sample

$$p_m \overset{\text{iid}}{\sim} \rho,$$

and compute

$$\min_{Q_N \in V(\Lambda)} \sum_{m=1}^{M} \frac{w(p_m)}{\rho(p_m)} \left(Q(p_m) - Q_N(p_m)\right)^2 .$$

Weighted least squares

Theorem

There is an absolute constant $c \sim 1$ such that, for any distribution of P and Λ if $M = (cK)N \log N$ samples from ρ are taken for any $K > 1$, then

$$\mathbb{E}_{P,p_m}\left[Q_N - Q\right]^2 \lesssim \epsilon_\Lambda(Q) + M^{-K/2}, \qquad \epsilon_\Lambda(Q) := \inf_{R \in V(\Lambda)} \mathbb{E}_P\left[R(P) - Q(P)\right]^2.$$

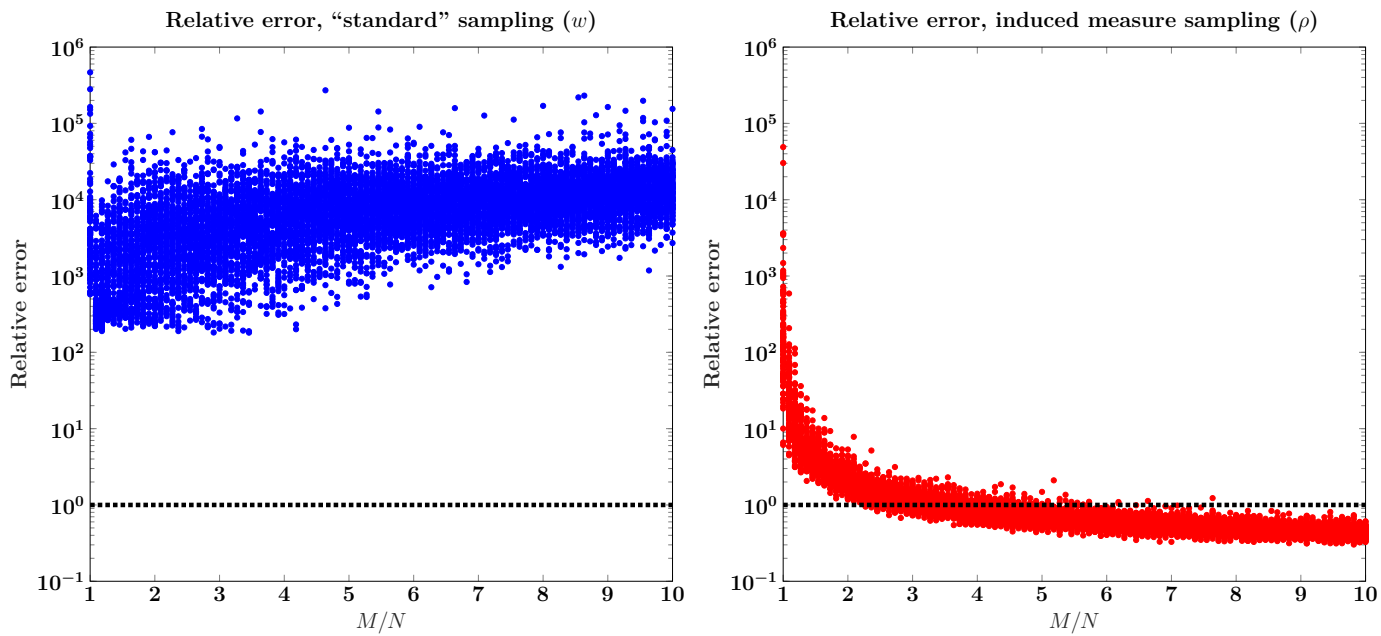
Weighted least squares

Theorem

There is an absolute constant $c \sim 1$ such that, for any distribution of P and Λ if $M = (cK)N \log N$ samples from ρ are taken for any $K > 1$, then

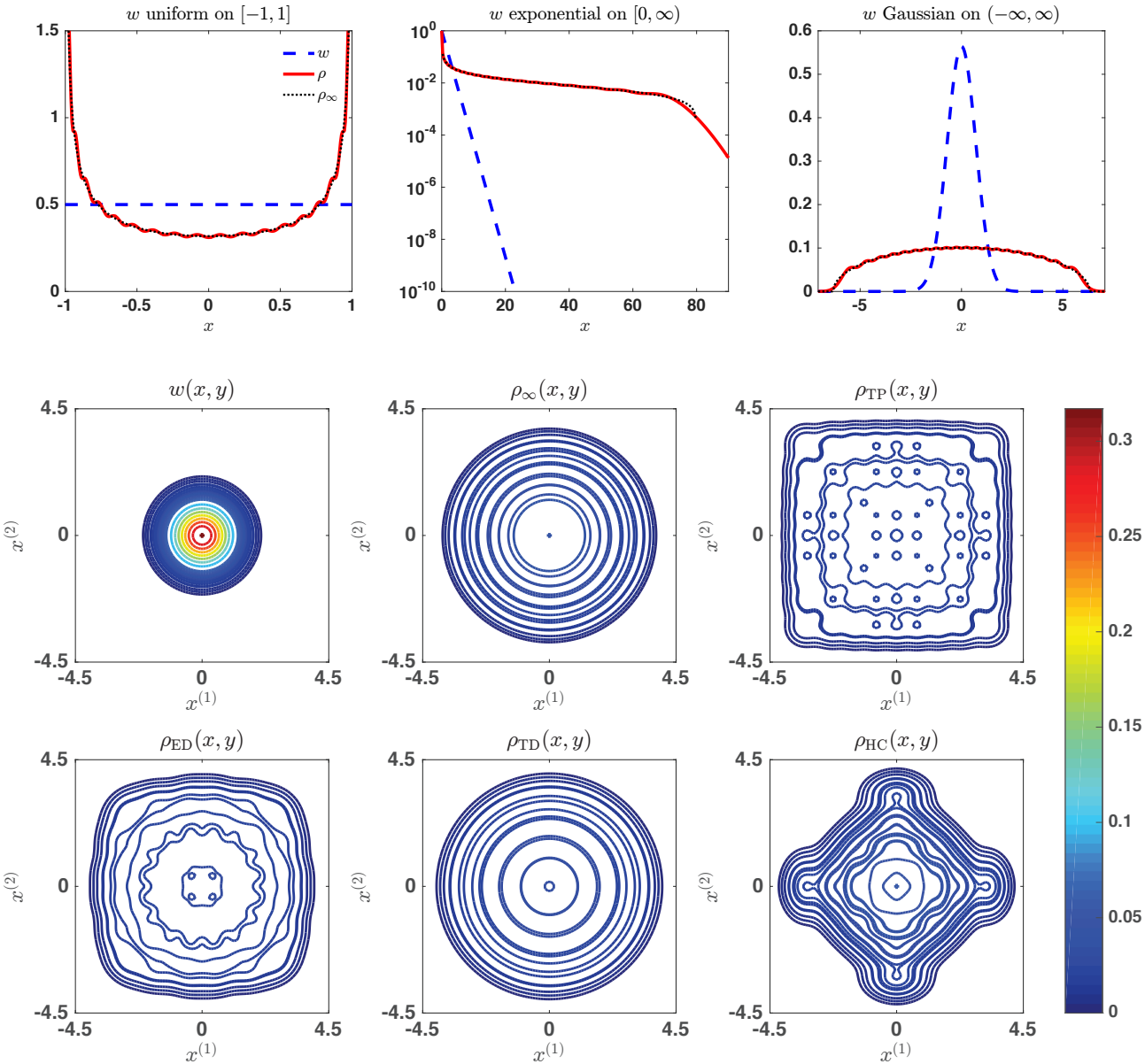
$$\mathbb{E}_{P,p_m}\left[Q_N - Q\right]^2 \lesssim \epsilon_\Lambda(Q) + M^{-K/2}, \qquad \epsilon_\Lambda(Q) := \inf_{R \in V(\Lambda)} \mathbb{E}_P\left[R(P) - Q(P)\right]^2.$$

This fixes the problem for essentially any (w, Λ):



Induced measure sampling

The induced measure ρ can be substantially different from w.



Optimizing sampling design

To enhance stability: optimize a least squares design using a type of D-optimal design.

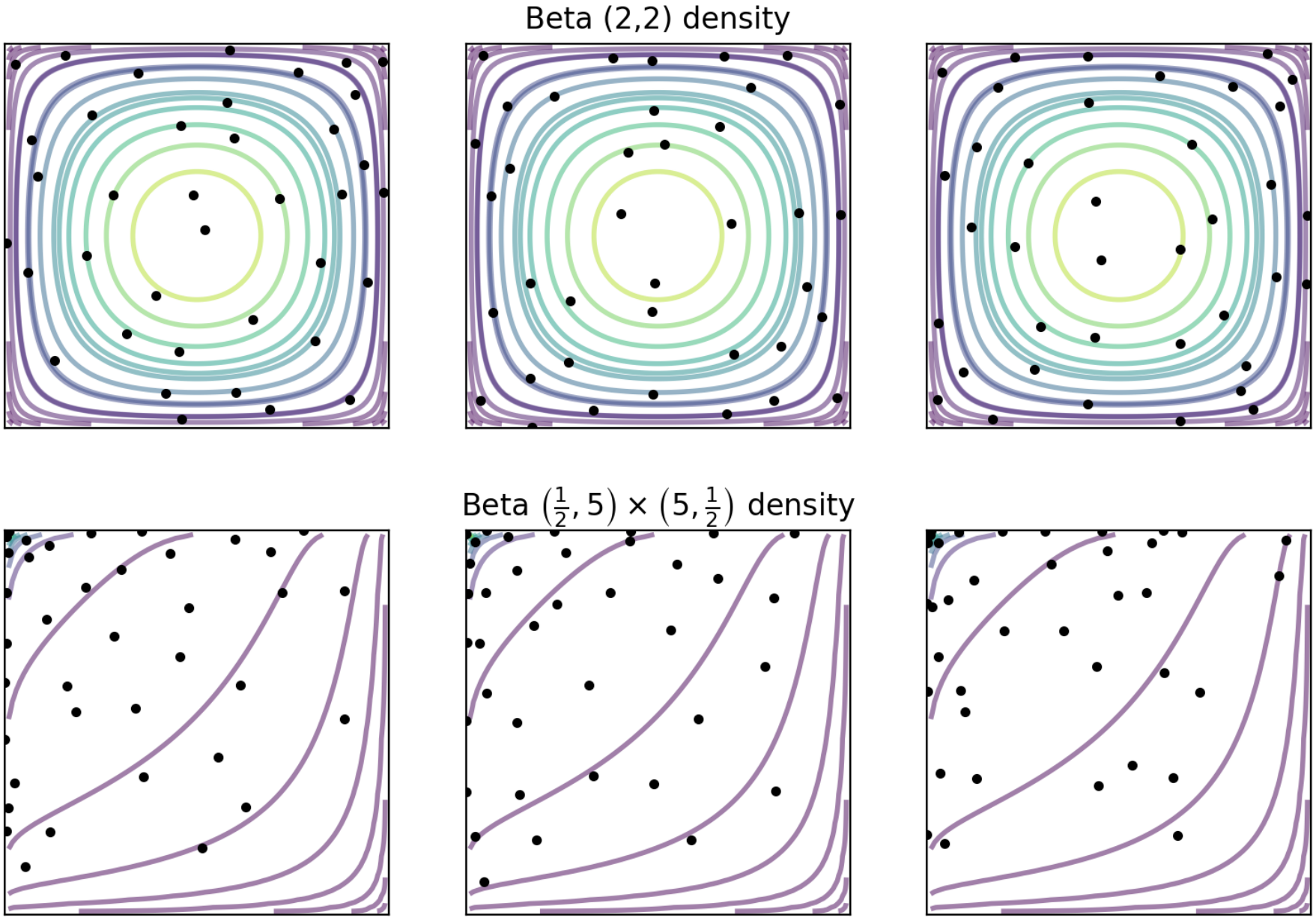
The samples we generate (approximately) solve the optimization problem,

$$\underset{p_1,\cdots,p_M}{\arg\max} \det\left(\widetilde{\boldsymbol{V}}^T\widetilde{\boldsymbol{V}}\right), \qquad (\widetilde{\boldsymbol{V}})_{m,j} = \frac{\phi_j(p_m)}{\sum_{\ell=1}^N \phi_\ell^2(p_m)}$$

We solve this problem using candidate points from induced measure (ρ) sampling.

$\Longrightarrow$ random ensemble $\{p_m\}_{m=1}^M$, but not iid samples.

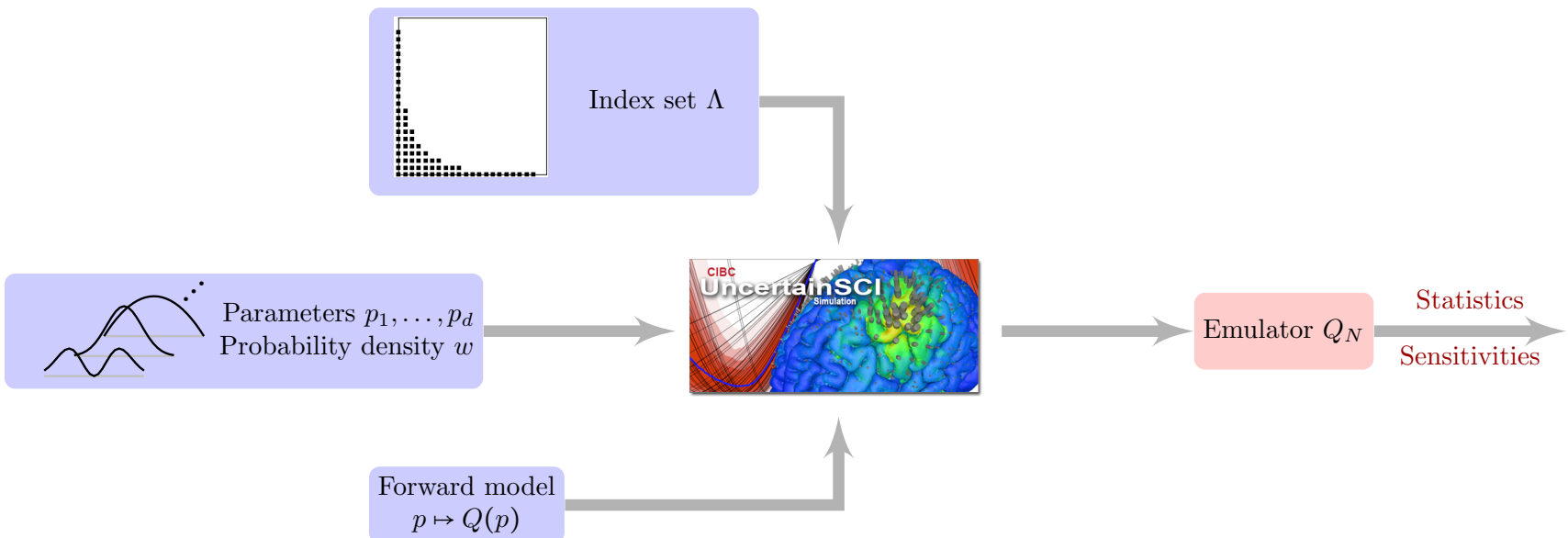
Sample designs



In summary

UncertainSCI performs forward UQ analysis,

- using PC emulators built by linear methods
- by non-intrusively sampling a provided forward model
- by sampling according to the induced distribution and a (weighted) D-optimal design
- through an emulator built by least squares

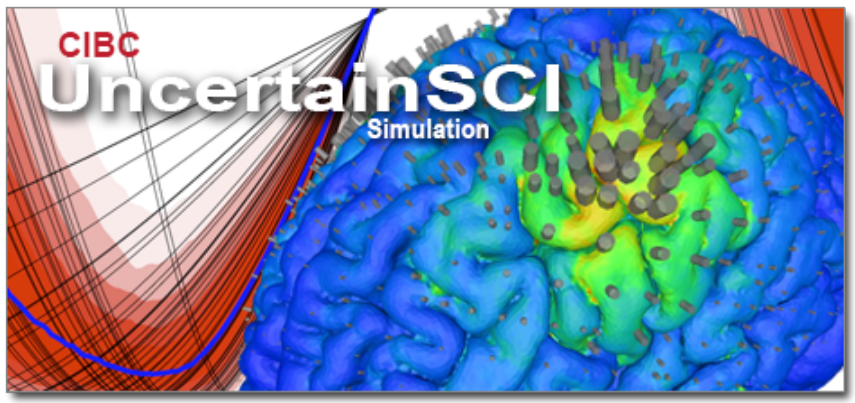


Moving forward

UncertainSCI: a Python framework for non-intrusive emulator-based forward UQ.

Upcoming features:

- non-tensorial densities w
- adaptive selection of index sets Λ
- positive multidimensional stochastic quadrature
- inverse problems, inference, design, and optimization



`https://www.sci.utah.edu/cibc-software/uncertainsci.html`