Simulating with uncertainty : the rough surface scattering problem

Uday Khankhoje

Assistant Professor, Electrical Engineering
Indian Institute of Technology Madras

A problem often encountered in numerical analysis

1. A simulation, $\sigma(\mathbf{x})$, $\mathbf{x} \in \mathbb{D}$ is computationally expensive
2. The input $\mathbf{x}$ or the domain $\mathbb{D}$ displays uncertainty,

   e.g. in simulating the modes of an optical fibre, the refractive index might not be exactly known, or the boundary may be rough

In this light, more useful than $\sigma(\mathbf{x})$ is the expectation $\langle \sigma(\mathbf{x}) \rangle$
Std. dev. in $\sigma(\mathbf{x})$ due to parameter uncertainty also interesting

A problem often encountered in numerical analysis

1. A simulation, $\sigma(\mathbf{x})$, $\mathbf{x} \in \mathbb{D}$ is computationally expensive
2. The input $\mathbf{x}$ or the domain $\mathbb{D}$ displays uncertainty,

   e.g. in simulating the modes of an optical fibre, the refractive index might not be exactly known, or the boundary may be rough

In this light, more useful than $\sigma(\mathbf{x})$ is the expectation $\langle\sigma(\mathbf{x})\rangle$
Std. dev. in $\sigma(\mathbf{x})$ due to parameter uncertainty also interesting

A problem often encountered in numerical analysis

1. A simulation, $\sigma(\mathbf{x})$, $\mathbf{x} \in \mathbb{D}$ is computationally expensive
2. The input $\mathbf{x}$ or the domain $\mathbb{D}$ displays uncertainty,

   e.g. in simulating the modes of an optical fibre, the refractive index

   might not be exactly known, or the boundary may be rough

In this light, more useful than $\sigma(\mathbf{x})$ is the expectation $\langle\sigma(\mathbf{x})\rangle$
Std. dev. in $\sigma(\mathbf{x})$ due to parameter uncertainty also interesting

Computing the expectation and standard deviation

Assumptions

Setup: d-dimensional $\mathbf{x} = [x_1, x_2, \ldots, x_d]$

Each x_i is mutually independent, distributed with known pdf ρ_i

Construct a multi-variate pdf, $\rho(\mathbf{x}) = \prod_{i=1}^{n} \rho_i(x_i)$

Quantities of interest

$$\text{Expectation} \quad \langle \sigma(\mathbf{x}) \rangle = \int_{\mathbb{D}} \sigma(\mathbf{x}) \rho(\mathbf{x}) d\mathbf{x}$$

$$\text{Std. dev.} \quad \Delta\sigma(\mathbf{x}) = \sqrt{\int_{\mathbb{D}} (\sigma(\mathbf{x}) - \langle \sigma(\mathbf{x}) \rangle)^2 \rho(\mathbf{x}) d\mathbf{x}}$$

Computing the expectation and standard deviation

Assumptions

Setup: d-dimensional $\mathbf{x} = [x_1, x_2, \ldots, x_d]$

Each x_i is mutually independent, distributed with known pdf ρ_i

Construct a multi-variate pdf, $\rho(\mathbf{x}) = \prod_{i=1}^{n} \rho_i(x_i)$

Quantities of interest

$$\text{Expectation} \quad \langle \sigma(\mathbf{x}) \rangle = \int_{\mathbb{D}} \sigma(\mathbf{x}) \rho(\mathbf{x}) d\mathbf{x}$$

$$\text{Std. dev.} \quad \Delta\sigma(\mathbf{x}) = \sqrt{\int_{\mathbb{D}} (\sigma(\mathbf{x}) - \langle \sigma(\mathbf{x}) \rangle)^2 \rho(\mathbf{x}) d\mathbf{x}}$$

Any problem with $\langle\sigma(\mathbf{x})\rangle = \int_{\mathbb{D}} \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x}$?

Yes!

$\sigma(\mathbf{x})$ is only known numerically.

Recall: $\sigma(\mathbf{x})$ is the output of an (possibly expensive) simulation

Monte Carlo method

The most commonly used method to estimate expectation.

Instantiate several $\mathbf{x}_i$'s and approximate $\langle\sigma(\mathbf{x})\rangle = \sum_{i=1}^{n} \frac{1}{n}\sigma(\mathbf{x}_i)$

Convergence rate: independent of d, but slow $O(\frac{1}{\sqrt{n}})$

Extra: Create a histogram from $\sigma(\mathbf{x}_i)$ values to estimate pdf of σ.

Objective of this talk: Discuss alternatives to Monte Carlo

Any problem with $\langle\sigma(\mathbf{x})\rangle = \int_{\mathbb{D}} \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x}$?

Yes!

$\sigma(\mathbf{x})$ is only known numerically.

Recall: $\sigma(\mathbf{x})$ is the output of an (possibly expensive) simulation

Monte Carlo method

The most commonly used method to estimate expectation.

Instantiate several $\mathbf{x}_i$'s and approximate $\langle\sigma(\mathbf{x})\rangle = \sum_{i=1}^{n} \frac{1}{n}\sigma(\mathbf{x}_i)$

Convergence rate: independent of d, but slow $O(\frac{1}{\sqrt{n}})$

Extra: Create a histogram from $\sigma(\mathbf{x}_i)$ values to estimate pdf of σ.

Objective of this talk: Discuss alternatives to Monte Carlo

Any problem with $\langle\sigma(\mathbf{x})\rangle = \int_{\mathbb{D}} \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x}$?

Yes!

$\sigma(\mathbf{x})$ is only known numerically.

Recall: $\sigma(\mathbf{x})$ is the output of an (possibly expensive) simulation

Monte Carlo method

The most commonly used method to estimate expectation.

Instantiate several $\mathbf{x}_i$'s and approximate $\langle\sigma(\mathbf{x})\rangle = \sum_{i=1}^{n} \frac{1}{n}\sigma(\mathbf{x}_i)$

Convergence rate: independent of d, but slow $O(\frac{1}{\sqrt{n}})$

Extra: Create a histogram from $\sigma(\mathbf{x}_i)$ values to estimate pdf of σ.

Objective of this talk: Discuss alternatives to Monte Carlo

The alternatives to Monte Carlo

Broadly, two families of methods will be discussed:

1. **Galerkin Polynomial Chaos** (gPC)

   Need to code a new solver, works for small range of d values

2. **Stochastic Collocation** (SC)

   Uses an existing solver, works for small–medium range of d values

   Attractive, because compatible with commercial software

What about Monte Carlo?

Use for benchmarking and for large range of d values

The alternatives to Monte Carlo

Broadly, two families of methods will be discussed:

1. **Galerkin Polynomial Chaos** (gPC)

   Need to code a <u>new</u> solver, works for small range of d values

2. **Stochastic Collocation** (SC)

   Uses an <u>existing</u> solver, works for small–medium range of d values

   Attractive, because compatible with commercial software

What about Monte Carlo?

Use for benchmarking and for large range of d values

Start with Stochastic Collocation: Polynomial interpolation

For simplicity, consider a 1-D case of the simulation, $\sigma(x)$

Polynomial interpolation and integration

- Choose n-points to evaluate $\sigma(x)$ and construct a $n-1$ degree polynomial (Lagrange interpolation): $\sigma_{n-1}(x) = \sum_{i=1}^{n} \sigma(x_i) L_i(x)$
  Recall, $L_i(x) = \prod_{j=1, j\neq i}^{n} (x - x_j)/(x_i - x_j)$
- Compute expectation as
  $\langle \sigma(x) \rangle \approx \sum_{i=1}^{n} \sigma(x_i) \int L_i(x)\rho(x)dx = \sum_{i=1}^{n} \sigma(x_i)\, \alpha_i$
- α_i can be pre-computed, and $\langle \sigma(x) \rangle$ is accurate to order $n-1$.

Can we do better?

Start with Stochastic Collocation: Polynomial interpolation

For simplicity, consider a 1-D case of the simulation, $\sigma(x)$

Polynomial interpolation and integration

- Choose n-points to evaluate $\sigma(x)$ and construct a $n-1$ degree polynomial (Lagrange interpolation): $\sigma_{n-1}(x) = \sum_{i=1}^{n} \sigma(x_i) L_i(x)$
  Recall, $L_i(x) = \prod_{j=1, j\neq i}^{n} (x - x_j)/(x_i - x_j)$
- Compute expectation as
  $\langle \sigma(x) \rangle \approx \sum_{i=1}^{n} \sigma(x_i) \int L_i(x)\rho(x)dx = \sum_{i=1}^{n} \sigma(x_i)\, \alpha_i$
- α_i can be pre-computed, and $\langle \sigma(x) \rangle$ is accurate to order $n-1$.

Can we do better?

Start with Stochastic Collocation: Polynomial interpolation

For simplicity, consider a 1-D case of the simulation, $\sigma(x)$

Polynomial interpolation and integration

- Choose n-points to evaluate $\sigma(x)$ and construct a $n-1$ degree polynomial (Lagrange interpolation): $\sigma_{n-1}(x) = \sum_{i=1}^{n} \sigma(x_i) L_i(x)$
  Recall, $L_i(x) = \prod_{j=1, j\neq i}^{n} (x - x_j)/(x_i - x_j)$
- Compute expectation as
  $\langle \sigma(x) \rangle \approx \sum_{i=1}^{n} \sigma(x_i) \int L_i(x)\rho(x)dx = \sum_{i=1}^{n} \sigma(x_i)\, \alpha_i$
- α_i can be pre-computed, and $\langle \sigma(x) \rangle$ is accurate to order $n-1$.

Can we do better?

Stochastic Collocation: Gaussian quadrature

Yes! Gaussian quadrature

- Use the theory of orthogonal polynomials, i.e. find polynomials s.t. $\int p_m(x)p_n(x)\rho(x)dx = \delta_{m,n}$
- Pick the n points, $\{x_i\}$ to be roots of n^{th} degree polynomial, $p_n(x)$
- This also gives $\langle\sigma(x)\rangle \approx \sum_{i=1}^{n} \sigma(x_i)\, \alpha_i$, but now integral is accurate to order $2n-1$

Common orthogonal polynomials

- *Legendre*, $x \in [-1,1]$, $\rho(x) = 1$.
- *Jacobi*, $x \in (-1,1)$, $\rho(x) = (1-x)^{\alpha}(1+x)^{\beta}$, $\alpha, \beta > -1$.
- *Hermite*, $x \in (-\infty,\infty)$, $\rho(x) = e^{-x^2}$.

Stochastic Collocation: Gaussian quadrature

Yes! Gaussian quadrature

- Use the theory of orthogonal polynomials, i.e. find polynomials s.t. $\int p_m(x)p_n(x)\rho(x)dx = \delta_{m,n}$
- Pick the n points, $\{x_i\}$ to be roots of n^{th} degree polynomial, $p_n(x)$
- This also gives $\langle\sigma(x)\rangle \approx \sum_{i=1}^{n} \sigma(x_i)\,\alpha_i$, but now integral is accurate to order $2n-1$

Common orthogonal polynomials

- *Legendre*, $x \in [-1,1]$, $\rho(x) = 1$.
- *Jacobi*, $x \in (-1,1)$, $\rho(x) = (1-x)^{\alpha}(1+x)^{\beta}$, $\alpha, \beta > -1$.
- *Hermite*, $x \in (-\infty,\infty)$, $\rho(x) = e^{-x^2}$.

Stochastic Collocation: Gaussian quadrature

Yes! Gaussian quadrature

- Use the theory of orthogonal polynomials, i.e. find polynomials s.t. $\int p_m(x)p_n(x)\rho(x)dx = \delta_{m,n}$
- Pick the n points, $\{x_i\}$ to be roots of n^{th} degree polynomial, $p_n(x)$
- This also gives $\langle\sigma(x)\rangle \approx \sum_{i=1}^{n} \sigma(x_i)\,\alpha_i$, but now integral is accurate to order $2n-1$

Common orthogonal polynomials

- *Legendre*, $x \in [-1,1]$, $\rho(x) = 1$.
- *Jacobi*, $x \in (-1,1)$, $\rho(x) = (1-x)^{\alpha}(1+x)^{\beta}$, $\alpha, \beta > -1$.
- *Hermite*, $x \in (-\infty,\infty)$, $\rho(x) = e^{-x^2}$.

But our problem is d-dimensional!

Solution: Extend one-dimensional Gaussian quadrature to d dimensions

- With $\mathbf{x} = [x_i, x_2, \ldots, x_d]$, express $\sigma(\mathbf{x}) = \prod_{i=1}^{d} \sigma_i(x_i)$ (implicitly)
- The integral splits up: $\int \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x} = \prod_i \int \sigma_i(x_i)\rho_i(x_i)dx_i$
- Apply n-point Gaussian quadrature (GQ) in each dimension: $\langle\sigma(\mathbf{x})\rangle = \prod_{i=1}^{d} \sum_{j=1}^{n} \sigma_i(x_{i,j})\alpha_j$. Combine the σ_i's to get $\sigma(\mathbf{x}_k)$.
- An example in 2D (x, y) and a 2-point GQ in each dimension: $\langle\sigma(x,y)\rangle = [\sigma_x(x_1)\alpha_1 + \sigma_x(x_2)\alpha_2][\sigma_y(y_1)\alpha_1 + \sigma_y(y_2)\alpha_2] =$ $\sigma(x_1,y_1)\alpha_{1,1} + \sigma(x_1,y_2)\alpha_{1,2} + \sigma(x_2,y_1)\alpha_{2,1} + \sigma(x_2,y_2)\alpha_{2,2}$

All dimensions are multiplied: called the tensor product rule

But our problem is d-dimensional!

Solution: Extend one-dimensional Gaussian quadrature to d dimensions

- With $\mathbf{x} = [x_i, x_2, \ldots, x_d]$, express $\sigma(\mathbf{x}) = \prod_{i=1}^{d} \sigma_i(x_i)$ (implicitly)
- The integral splits up: $\int \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x} = \prod_i \int \sigma_i(x_i)\rho_i(x_i)dx_i$
- Apply n-point Gaussian quadrature (GQ) in each dimension: $\langle\sigma(\mathbf{x})\rangle = \prod_{i=1}^{d} \sum_{j=1}^{n} \sigma_i(x_{i,j})\alpha_j$. Combine the σ_i's to get $\sigma(\mathbf{x}_k)$.
- An example in 2D (x, y) and a 2-point GQ in each dimension: $\langle\sigma(x,y)\rangle = [\sigma_x(x_1)\alpha_1 + \sigma_x(x_2)\alpha_2][\sigma_y(y_1)\alpha_1 + \sigma_y(y_2)\alpha_2] =$ $\sigma(x_1,y_1)\alpha_{1,1} + \sigma(x_1,y_2)\alpha_{1,2} + \sigma(x_2,y_1)\alpha_{2,1} + \sigma(x_2,y_2)\alpha_{2,2}$

All dimensions are multiplied: called the tensor product rule

But our problem is d-dimensional!

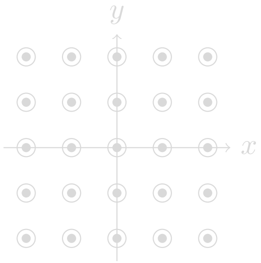
Solution: Extend one-dimensional Gaussian quadrature to d dimensions

- With $\mathbf{x} = [x_i, x_2, \ldots, x_d]$, express $\sigma(\mathbf{x}) = \prod_{i=1}^{d} \sigma_i(x_i)$ (implicitly)
- The integral splits up: $\int \sigma(\mathbf{x})\rho(\mathbf{x})d\mathbf{x} = \prod_i \int \sigma_i(x_i)\rho_i(x_i)dx_i$
- Apply n-point Gaussian quadrature (GQ) in each dimension: $\langle\sigma(\mathbf{x})\rangle = \prod_{i=1}^{d} \sum_{j=1}^{n} \sigma_i(x_{i,j})\alpha_j$. Combine the σ_i's to get $\sigma(\mathbf{x}_k)$.
- An example in 2D (x, y) and a 2-point GQ in each dimension: $\langle\sigma(x,y)\rangle = [\sigma_x(x_1)\alpha_1 + \sigma_x(x_2)\alpha_2][\sigma_y(y_1)\alpha_1 + \sigma_y(y_2)\alpha_2] =$ $\sigma(x_1,y_1)\alpha_{1,1} + \sigma(x_1,y_2)\alpha_{1,2} + \sigma(x_2,y_1)\alpha_{2,1} + \sigma(x_2,y_2)\alpha_{2,2}$

All dimensions are multiplied: called the tensor product rule

Visualizing the tensor product rule

- E.g. function evaluation points in a 2D 5-point GQ (25 evals) : Denote as $\langle\sigma\rangle_{5,5}$
- Curse of dimensionality is clear: number of function evaluations $= n^d$
- Can we do better?



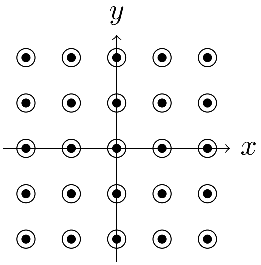
Theorem: (Mysovskikh 1968, Möller 1976)

To attain a polynomial exactness equal to m, the (optimal) required number of grid-points has lower and upper bounds given by

$$N_{\text{min}} = \binom{d + \lfloor m \rfloor/2}{\lfloor m \rfloor/2} \leq N_{\text{opt}} \leq \binom{d+m}{m} = N_{\text{max}}$$

Visualizing the tensor product rule

- E.g. function evaluation points in a 2D 5-point GQ (25 evals) : Denote as $\langle\sigma\rangle_{5,5}$
- Curse of dimensionality is clear: number of function evaluations $= n^d$
- Can we do better?



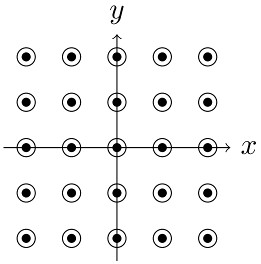
Theorem: (Mysovskikh 1968, Möller 1976)

To attain a polynomial exactness equal to m, the (optimal) required number of grid-points has lower and upper bounds given by

$$N_{\text{min}} = \binom{d + \lfloor m \rfloor/2}{\lfloor m \rfloor/2} \leq N_{\text{opt}} \leq \binom{d+m}{m} = N_{\text{max}}$$

Visualizing the tensor product rule

- E.g. function evaluation points in a 2D 5-point GQ (25 evals) : Denote as $\langle\sigma\rangle_{5,5}$
- Curse of dimensionality is clear: number of function evaluations $= n^d$
- Can we do better?



Theorem: (Mysovskikh 1968, Möller 1976)

To attain a polynomial exactness equal to m, the (optimal) required number of grid-points has lower and upper bounds given by

$$N_{\text{min}} = \binom{d + \lfloor m \rfloor/2}{\lfloor m \rfloor/2} \le N_{\text{opt}} \le \binom{d+m}{m} = N_{\text{max}}$$

Sparse Grids: Smolyak (1963)

1. Every $\langle \sigma \rangle_{i,j}$ is an approximation to the actual integral, $\langle \sigma \rangle$
2. Introduce a "level" parameter $k = (i + j)$ and let the max level be denoted by $l = \max\{k\}$

Telescope a series of *different levels* to approximate $\langle \sigma \rangle$

- e.g. (Max) level $l = 4$:
  $\langle \sigma \rangle \approx [\langle \sigma \rangle_{3,1} + \langle \sigma \rangle_{1,3} + \langle \sigma \rangle_{2,2}]_{\text{k=4}} - [\langle \sigma \rangle_{2,1} + \langle \sigma \rangle_{1,2}]_{\text{k=3}}$
- e.g. (Max) level $l = 5$:
  $\langle \sigma \rangle \approx [\langle \sigma \rangle_{3,2} + \langle \sigma \rangle_{2,3} + \langle \sigma \rangle_{1,4} + \langle \sigma \rangle_{4,1}] - [\langle \sigma \rangle_{3,1} + \langle \sigma \rangle_{1,3} + \langle \sigma \rangle_{2,2}]$

As the max level increases, approximation becomes better

Sparse Grids: Smolyak (1963)

1. Every $\langle\sigma\rangle_{i,j}$ is an approximation to the actual integral, $\langle\sigma\rangle$
2. Introduce a "level" parameter $k = (i + j)$ and let the max level be denoted by $l = \max\{k\}$

Telescope a series of *different levels* to approximate $\langle\sigma\rangle$

- e.g. (Max) level $l = 4$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{\text{k=4}} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{\text{k=3}}$
- e.g. (Max) level $l = 5$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,2} + \langle\sigma\rangle_{2,3} + \langle\sigma\rangle_{1,4} + \langle\sigma\rangle_{4,1}] - [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]$

As the max level increases, approximation becomes better

Sparse Grids: Smolyak (1963)

1. Every $\langle\sigma\rangle_{i,j}$ is an approximation to the actual integral, $\langle\sigma\rangle$
2. Introduce a “level” parameter $k = (i + j)$ and let the max level be denoted by $l = \max\{k\}$

Telescope a series of *different levels* to approximate $\langle\sigma\rangle$

- e.g. (Max) level $l = 4$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{k=4} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{k=3}$
- e.g. (Max) level $l = 5$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,2} + \langle\sigma\rangle_{2,3} + \langle\sigma\rangle_{1,4} + \langle\sigma\rangle_{4,1}] - [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]$

As the max level increases, approximation becomes better

Sparse Grids: Smolyak (1963)

1. Every $\langle\sigma\rangle_{i,j}$ is an approximation to the actual integral, $\langle\sigma\rangle$
2. Introduce a “level” parameter $k = (i + j)$ and let the max level be denoted by $l = \max\{k\}$

Telescope a series of *different levels* to approximate $\langle\sigma\rangle$

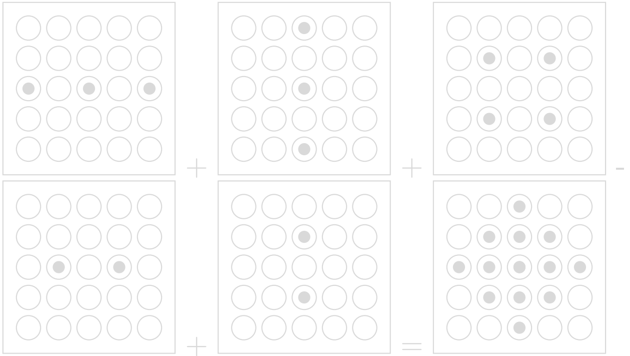
- e.g. (Max) level $l = 4$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{\text{k=4}} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{\text{k=3}}$
- e.g. (Max) level $l = 5$:
  $\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,2} + \langle\sigma\rangle_{2,3} + \langle\sigma\rangle_{1,4} + \langle\sigma\rangle_{4,1}] - [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]$

As the max level increases, approximation becomes better

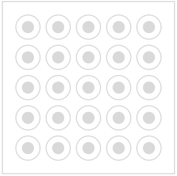
Visualizing the sparse grid rule

$$\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{\mathsf{k=4}} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{\mathsf{k=3}}$$

Sparse Grid Points



Tensor Grid Points



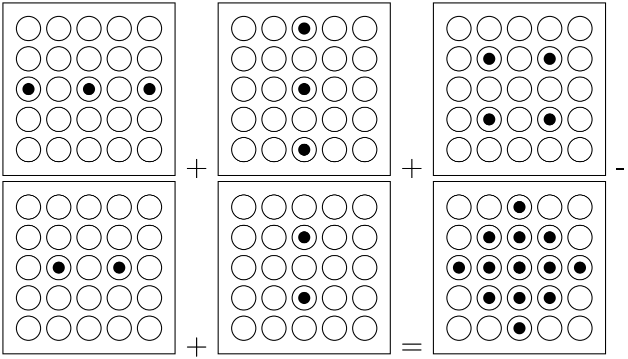
Substantial savings in higher dims

13 Points (SG) v/s **25** points (TP)!

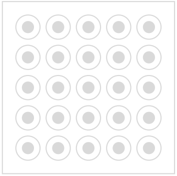
Visualizing the sparse grid rule

$$\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{k=4} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{k=3}$$

Sparse Grid Points



Tensor Grid Points



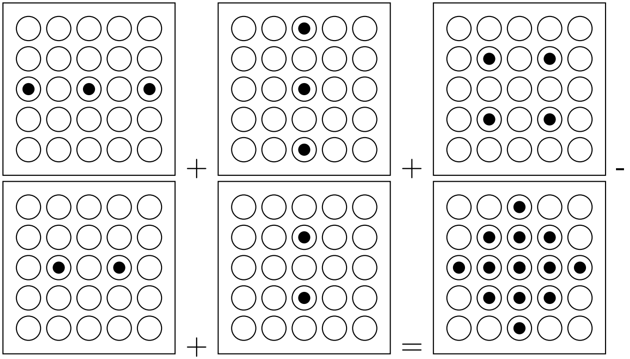
Substantial savings in higher dims

13 Points (SG) v/s **25** points (TP)!

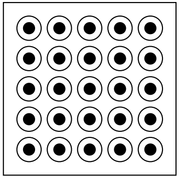
Visualizing the sparse grid rule

$$\langle\sigma\rangle \approx [\langle\sigma\rangle_{3,1} + \langle\sigma\rangle_{1,3} + \langle\sigma\rangle_{2,2}]_{\text{k=4}} - [\langle\sigma\rangle_{2,1} + \langle\sigma\rangle_{1,2}]_{\text{k=3}}$$

Sparse Grid Points



Tensor Grid Points



Substantial savings in higher dims

13 Points (SG) v/s **25** points (TP)!

Sparse Grid rule: finer points

TP rule: $\langle\sigma\rangle = \sum_{i_1=1}^{n_1} \dots \sum_{i_d=1}^{n_d} \sigma(x_{i_1}, \dots, x_{i_d})\alpha_{i_1} \dots \alpha_{i_d}$
$= (Q^{n_1} \times Q^{n_2} \times \dots \times Q^{n_d})[\sigma]$
n_i point quadrature in the i^{th} dim; $\prod n_i \approx n^d$ points

SG rule: With max level l, and $k = k_1 + k_2 + \dots + k_d$:
$$\langle\sigma\rangle = \sum_{l-d+1 \le k \le l} (-1)^{l-k} \binom{d-1}{l-k} (Q^{k_1} \times Q^{k_2} \times \dots \times Q^{k_d})[\sigma]$$

- k_i point quadrature in the i^{th} dim; $\approx 2^l d^l / l!$ points
- Nested quadrature rules (e.g. Clenshaw-Curtis, Gauss-Konrad) re-use fn evaluation points between levels.

Sparse Grid rule: finer points

TP rule: $\langle\sigma\rangle = \sum_{i_1=1}^{n_1} \ldots \sum_{i_d=1}^{n_d} \sigma(x_{i_1}, \ldots, x_{i_d})\alpha_{i_1} \ldots \alpha_{i_d}$
$= (Q^{n_1} \times Q^{n_2} \times \ldots \times Q^{n_d})[\sigma]$
n_i point quadrature in the i^{th} dim; $\prod n_i \approx n^d$ points

SG rule: With max level l, and $k = k_1 + k_2 + \ldots + k_d$:

$$\langle\sigma\rangle = \sum_{l-d+1 \le k \le l} (-1)^{l-k} \binom{d-1}{l-k} (Q^{k_1} \times Q^{k_2} \times \ldots \times Q^{k_d})[\sigma]$$

- k_i point quadrature in the i^{th} dim; $\approx 2^l d^l / l!$ points
- Nested quadrature rules (e.g. Clenshaw-Curtis, Gauss-Konrad) re-use fn evaluation points between levels.

Sparse Grid rule: finer points

TP rule: $\langle\sigma\rangle = \sum_{i_1=1}^{n_1} \ldots \sum_{i_d=1}^{n_d} \sigma(x_{i_1}, \ldots, x_{i_d})\alpha_{i_1} \ldots \alpha_{i_d}$
$= (Q^{n_1} \times Q^{n_2} \times \ldots \times Q^{n_d})[\sigma]$
n_i point quadrature in the i^{th} dim; $\prod n_i \approx n^d$ points

SG rule: With max level l, and $k = k_1 + k_2 + \ldots + k_d$:

$$\langle\sigma\rangle = \sum_{l-d+1 \le k \le l} (-1)^{l-k} \binom{d-1}{l-k} (Q^{k_1} \times Q^{k_2} \times \ldots \times Q^{k_d})[\sigma]$$

- k_i point quadrature in the i^{th} dim; $\approx 2^l d^l / l!$ points
- Nested quadrature rules (e.g. Clenshaw-Curtis, Gauss-Konrad) re-use fn evaluation points between levels.

Switching gears: from sampling to projection

The two methods (Monte Carlo, Stochastic Collocation) considered so far were of a "sampling" kind: $\langle\sigma\rangle$ estimated using samples of $\sigma(\mathbf{x})$.

Projection based approach: Overview of Galerkin method

- Governing equation: $\Theta f(\mathbf{x}) = g(\mathbf{x})$, where Θ is an operator, g is a known function, and f is to be determined.
- Project f in a known basis, $\{\phi_j\}$: $f(\mathbf{x}) = \sum_j u_j \phi_j(\mathbf{x})$
- Take an inner product with the same basis functions on both sides to get: $\sum_j \langle \phi_i(\mathbf{x}), \Theta\phi_j(\mathbf{x})\rangle u_j = \langle \phi_i(\mathbf{x}), g(\mathbf{x})\rangle$. *BC* used to simplify.
- This is a system of equations; solve for u and get f.

Straightforward when $\mathbf{x}$ is spatio-temporal. But when stochastic?

Switching gears: from sampling to projection

The two methods (Monte Carlo, Stochastic Collocation) considered so far were of a "sampling" kind: $\langle \sigma \rangle$ estimated using samples of $\sigma(\mathbf{x})$.

Projection based approach: Overview of Galerkin method

- Governing equation: $\Theta f(\mathbf{x}) = g(\mathbf{x})$, where Θ is an operator, g is a known function, and f is to be determined.
- Project f in a known basis, $\{\phi_j\}$: $f(\mathbf{x}) = \sum_j u_j \phi_j(\mathbf{x})$
- Take an inner product with the same basis functions on both sides to get: $\sum_j \langle \phi_i(\mathbf{x}), \Theta \phi_j(\mathbf{x}) \rangle u_j = \langle \phi_i(\mathbf{x}), g(\mathbf{x}) \rangle$. *BC* used to simplify.
- This is a system of equations; solve for u and get f.

Straightforward when $\mathbf{x}$ is spatio-temporal. But when stochastic?

Switching gears: from sampling to projection

The two methods (Monte Carlo, Stochastic Collocation) considered so far were of a "sampling" kind: $\langle\sigma\rangle$ estimated using samples of $\sigma(\mathbf{x})$.

Projection based approach: Overview of Galerkin method

- Governing equation: $\Theta f(\mathbf{x}) = g(\mathbf{x})$, where Θ is an operator, g is a known function, and f is to be determined.
- Project f in a known basis, $\{\phi_j\}$: $f(\mathbf{x}) = \sum_j u_j\phi_j(\mathbf{x})$
- Take an inner product with the same basis functions on both sides to get: $\sum_j \langle\phi_i(\mathbf{x}), \Theta\phi_j(\mathbf{x})\rangle u_j = \langle\phi_i(\mathbf{x}), g(\mathbf{x})\rangle$. *BC* used to simplify.
- This is a system of equations; solve for u and get f.

Straightforward when $\mathbf{x}$ is spatio-temporal. But when stochastic?

Switching gears: from sampling to projection

The two methods (Monte Carlo, Stochastic Collocation) considered so far were of a “sampling” kind: $\langle \sigma \rangle$ estimated using samples of $\sigma(\mathbf{x})$.

Projection based approach: Overview of Galerkin method

- Governing equation: $\Theta f(\mathbf{x}) = g(\mathbf{x})$, where Θ is an operator, g is a known function, and f is to be determined.
- Project f in a known basis, $\{\phi_j\}$: $f(\mathbf{x}) = \sum_j u_j \phi_j(\mathbf{x})$
- Take an inner product with the same basis functions on both sides to get: $\sum_j \langle \phi_i(\mathbf{x}), \Theta \phi_j(\mathbf{x}) \rangle u_j = \langle \phi_i(\mathbf{x}), g(\mathbf{x}) \rangle$. *BC* used to simplify.
- This is a system of equations; solve for u and get f.

Straightforward when $\mathbf{x}$ is spatio-temporal. But when stochastic?

generalized Polynomial Chaos (gPC): a very brief history

- Polynomial Chaos (PC): coined by Norbert Wiener studying Gaussian stochastic processes (1938). Used Hermite polynomials as basis.
- Ghanem (1998) used theory of Wiener-Hermite PC to represent random processes in an orthogonal basis of Hermite polynomials.
- Xiu, Karniadakis (2002) generalize to non-Gaussian using other orthogonal polynomials, wavelets, etc: generalized PC (gPC)

Finally, solve the gPC system of equations using galerkin projection: stochastic galerkin (SG) method.

generalized Polynomial Chaos (gPC): a very brief history

- Polynomial Chaos (PC): coined by Norbert Wiener studying Gaussian stochastic processes (1938). Used Hermite polynomials as basis.
- Ghanem (1998) used theory of Wiener-Hermite PC to represent random processes in an orthogonal basis of Hermite polynomials.
- Xiu, Karniadakis (2002) generalize to non-Gaussian using other orthogonal polynomials, wavelets, etc: generalized PC (gPC)

Finally, solve the gPC system of equations using galerkin projection: stochastic galerkin (SG) method.

generalized Polynomial Chaos (gPC): a very brief history

- Polynomial Chaos (PC): coined by Norbert Wiener studying Gaussian stochastic processes (1938). Used Hermite polynomials as basis.
- Ghanem (1998) used theory of Wiener-Hermite PC to represent random processes in an orthogonal basis of Hermite polynomials.
- Xiu, Karniadakis (2002) generalize to non-Gaussian using other orthogonal polynomials, wavelets, etc: generalized PC (gPC)

Finally, solve the gPC system of equations using galerkin projection: stochastic galerkin (SG) method.

The basis functions in gPC

- Start with a RV, call it z as before. Let it have a distribution function, $F_z(\theta) = P(z \leq \theta)$ and a pdf $\rho(\theta)$ s.t. $dF_z(\theta) = \rho(\theta)d\theta$
- The generalized Polynomial Chaos basis functions are orthogonal basis functions, $\psi_i(z)$, satisfying :
  $\langle \psi_i(z)\psi_j(z)\rangle = \int \psi_i(\theta)\psi_j(\theta)\rho(\theta)d\theta = \gamma_i \delta_{ij}$
- Construct linear space of polynomials of degree at most n: $\mathbb{P}_n(z)$
- Various kinds depending on $\rho(\theta)$
  - *Legendre*, $\theta \in [-1, 1]$, $\rho(\theta) = 1/2$.
  - *Jacobi*, $\theta \in (-1, 1)$, $\rho(\theta) = (1-\theta)^\alpha(1+\theta)^\beta$, $\alpha, \beta > -1$.
  - *Hermite*, $\theta \in (-\infty, \infty)$, $\rho(\theta) = e^{-\theta^2}$.

The basis functions in gPC

- Start with a RV, call it z as before. Let it have a distribution function, $F_z(\theta) = P(z \leq \theta)$ and a pdf $\rho(\theta)$ s.t. $dF_z(\theta) = \rho(\theta) d\theta$
- The generalized Polynomial Chaos basis functions are orthogonal basis functions, $\psi_i(z)$, satisfying :
  $\langle \psi_i(z) \psi_j(z) \rangle = \int \psi_i(\theta) \psi_j(\theta) \rho(\theta) d\theta = \gamma_i \delta_{ij}$
- Construct linear space of polynomials of degree at most n: $\mathbb{P}_n(z)$
- Various kinds depending on $\rho(\theta)$
  - *Legendre*, $\theta \in [-1, 1]$, $\rho(\theta) = 1/2$.
  - *Jacobi*, $\theta \in (-1, 1)$, $\rho(\theta) = (1 - \theta)^\alpha (1 + \theta)^\beta$, $\alpha, \beta > -1$.
  - *Hermite*, $\theta \in (-\infty, \infty)$, $\rho(\theta) = e^{-\theta^2}$.

The basis functions in gPC

- Start with a RV, call it z as before. Let it have a distribution function, $F_z(\theta) = P(z \leq \theta)$ and a pdf $\rho(\theta)$ s.t. $dF_z(\theta) = \rho(\theta)d\theta$
- The generalized Polynomial Chaos basis functions are orthogonal basis functions, $\psi_i(z)$, satisfying :
  $\langle \psi_i(z)\psi_j(z)\rangle = \int \psi_i(\theta)\psi_j(\theta)\rho(\theta)d\theta = \gamma_i\delta_{ij}$
- Construct linear space of polynomials of degree at most n: $\mathbb{P}_n(z)$
- Various kinds depending on $\rho(\theta)$
  - *Legendre*, $\theta \in [-1, 1]$, $\rho(\theta) = 1/2$.
  - *Jacobi*, $\theta \in (-1, 1)$, $\rho(\theta) = (1-\theta)^\alpha(1+\theta)^\beta$, $\alpha, \beta > -1$.
  - *Hermite*, $\theta \in (-\infty, \infty)$, $\rho(\theta) = e^{-\theta^2}$.

A scalar example of gPC-SG

- Consider: a simple equation in one unknown, u: $a\,u = b$
  Now: Let the system parameters a, b have some uncertainty,
  e.g. $a(\theta) = a_0 + \alpha\theta$, where θ is a uniform RV in $[-0.5, 0.5]$.
  In this case what is $\langle u \rangle$ or $\langle u^2 \rangle$?
- Expand: solution in the ψ basis – $u(\theta) = \sum_{j=1}^{n} u_j \psi_j(\theta)$
- Do: Galerkin testing with the same basis functions:
  Get a system of equations, where we solve for u: $Au = b$, where
  $A_{ij} = \langle \psi_i(\theta) a_0 \psi_j(\theta) + \psi_i(\theta)\alpha\theta\psi_j(\theta) \rangle = a_0\gamma_i\delta_{ij} + \alpha\langle \psi_i(\theta)\theta\psi_j(\theta) \rangle$
  and $b_j = \langle \psi_i(\theta) b(\theta) \rangle$

So: $\langle u \rangle = \int u(\theta)\rho(\theta)d\theta = \sum_i u_i \langle \psi_i(\theta) \rangle$ and $\langle u^2 \rangle = \sum_i u_i^2 \gamma_i$.
$\Longrightarrow$ std. dev. in u can be computed : $\sqrt{\langle u^2 \rangle - \langle u \rangle^2}$
$\Longrightarrow$ uncertainty **quantified**!

A scalar example of gPC-SG

- Consider: a simple equation in one unknown, u: $a\,u = b$
  Now: Let the system parameters a, b have some uncertainty,
  e.g. $a(\theta) = a_0 + \alpha\theta$, where θ is a uniform RV in $[-0.5, 0.5]$.
  In this case what is $\langle u \rangle$ or $\langle u^2 \rangle$?
- Expand: solution in the ψ basis – $u(\theta) = \sum_{j=1}^{n} u_j \psi_j(\theta)$
- Do: Galerkin testing with the same basis functions:
  Get a system of equations, where we solve for u: $Au = b$, where
  $A_{ij} = \langle \psi_i(\theta) a_0 \psi_j(\theta) + \psi_i(\theta)\alpha\theta\psi_j(\theta) \rangle = a_0 \gamma_i \delta_{ij} + \alpha \langle \psi_i(\theta)\theta\psi_j(\theta) \rangle$
  and $b_j = \langle \psi_i(\theta) b(\theta) \rangle$

So: $\langle u \rangle = \int u(\theta)\rho(\theta)d\theta = \sum_i u_i \langle \psi_i(\theta) \rangle$ and $\langle u^2 \rangle = \sum_i u_i^2 \gamma_i$.
$\implies$ std. dev. in u can be computed : $\sqrt{\langle u^2 \rangle - \langle u \rangle^2}$
$\implies$ uncertainty **quantified**!

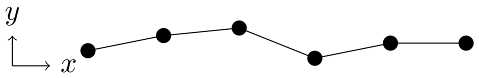
A scalar example of gPC-SG

- Consider: a simple equation in one unknown, u: $a\,u = b$
  Now: Let the system parameters a, b have some uncertainty,
  e.g. $a(\theta) = a_0 + \alpha\theta$, where θ is a uniform RV in $[-0.5, 0.5]$.
  In this case what is $\langle u\rangle$ or $\langle u^2\rangle$?
- Expand: solution in the ψ basis – $u(\theta) = \sum_{j=1}^{n} u_j \psi_j(\theta)$
- Do: Galerkin testing with the same basis functions:
  Get a system of equations, where we solve for u: $Au = b$, where
  $A_{ij} = \langle \psi_i(\theta) a_0 \psi_j(\theta) + \psi_i(\theta)\alpha\theta\psi_j(\theta)\rangle = a_0\gamma_i\delta_{ij} + \alpha\langle\psi_i(\theta)\theta\psi_j(\theta)\rangle$
  and $b_j = \langle \psi_i(\theta) b(\theta)\rangle$

So: $\langle u\rangle = \int u(\theta)\rho(\theta)d\theta = \sum_i u_i \langle\psi_i(\theta)\rangle$ and $\langle u^2\rangle = \sum_i u_i^2\gamma_i$.
$\implies$ std. dev. in u can be computed : $\sqrt{\langle u^2\rangle - \langle u\rangle^2}$
$\implies$ uncertainty **quantified**!

Random inputs? (More than just a collection of RVs!)

Example: A random surface – adjacent points are not independent of each other, there is some correlation:

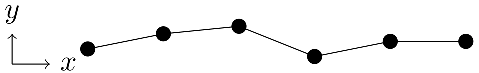


The Kosambi-Karhunen-Loeve (KL) expansion is widely used to represent random processes: $s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$

- $s_0(x)$ is the mean of the random process
- η, f solve this eigenvalue problem: $\int C(i,j) f_k(j) dj = \eta_k f_k(i)$ where $C(i,j) = \text{cov}(z_i, z_j)$ is the correlation between two RVs, z_i, z_j
- $z(\theta)$ represents mutually uncorrelated normal RVs ($\langle z_k \rangle = 0$)
- Expansion truncated to d terms in practice

Random inputs? (More than just a collection of RVs!)

Example: A random surface – adjacent points are not independent of each other, there is some correlation:

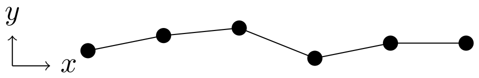


The Kosambi-Karhunen-Loeve (KL) expansion is widely used to represent random processes: $s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$

- $s_0(x)$ is the mean of the random process
- η, f solve this eigenvalue problem: $\int C(i,j) f_k(j) dj = \eta_k f_k(i)$ where $C(i,j) = \text{cov}(z_i, z_j)$ is the correlation between two RVs, z_i, z_j
- $z(\theta)$ represents mutually uncorrelated normal RVs ($\langle z_k \rangle = 0$)
- Expansion truncated to d terms in practice

Random inputs? (More than just a collection of RVs!)

Example: A random surface – adjacent points are not independent of each other, there is some correlation:

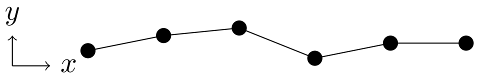


The Kosambi-Karhunen-Loeve (KL) expansion is widely used to represent random processes: $s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$

- $s_0(x)$ is the mean of the random process
- η, f solve this eigenvalue problem: $\int C(i,j) f_k(j) dj = \eta_k f_k(i)$ where $C(i,j) = \text{cov}(z_i, z_j)$ is the correlation between two RVs, z_i, z_j
- $z(\theta)$ represents mutually uncorrelated normal RVs ($\langle z_k \rangle = 0$)
- Expansion truncated to d terms in practice

Random inputs? (More than just a collection of RVs!)

Example: A random surface – adjacent points are not independent of each other, there is some correlation:



The Kosambi-Karhunen-Loeve (KL) expansion is widely used to represent random processes: $s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$

- $s_0(x)$ is the mean of the random process
- η, f solve this eigenvalue problem: $\int C(i,j) f_k(j) dj = \eta_k f_k(i)$ where $C(i,j) = \text{cov}(z_i, z_j)$ is the correlation between two RVs, z_i, z_j
- $z(\theta)$ represents mutually uncorrelated normal RVs ($\langle z_k \rangle = 0$)
- Expansion truncated to d terms in practice

KL expansion for exponential correlation

$$\boxed{s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)}$$

- KL expansion eigenvalues and functions can be analytically calculated in some cases, e.g. exponential correlation function $C(i,j) = \exp(-|i-j|/l)$, (correlation length l).
- Decay rate of eigenvalues depends inversely on the correlation length $\implies$ *more RVs for smaller l.*

KL expansion for exponential correlation

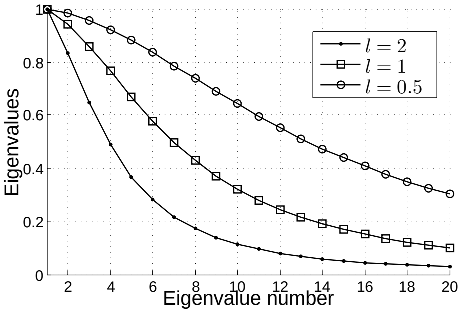
$$s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$$

- KL expansion eigenvalues and functions can be analytically calculated in some cases, e.g. exponential correlation function $C(i,j) = \exp(-|i-j|/l)$, (correlation length l).
- Decay rate of eigenvalues depends inversely on the correlation length $\implies$ *more RVs for smaller l.*

KL expansion for exponential correlation

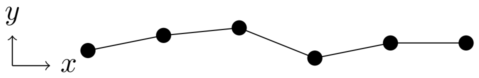
$$s(x,\theta) = s_0(x) + \sum_{k=1}^{\infty} \sqrt{\eta_k} f_k(x) z_k(\theta)$$

- KL expansion eigenvalues and functions can be analytically calculated in some cases, e.g. exponential correlation function $C(i,j) = \exp(-|i-j|/l)$, (correlation length l).
- Decay rate of eigenvalues depends inversely on the correlation length $\implies$ *more RVs for smaller l.*



The case of multiple random variables with correlation

Consider again the random rough surface ...

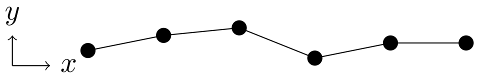


given by the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$
consists of *multiple* random variables, z_i.

- Multivariate gPC (i.e. tensor product of univariate gPC)
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1) \dots \psi_{i_d}(z_d), \quad 0 \le \sum_{j=1}^{d} i_j \le n$ (max degree)
  where $\mathbf{i} = (i_1, \dots, \_d)$: indices, $\vec{z} = (z_1, \dots, z_d)$ RVs
  Belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
- As in univariate case, proceed by Galerkin method and compute mean, variance, etc

The case of multiple random variables with correlation

Consider again the random rough surface ...

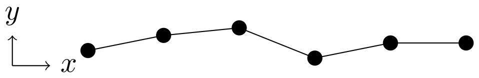


given by the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$ consists of *multiple* random variables, z_i.

- Multivariate gPC (i.e. tensor product of univariate gPC)
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1)\dots\psi_{i_d}(z_d), \quad 0 \leq \sum_{j=1}^{d} i_j \leq n$ (max degree)
  where $\mathbf{i} = (i_1, \dots, \_d)$: indices, $\vec{z} = (z_1, \dots, z_d)$ RVs
  Belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
- As in univariate case, proceed by Galerkin method and compute mean, variance, etc

The case of multiple random variables with correlation

Consider again the random rough surface ...



given by the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$
consists of *multiple* random variables, z_i.

- Multivariate gPC (i.e. tensor product of univariate gPC)
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1) \dots \psi_{i_d}(z_d), \quad 0 \leq \sum_{j=1}^{d} i_j \leq n$ (max degree)
  where $\mathbf{i} = (i_1, \dots, \_d)$: indices, $\vec{z} = (z_1, \dots, z_d)$ RVs
  Belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
- As in univariate case, proceed by Galerkin method and compute mean, variance, etc

Changing gears ...

So far ...

This completes an overview of stochastic computation:

1. Monte Carlo (MC)
2. Stochastic Collocation (SC)
3. Stochastic Galerkin (SG) using generalized Polynomial Chaos

Moving on ...

To make matters more concrete, consider the problem of computing the electromagnetic scattering from a random rough surface:
e.g. seen in microwave remote sensing, [MC][a], [SC,SG][b]

[a]Khankhoje et al. "Computation of radar scattering from heterogeneous rough soil using the finite element method", 2013 IEEE TGRS

[b]Khankhoje et al. "Stochastic Solutions to Rough Surface Scattering using the finite element method", 2017 IEEE TAP

Changing gears ...

So far ...

This completes an overview of stochastic computation:

1. Monte Carlo (MC)
2. Stochastic Collocation (SC)
3. Stochastic Galerkin (SG) using generalized Polynomial Chaos

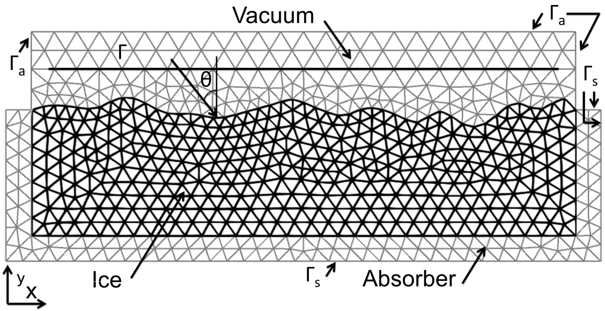
Moving on ...

To make matters more concrete, consider the problem of computing the electromagnetic scattering from a random rough surface:
e.g. seen in microwave remote sensing, [MC][a], [SC,SG][b]

[a]Khankhoje et al. "Computation of radar scattering from heterogeneous rough soil using the finite element method", 2013 IEEE TGRS

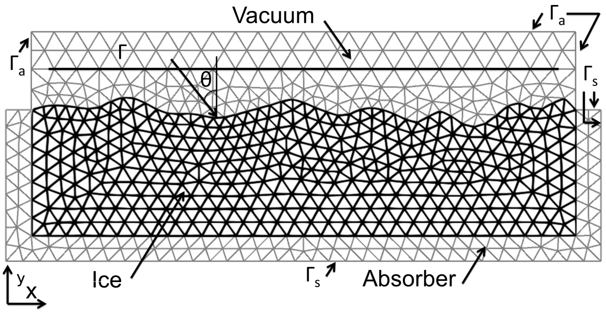
[b]Khankhoje et al. "Stochastic Solutions to Rough Surface Scattering using the finite element method", 2017 IEEE TAP

Traditional FEM setup for rough surface scattering



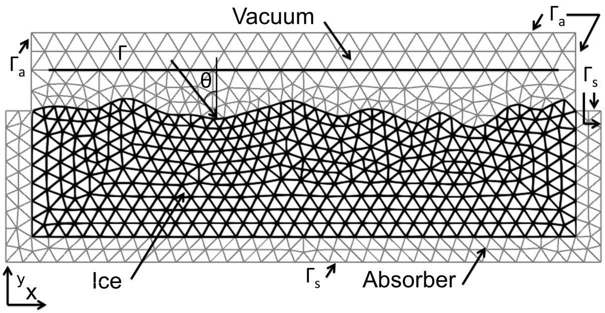
- Random rough surface instance generated, and the domain meshed
- Based on incident field, Radar cross-section (RCS) computed
- Above steps repeated for several (≈ 100) instances
- Quite inefficient!

Traditional FEM setup for rough surface scattering



- Random rough surface instance generated, and the domain meshed
- Based on incident field, Radar cross-section (RCS) computed
- Above steps repeated for several (≈ 100) instances
- Quite inefficient!

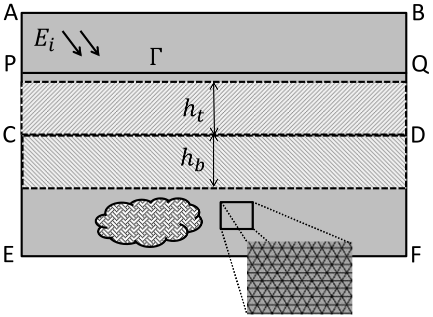
Traditional FEM setup for rough surface scattering



- Random rough surface instance generated, and the domain meshed
- Based on incident field, Radar cross-section (RCS) computed
- Above steps repeated for several (≈ 100) instances
- Quite inefficient!

Handle the rough surface intelligently

Note that the mesh need only change near the surface ...



Partition the domain into parts that can move, and those that need not

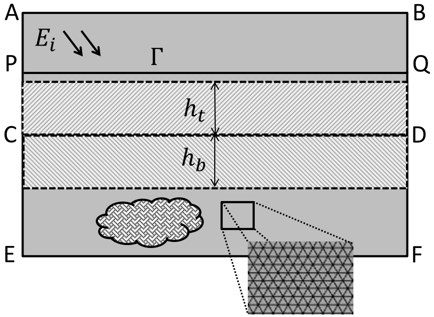
- Let $s(x)$ define rough surface (e.g. KL expansion)
- Move each node smoothly within 'sandwich' region: $y \to y + \Delta y$

$$\Delta y = \begin{cases} s(x)\left(\frac{h_t - y}{h_t}\right), 0 < y < h_t \\ s(x)\left(\frac{y + h_b}{h_b}\right), -h_b < y < 0 \end{cases}$$

- CD will deform to rough surface
- Zero deformation by the time $y = h_t$ or $y = -h_b$

Handle the rough surface intelligently

Note that the mesh need only change near the surface ...



Partition the domain into parts that can move, and those that need not

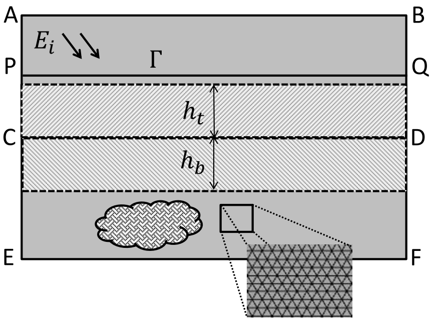
- Let $s(x)$ define rough surface (e.g. KL expansion)
- Move each node smoothly within 'sandwich' region: $y \to y + \Delta y$

$$\Delta y = \begin{cases} s(x)(\frac{h_t - y}{h_t}), \, 0 < y < h_t \\ s(x)(\frac{y + h_b}{h_b}), -h_b < y < 0 \end{cases}$$

- CD will deform to rough surface
- Zero deformation by the time $y = h_t$ or $y = -h_b$

Handle the rough surface intelligently

Note that the mesh need only change near the surface ...



Partition the domain into parts that can move, and those that need not

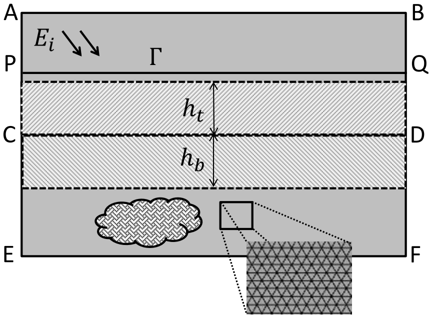
- Let $s(x)$ define rough surface (e.g. KL expansion)
- Move each node smoothly within 'sandwich' region: $y \rightarrow y + \Delta y$

$$\Delta y = \begin{cases} s(x)(\frac{h_t - y}{h_t}), \; 0 < y < h_t \\ s(x)(\frac{y + h_b}{h_b}), -h_b < y < 0 \end{cases}$$

- CD will deform to rough surface
- Zero deformation by the time $y = h_t$ or $y = -h_b$

Handle the rough surface intelligently

Note that the mesh need only change near the surface ...



Partition the domain into parts that can move, and those that need not

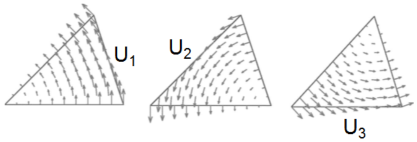
- Let $s(x)$ define rough surface (e.g. KL expansion)
- Move each node smoothly within 'sandwich' region: $y \to y + \Delta y$

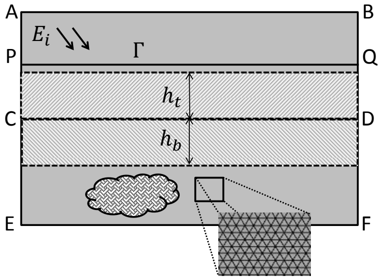
$$\Delta y = \begin{cases} s(x)(\frac{h_t - y}{h_t}), \; 0 < y < h_t \\ s(x)(\frac{y + h_b}{h_b}), -h_b < y < 0 \end{cases}$$

- CD will deform to rough surface
- Zero deformation by the time $y = h_t$ or $y = -h_b$

The standard FEM recipe: deterministic solver

2D vector FE basis functions $\vec{W}$





Maxwell's equations in weak form:

$$\int \vec{W} \cdot [\nabla \times (\tfrac{1}{\epsilon_r} \nabla \times \vec{H}) - k_0^2 \mu_r \vec{H}]\, dS = 0$$

- First order absorbing boundary conditions on $A - B - F - E - A$
- Performing Galerkin testing

$$A\mathbf{u} = \mathbf{b},\ A \in \mathbb{C}^{m \times m}, \mathbf{u}, \mathbf{b} \in \mathbb{C}^m,$$

$$A_{pq} = \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\, \nu_p(\vec{r}_p),$$

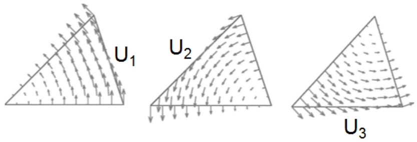
$b_p = \tau_p(\vec{r}_p)$, where

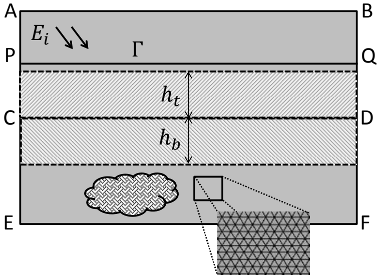
$\vec{r}_e = (x_i, x_j, x_k, y_i, y_j, y_k)$, e^{th}ele

$\vec{r}_p = (x_i, x_j, y_i, y_j)$, p^{th}edge

The standard FEM recipe: deterministic solver

2D vector FE basis functions $\vec{W}$





Maxwell's equations in weak form:

$$\int \vec{W} \cdot [\nabla \times (\frac{1}{\epsilon_r} \nabla \times \vec{H}) - k_0^2 \mu_r \vec{H}]\, dS = 0$$

- First order absorbing boundary conditions on $A - B - F - E - A$
- Performing Galerkin testing

$$A\mathbf{u} = \mathbf{b},\ A \in \mathbb{C}^{m \times m}, \mathbf{u}, \mathbf{b} \in \mathbb{C}^m,$$

$$A_{pq} = \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\, \nu_p(\vec{r}_p),$$

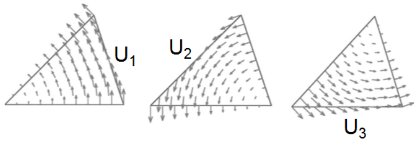
$b_p = \tau_p(\vec{r}_p)$, where

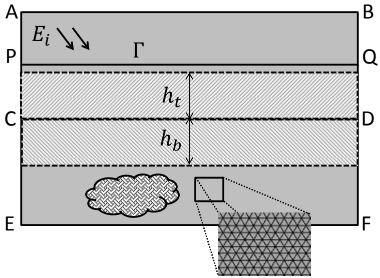
$\vec{r}_e = (x_i, x_j, x_k, y_i, y_j, y_k)$, e^{th}ele

$\vec{r}_p = (x_i, x_j, y_i, y_j)$, p^{th}edge

The standard FEM recipe: deterministic solver

2D vector FE basis functions $\vec{W}$





Maxwell's equations in weak form:

$$\int \vec{W} \cdot [\nabla \times (\tfrac{1}{\epsilon_r} \nabla \times \vec{H}) - k_0^2 \mu_r \vec{H}]\, dS = 0$$

- First order absorbing boundary conditions on $A - B - F - E - A$
- Performing Galerkin testing

$$A\mathbf{u} = \mathbf{b},\ A \in \mathbb{C}^{m \times m}, \mathbf{u}, \mathbf{b} \in \mathbb{C}^m,$$

$$A_{pq} = \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\, \nu_p(\vec{r}_p),$$

$$b_p = \tau_p(\vec{r}_p), \quad \text{where}$$

$$\vec{r}_e = (x_i, x_j, x_k, y_i, y_j, y_k), e^{\text{th}}\text{ele}$$

$$\vec{r}_p = (x_i, x_j, y_i, y_j), p^{\text{th}}\text{edge}$$

The standard FEM recipe: deterministic solver

Quantity of interest: radar cross-section $\sigma_{2D} = \lim_{r\to\infty} 2\pi r \left|\frac{E_z^f}{E_z^i}\right|^2$

$$E_z^f(\vec{r}) = \sqrt{\frac{k_0}{8\pi}}\frac{e^{-i(k_0 r-\pi/4)}}{\sqrt{r}} \times \oint \hat{z}\cdot(\hat{r}\times\vec{M}(\vec{r'}) + Z_0\mu_r\hat{r}\times\hat{r}\times\vec{J}(\vec{r'}))e^{ik_0\hat{r}\cdot\vec{r'}}dl'$$

1. *Start* with a "blank" mesh: rough surface is flat
2. *Deform* mesh "virtually" using KL expansion for rough surface
3. *Compute* σ_{2D} for this mesh
4. *Reset* the mesh and go to (**2**) till convergence (≈ 100 times)

The standard FEM recipe: deterministic solver

Quantity of interest: radar cross-section $\sigma_{2D} = \lim_{r\to\infty} 2\pi r \left| \frac{E_z^f}{E_z^i} \right|^2$

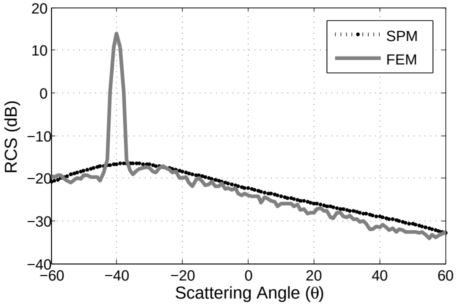
$$E_z^f(\vec{r}) = \sqrt{\frac{k_0}{8\pi}} \frac{e^{-i(k_0 r - \pi/4)}}{\sqrt{r}} \times \oint \hat{z} \cdot (\hat{r} \times \vec{M}(\vec{r'}) + Z_0 \mu_r \hat{r} \times \hat{r} \times \vec{J}(\vec{r'})) e^{ik_0 \hat{r}\cdot\vec{r'}} dl'$$

1. *Start* with a “blank” mesh: rough surface is flat
2. *Deform* mesh “virtually” using KL expansion for rough surface
3. *Compute* σ_{2D} for this mesh
4. *Reset* the mesh and go to **(2)** till convergence (≈ 100 times)

The standard FEM recipe: deterministic solver

Quantity of interest: radar cross-section $\sigma_{2D} = \lim_{r\to\infty} 2\pi r \left|\frac{E_z^f}{E_z^i}\right|^2$

$$E_z^f(\vec{r}) = \sqrt{\frac{k_0}{8\pi}} \frac{e^{-i(k_0 r - \pi/4)}}{\sqrt{r}} \times \oint \hat{z}\cdot(\hat{r}\times\vec{M}(\vec{r'}) + Z_0\mu_r\hat{r}\times\hat{r}\times\vec{J}(\vec{r'}))e^{ik_0\hat{r}\cdot\vec{r'}}dl'$$



1. *Start* with a "blank" mesh: rough surface is flat
2. *Deform* mesh "virtually" using KL expansion for rough surface
3. *Compute* σ_{2D} for this mesh
4. *Reset* the mesh and go to **(2)** till convergence (≈ 100 times)

What is deterministic here?

Input to the FEM solver:

1. Incidence angle, $\epsilon(r)$ for object
2. Set of d normal RVs, $\{z_k\}$, to use in the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$

The FEM is run for a specified surface, hence "deterministic" solver

- Monte Carlo process converges as $\mathbb{O}(1/\sqrt{n_{mc}})$, *independent* of d. (Recall: d depends on the correlation length of the surface) i.e. $\langle\sigma\rangle = \sum_{i=1}^{n_{mc}} \sigma(\vec{r}, \vec{z}_i)/n_{mc}$
- Can we keep the same solver, but have faster convergence?

Possibly: Let's try stochastic collocation

What is deterministic here?

Input to the FEM solver:

1. Incidence angle, $\epsilon(r)$ for object
2. Set of d normal RVs, $\{z_k\}$, to use in the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$

The FEM is run for a specified surface, hence "deterministic" solver

- Monte Carlo process converges as $\mathbb{O}(1/\sqrt{n_{mc}})$, *independent* of d. (Recall: d depends on the correlation length of the surface) i.e. $\langle\sigma\rangle = \sum_{i=1}^{n_{mc}} \sigma(\vec{r}, \vec{z}_i)/n_{mc}$
- Can we keep the same solver, but have faster convergence?

Possibly: Let's try stochastic collocation

What is deterministic here?

Input to the FEM solver:

1. Incidence angle, $\epsilon(r)$ for object
2. Set of d normal RVs, $\{z_k\}$, to use in the KL expansion: $s(x,\theta) = s_0(x) + \sum_{k=1}^{d} \sqrt{\eta_k} f_k(x) z_k(\theta)$

The FEM is run for a specified surface, hence "deterministic" solver

- Monte Carlo process converges as $\mathbb{O}(1/\sqrt{n_{mc}})$, *independent* of d. (Recall: d depends on the correlation length of the surface) i.e. $\langle\sigma\rangle = \sum_{i=1}^{n_{mc}} \sigma(\vec{r}, \vec{z}_i)/n_{mc}$
- Can we keep the same solver, but have faster convergence?

Possibly: Let's try stochastic collocation

From Monte Carlo to Stochastic Collocation

1. Construct multivariate pdf of the d random normal variables: $\rho(\vec{z}) = \prod_{j=1}^{d} \rho_j(z_j)$ over domain $\mathcal{D}^d$, $\mathcal{D} = (-\infty, \infty)$
2. Express expected value as $\langle \sigma \rangle = \int_{\mathcal{D}^d} \sigma(\vec{r}, \vec{z}) \rho(\vec{z}) d\vec{z}$
3. Consider n_{sc} evals of the above integral at predecided quadrature points, $\vec{z}_i$.
4. Express $\sigma(\vec{r}, \vec{z})$ in terms of interpolating multivariate polynomials (e.g. Langrage) $\{P^{(i)}(\vec{z})\}_{i=1}^{d}$ giving $\sigma(\vec{r}, \vec{z}) = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) P^{(i)}(\vec{z})$
5. Finally, $\langle \sigma \rangle = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) \alpha_i$ where $\alpha_i = \int_{\mathcal{D}^d} \rho(\vec{z}) P^{(i)}(\vec{z}) d\vec{z}$

From Monte Carlo to Stochastic Collocation

1. Construct multivariate pdf of the d random normal variables: $\rho(\vec{z}) = \prod_{j=1}^{d} \rho_j(z_j)$ over domain $\mathcal{D}^d$, $\mathcal{D} = (-\infty, \infty)$
2. Express expected value as $\langle \sigma \rangle = \int_{\mathcal{D}^d} \sigma(\vec{r}, \vec{z}) \rho(\vec{z}) d\vec{z}$
3. Consider n_{sc} evals of the above integral at predecided quadrature points, $\vec{z_i}$.
4. Express $\sigma(\vec{r}, \vec{z})$ in terms of interpolating multivariate polynomials (e.g. Langrage) $\{P^{(i)}(\vec{z})\}_{i=1}^{d}$ giving $\sigma(\vec{r}, \vec{z}) = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) P^{(i)}(\vec{z})$
5. Finally, $\langle \sigma \rangle = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) \alpha_i$ where $\alpha_i = \int_{\mathcal{D}^d} \rho(\vec{z}) P^{(i)}(\vec{z}) d\vec{z}$

From Monte Carlo to Stochastic Collocation

1. Construct multivariate pdf of the d random normal variables: $\rho(\vec{z}) = \prod_{j=1}^{d} \rho_j(z_j)$ over domain $\mathcal{D}^d$, $\mathcal{D} = (-\infty, \infty)$
2. Express expected value as $\langle \sigma \rangle = \int_{\mathcal{D}^d} \sigma(\vec{r}, \vec{z}) \rho(\vec{z}) d\vec{z}$
3. Consider n_{sc} evals of the above integral at predecided quadrature points, $\vec{z_i}$.
4. Express $\sigma(\vec{r}, \vec{z})$ in terms of interpolating multivariate polynomials (e.g. Langrage) $\{P^{(i)}(\vec{z})\}_{i=1}^{d}$ giving $\sigma(\vec{r}, \vec{z}) = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) P^{(i)}(\vec{z})$
5. Finally, $\langle \sigma \rangle = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i) \alpha_i$ where $\alpha_i = \int_{\mathcal{D}^d} \rho(\vec{z}) P^{(i)}(\vec{z}) d\vec{z}$

Stochastic Collocation for rough surface scattering

SC recipe: $\langle\sigma\rangle = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i)\alpha_i$

This just looks like the old MC formula! $\langle\sigma\rangle = \sum_{i=1}^{n_{mc}} \sigma(\vec{r}, \vec{z}_i)/n_{mc}$
So, we can use the **same** solver for $\sigma(\vec{r}, \vec{z}_i)$

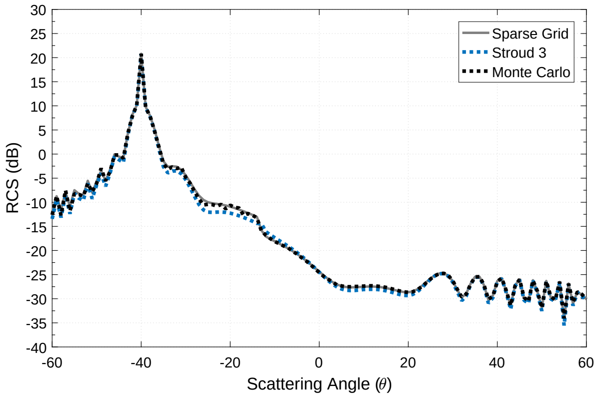
Results
How many function evals using sparse grid? $\approx 2^l d^l / l$
$\leftarrow$ Compare MC and SC for $d = 50, l = 1$
Evals:
MC=100, SC=101

Stochastic Collocation for rough surface scattering

SC recipe: $\langle\sigma\rangle = \sum_{i=1}^{n_{sc}} \sigma(\vec{r}, \vec{z}_i)\alpha_i$

This just looks like the old MC formula! $\langle\sigma\rangle = \sum_{i=1}^{n_{mc}} \sigma(\vec{r}, \vec{z}_i)/n_{mc}$

So, we can use the **same** solver for $\sigma(\vec{r}, \vec{z}_i)$



Results

How many function evals using sparse grid? $\approx 2^l d^l / l$

$\leftarrow$ Compare MC and SC for $d = 50, l = 1$

Evals:

MC=100, SC=101

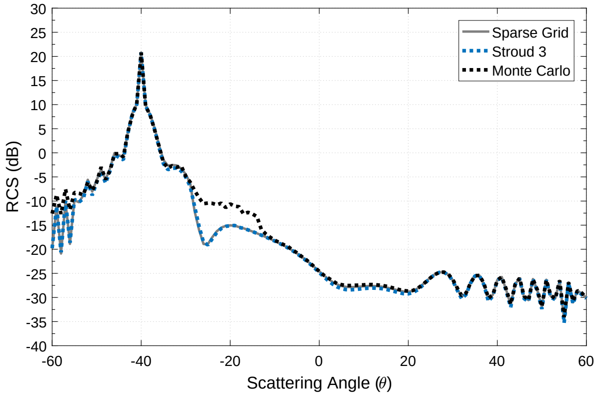
The devil is in the details

For the same surface as before, what happens if you reduce d?

$\leftarrow$ Compare MC and SC for $d = 20, l = 1$
Evals:
MC=100, SC=41

The devil is in the details

For the same surface as before, what happens if you reduce d?



$\leftarrow$ Compare MC and SC for $d = 20, l = 1$
Evals:
MC=100, SC=41

Conclusions about Stochastic Collocation

- Given surface needs to be accurately represented by the finite KL expansion, i.e. there is an optimal d.
- For MC, found that ≈ 100 iterations give convergence to within 1dB.
- The cheapest SC would have level $l = 1$, giving $n_{sc} \approx 2d$
  $\implies$ if the surface requires $d > 50$, MC is better
- This critical number may decrease if we employ anisotropic sparse grids. Why? KL eigen values decay and not all dims are as important.

Conclusions about Stochastic Collocation

- Given surface needs to be accurately represented by the finite KL expansion, i.e. there is an $\underline{\text{optimal } d}$.
- For MC, found that ≈ 100 iterations give convergence to within 1dB.
- The cheapest SC would have level $l = 1$, giving $n_{sc} \approx 2d$

  $\implies$ if the surface requires $d > 50$, MC is better
- This critical number $\underline{\text{may decrease}}$ if we employ anisotropic sparse grids. Why? KL eigen values decay and not all dims are as important.

Conclusions about Stochastic Collocation

- Given surface needs to be accurately represented by the finite KL expansion, i.e. there is an $\underline{\text{optimal } d}$.
- For MC, found that ≈ 100 iterations give convergence to within 1dB.
- The cheapest SC would have level $l = 1$, giving $n_{sc} \approx 2d$

  $\implies$ if the surface requires $d > 50$, MC is better
- This critical number $\underline{\text{may decrease}}$ if we employ anisotropic sparse grids. Why? KL eigen values decay and not all dims are as important.

Moving over to Stochastic Galerkin

- Recall the earlier (scalar) example of $a\,u = b$, where we replaced $a \to a_0 + \alpha\theta$, where θ was a uniform RV.
- In the case of the FEM, we have a sparse matrix equation $A\mathbf{u} = \mathbf{b}$ to solve. Now, A_{pq} must be transformed as per the KL expansion for the surface. Recall:

$$\begin{aligned}
A_{pq} &= \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\,\nu_p(\vec{r}_p),\\
b_p &= \tau_p(\vec{r}_p), \quad \text{where}\\
\vec{r}_e &= (x_i, x_j, x_k, y_i, y_j, y_k), e^{\text{th}}\text{ele}\\
\vec{r}_p &= (x_i, x_j, y_i, y_j), p^{\text{th}}\text{edge}
\end{aligned}$$

Use our mesh deformation scheme, which transformed $y \to y + \Delta y$. This leads to $A_{pq} \to \tilde{A}_{pq}$, $b_p \to \tilde{b}_p$

Moving over to Stochastic Galerkin

- Recall the earlier (scalar) example of $a\,u = b$, where we replaced $a \to a_0 + \alpha\theta$, where θ was a uniform RV.
- In the case of the FEM, we have a sparse matrix equation $A\mathbf{u} = \mathbf{b}$ to solve. Now, A_{pq} must be transformed as per the KL expansion for the surface. Recall:

$$\begin{aligned} A_{pq} &= \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\,\nu_p(\vec{r}_p), \\ b_p &= \tau_p(\vec{r}_p), \quad \text{where} \\ \vec{r}_e &= (x_i, x_j, x_k, y_i, y_j, y_k), e^{\text{th}}\text{ele} \\ \vec{r}_p &= (x_i, x_j, y_i, y_j), p^{\text{th}}\text{edge} \end{aligned}$$

Use our mesh deformation scheme, which transformed $y \to y + \Delta y$. This leads to $A_{pq} \to \tilde{A}_{pq}$, $b_p \to \tilde{b}_p$

Moving over to Stochastic Galerkin

- Recall the earlier (scalar) example of $a\,u = b$, where we replaced $a \to a_0 + \alpha\theta$, where θ was a uniform RV.
- In the case of the FEM, we have a sparse matrix equation $A\mathbf{u} = \mathbf{b}$ to solve. Now, A_{pq} must be transformed as per the KL expansion for the surface. Recall:

$$\begin{aligned}
A_{pq} &= \sum_e \alpha_{e,pq}(\vec{r}_e) + \delta_{pq}\,\nu_p(\vec{r}_p),\\
b_p &= \tau_p(\vec{r}_p), \quad \text{where}\\
\vec{r}_e &= (x_i, x_j, x_k, y_i, y_j, y_k), e^{\text{th}}\text{ele}\\
\vec{r}_p &= (x_i, x_j, y_i, y_j), p^{\text{th}}\text{edge}
\end{aligned}$$

Use our mesh deformation scheme, which transformed $y \to y + \Delta y$. This leads to $A_{pq} \to \tilde{A}_{pq}$, $b_p \to \tilde{b}_p$

Steps in Stochastic Galerkin

1. Accomplish the perturbations by Taylor expanding to second order: (1^{st}, 2^{nd} derivatives are computed by finite differences)

$$\tilde{A}_{pq} = \beta_{pq} + \sum_{i=1}^{d} \beta_{pq}^{(i)} z_i + \sum_{i,j=1}^{d} \beta_{pq}^{(i,j)} z_i z_j, \leftarrow \text{matrices } m \times m$$
$$\tilde{b}_p = \zeta_p + \sum_{i=1}^{d} \zeta_p^{(i)} z_i, \leftarrow \text{vectors } m \times 1$$

2. Now do Galerkin testing in the basis of orthogonal polynomials corresponding to the PDF of the RVs:
   - Expand each $u \to \sum_{i=1}^{w} u_i \Psi_i(\vec{z})$
   - Take inner products with $\Psi_i(\vec{z})$ on both sides of $A\mathbf{u} = \mathbf{b}$
3. Resulted in another bigger system of equations: $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

Steps in Stochastic Galerkin

1. Accomplish the perturbations by Taylor expanding to second order: (1^{st}, 2^{nd} derivatives are computed by finite differences)

$$\tilde{A}_{pq} = \beta_{pq} + \sum_{i=1}^{d} \beta_{pq}^{(i)} z_i + \sum_{i,j=1}^{d} \beta_{pq}^{(i,j)} z_i z_j, \leftarrow \text{matrices } m \times m$$
$$\tilde{b}_p = \zeta_p + \sum_{i=1}^{d} \zeta_p^{(i)} z_i, \leftarrow \text{vectors } m \times 1$$

2. Now do Galerkin testing in the basis of orthogonal polynomials corresponding to the PDF of the RVs:
   - Expand each $u \rightarrow \sum_{i=1}^{w} u_i \Psi_i(\vec{z})$
   - Take inner products with $\Psi_i(\vec{z})$ on both sides of $A\mathbf{u} = \mathbf{b}$

3. Resulted in another bigger system of equations: $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

Steps in Stochastic Galerkin

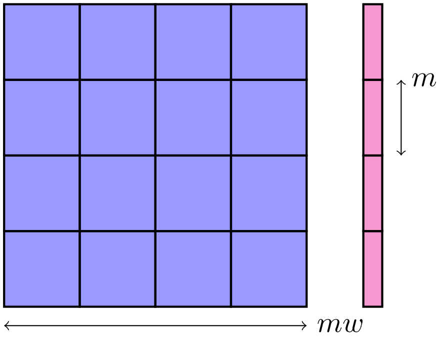
1. Accomplish the perturbations by Taylor expanding to second order: (1^{st}, 2^{nd} derivatives are computed by finite differences)

$$\boxed{\begin{array}{l}\tilde{A}_{pq} = \beta_{pq} + \sum_{i=1}^{d} \beta_{pq}^{(i)} z_i + \sum_{i,j=1}^{d} \beta_{pq}^{(i,j)} z_i z_j, \leftarrow \text{matrices } m \times m \\ \tilde{b}_p = \zeta_p + \sum_{i=1}^{d} \zeta_p^{(i)} z_i, \leftarrow \text{vectors } m \times 1\end{array}}$$

2. Now do Galerkin testing in the basis of orthogonal polynomials corresponding to the PDF of the RVs:
   - Expand each $u \to \sum_{i=1}^{w} u_i \Psi_i(\vec{z})$
   - Take inner products with $\Psi_i(\vec{z})$ on both sides of $A\mathbf{u} = \mathbf{b}$
3. Resulted in another bigger system of equations:
   $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

Details of Stochastic Galerkin

- What is a the structure of: $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw\times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

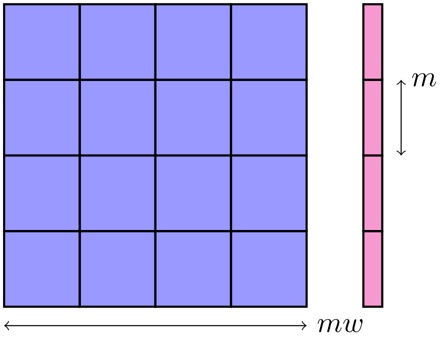


- Each block is sparse, multiplied by stochastic inner products like: $E_{ab} = \langle \Psi_a(\vec{z})\Psi_b(\vec{z})\rangle$, $E_{ab}^{(i)} = \langle \Psi_a(\vec{z}) z_i \Psi_b(\vec{z})\rangle$, $E_{ab}^{(i,j)} = \langle \Psi_a(\vec{z})\, z_i\, z_j\, \Psi_b(\vec{z})\rangle$,
- Needs to be solved only once

Once we solve this equation, we get an expression for the far field, and from that we get the RCS as $\langle |E^f(\vec{r})|^2\rangle$.

Details of Stochastic Galerkin

- What is a the structure of: $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw\times mw}$, $\mathbf{v},\mathbf{g} \in \mathbb{C}^{mw}$

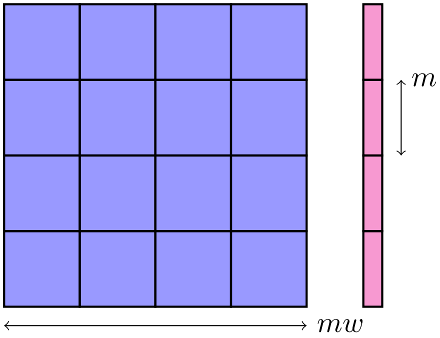


- Each block is sparse, multiplied by stochastic inner products like: $E_{ab} = \langle \Psi_a(\vec{z})\Psi_b(\vec{z})\rangle$, $E_{ab}^{(i)} = \langle \Psi_a(\vec{z}) z_i \Psi_b(\vec{z})\rangle$, $E_{ab}^{(i,j)} = \langle \Psi_a(\vec{z})\, z_i\, z_j\, \Psi_b(\vec{z})\rangle$,
- Needs to be solved only once

Once we solve this equation, we get an expression for the far field, and from that we get the RCS as $\langle |E^f(\vec{r})|^2\rangle$.

Details of Stochastic Galerkin

- What is a the structure of: $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw\times mw}$, $\mathbf{v},\mathbf{g} \in \mathbb{C}^{mw}$



- Each block is sparse, multiplied by stochastic inner products like: $E_{ab} = \langle \Psi_a(\vec{z})\Psi_b(\vec{z})\rangle$, $E_{ab}^{(i)} = \langle \Psi_a(\vec{z}) z_i \Psi_b(\vec{z})\rangle$, $E_{ab}^{(i,j)} = \langle \Psi_a(\vec{z})\, z_i\, z_j\, \Psi_b(\vec{z})\rangle$,
- Needs to be solved only once

Once we solve this equation, we get an expression for the far field, and from that we get the RCS as $\langle |E^f(\vec{r})|^2\rangle$.

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

- Not a "cheap" computation: ~~direct matrix solvers~~, use an **iterative** method: BiCgStab + block-diagonal, mean-based pre-conditioner
- The matrix F is never stored explicitly, instead we only need to be able to compute the **product** of F with another vector for the iterative method to work
- Compute times and memory demands are still much higher than MC/SC. Scope for lot of research in sparse data structures for F, g

SG-gPC	MC
1200s (10 iters), 10 GB RAM	950s (100 surfaces), 150 MB RAM

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

- Not a "cheap" computation: ~~direct matrix solvers~~, use an **iterative** method: BiCgStab + block-diagonal, mean-based pre-conditioner
- The matrix F is never stored explicitly, instead we only need to be able to compute the **product** of F with another vector for the iterative method to work
- Compute times and memory demands are still much higher than MC/SC. Scope for lot of research in sparse data structures for $F, \mathbf{g}$

SG-gPC	MC
1200s (10 iters), 10 GB RAM	950s (100 surfaces), 150 MB RAM

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

- Not a "cheap" computation: ~~direct matrix solvers~~, use an **iterative** method: BiCgStab + block-diagonal, mean-based pre-conditioner
- The matrix F is never stored explicitly, instead we only need to be able to compute the **product** of F with another vector for the iterative method to work
- Compute times and memory demands are still much higher than MC/SC. Scope for lot of research in sparse data structures for $F, \mathbf{g}$

SG-gPC	MC
1200s (10 iters), 10 GB RAM	950s (100 surfaces), 150 MB RAM

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

- How do we choose the number of edges in the FEM formulation, m?
  Horizontally: Long enough to capture several correlation lengths
  Vertically: Sufficient for boundary conditions to be accurate
- How many basis functions to keep, w? Recall:
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1) \dots \psi_{i_d}(z_d), \quad 0 \le \mathbf{i} = \sum_{j=1}^{d} i_j \le n$ (max degree)
  belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
  Choose d: no of KL terms till eigenvalue falls by $1/10$
  Choose n: depends on available computer resources

Values of d v/s surface length (a) and correlation length (l)

$\downarrow l \setminus$ a$\rightarrow$	15	30
1	15	29
0.5	29	58

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw\times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

- How do we choose the number of edges in the FEM formulation, m?
  Horizontally: Long enough to capture several correlation lengths
  Vertically: Sufficient for boundary conditions to be accurate
- How many basis functions to keep, w? Recall:
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1)\dots\psi_{i_d}(z_d), \quad 0 \le \mathbf{i} = \sum_{j=1}^{d} i_j \le n$ (max degree)
  belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
  Choose d: no of KL terms till eigenvalue falls by $1/10$
  Choose n: depends on available computer resources

Values of d v/s surface length (a) and correlation length (l)

$\downarrow l \setminus$ a$\rightarrow$	15	30
1	15	29
0.5	29	58

Computational details of Stochastic Galerkin

In solving $F\,\mathbf{v} = \mathbf{g}$, $F \in \mathbb{C}^{mw \times mw}$, $\mathbf{v}, \mathbf{g} \in \mathbb{C}^{mw}$

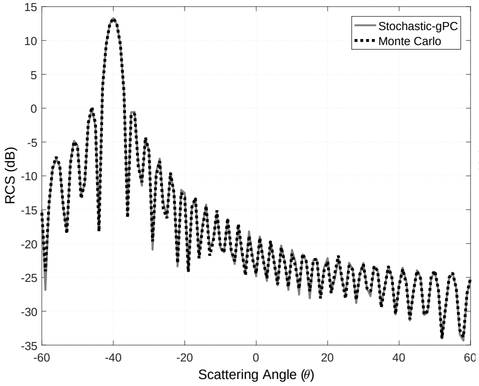
- How do we choose the number of edges in the FEM formulation, m?
  Horizontally: Long enough to capture several correlation lengths
  Vertically: Sufficient for boundary conditions to be accurate
- How many basis functions to keep, w? Recall:
  $\Psi_{\mathbf{i}}(\vec{z}) = \psi_{i_1}(z_1)\ldots\psi_{i_d}(z_d), \quad 0 \le \mathbf{i} = \sum_{j=1}^{d} i_j \le n$ (max degree)
  belong to the space $\mathbb{P}_n^d$ of dimension $w = \binom{n+d}{n}$
  Choose d: no of KL terms till eigenvalue falls by $1/10$
  Choose n: depends on available computer resources

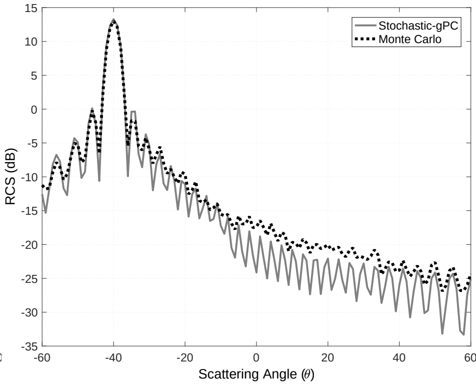
Values of d v/s surface length (a) and correlation length (l)

$\downarrow l \setminus$ a$\rightarrow$	15	30
1	15	29
0.5	29	58

Compare Stochastic Galerkin with Monte Carlo

$h = 0.05/(2\pi)$ $h = 0.20/(2\pi)$





$$\theta_i = 40^\circ, \epsilon_r = 4 - j, a = 20, l = 1, \lambda = 1 \quad d = 15, n = 1$$

Reflections

- SG-gPC is a deal breaker for even small values of d
  $\rightarrow$ Computational innovation required
- For small-moderate values of d, (e.g. $d < 50$) SC methods are a powerful alternative to MC
  $\rightarrow$ Can be improved by anisotropic sparse grids.
- For higher values of d, MC is the still optimal
  $\rightarrow$ Can be improved by Markov chain monte carlo

Reflections

- SG-gPC is a deal breaker for even small values of d
  $\rightarrow$ Computational innovation required
- For small-moderate values of d, (e.g. $d < 50$) SC methods are a powerful alternative to MC
  $\rightarrow$ Can be improved by anisotropic sparse grids.
- For higher values of d, MC is the still optimal
  $\rightarrow$ Can be improved by Markov chain monte carlo

Reflections

- SG-gPC is a deal breaker for even <u>small</u> values of d
  $\rightarrow$ Computational innovation required
- For <u>small-moderate</u> values of d, (e.g. $d < 50$) SC methods are a powerful alternative to MC
  $\rightarrow$ Can be improved by anisotropic sparse grids.
- For <u>higher</u> values of d, MC is the still optimal
  $\rightarrow$ Can be improved by Markov chain monte carlo

Thanks!

Most of this talk was based on:

"Stochastic Solutions to Rough Surface Scattering using the finite element method", Uday K Khankhoje and Shreyas Padhy; *IEEE Transactions on Antennas and Propagation*, 65(08) 2017.

Refer to `http://www.ee.iitm.ac.in/uday` for full-text of relevant papers. Or just Google my name.

E-mail: uday@ee.iitm.ac.in

Aside: Why is Guass quadrature so effective?

- To approximate $\int f(x)w(x)dx$ by an n-point rule, express $f(x)$ in terms of: nth order polynomial $p_n(x)$, a quotient $q(x)$ (degree $< n$), & a remainder $r(x)$ (degree $< n$):
  $$f(x) = q(x) * p_n(x) + r(x) \qquad (1)$$
- Integrate w.r.t. $w(x)$, and use orthogonality property of $p_n(x)$: we get $\int f(x)w(x)dx = \int r(x)w(x)dx$. Next, construct a $n-1$ degree polynomial from n points (Lagrange interpolation) and get:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} r(x_i) \int w(x)L_i(x)dx = \sum_{i=1}^{n} r(x_i)\alpha_i \qquad (2)$$
- Here's where the choice of points x_i is crucial: If x_i is a root of $p_n(x)$, then from (1) it follows that $f(x_i) = r(x_i)$ and from (2) that:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} f(x_i)\alpha_i \qquad (3)$$
- Thus we say that Gauss quadrature is accurate to order $2n-1$

Aside: Why is Guass quadrature so effective?

- To approximate $\int f(x)w(x)dx$ by an n-point rule, express $f(x)$ in terms of: nth order polynomial $p_n(x)$, a quotient $q(x)$ (degree $< n$), & a remainder $r(x)$ (degree $< n$):
$$f(x) = q(x) * p_n(x) + r(x) \qquad (1)$$
- Integrate w.r.t. $w(x)$, and use orthogonality property of $p_n(x)$: we get $\int f(x)w(x)dx = \int r(x)w(x)dx$. Next, construct a $n-1$ degree polynomial from n points (Lagrange interpolation) and get:
$$\int f(x)w(x)dx = \sum_{i=1}^{n} r(x_i) \int w(x)L_i(x)dx = \sum_{i=1}^{n} r(x_i)\alpha_i \qquad (2)$$
- Here's where the choice of points x_i is crucial: If x_i is a root of $p_n(x)$, then from (1) it follows that $f(x_i) = r(x_i)$ and from (2) that:
$$\int f(x)w(x)dx = \sum_{i=1}^{n} f(x_i)\alpha_i \qquad (3)$$
- Thus we say that Gauss quadrature is accurate to order $2n-1$

Aside: Why is Guass quadrature so effective?

- To approximate $\int f(x)w(x)dx$ by an n-point rule, express $f(x)$ in terms of: nth order polynomial $p_n(x)$, a quotient $q(x)$ (degree $< n$), & a remainder $r(x)$ (degree $< n$):
  $$f(x) = q(x) * p_n(x) + r(x) \qquad (1)$$
- Integrate w.r.t. $w(x)$, and use orthogonality property of $p_n(x)$: we get $\int f(x)w(x)dx = \int r(x)w(x)dx$. Next, construct a $n-1$ degree polynomial from n points (Lagrange interpolation) and get:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} r(x_i) \int w(x)L_i(x)dx = \sum_{i=1}^{n} r(x_i)\alpha_i \qquad (2)$$
- Here's where the choice of points x_i is crucial: If x_i is a root of $p_n(x)$, then from (1) it follows that $f(x_i) = r(x_i)$ and from (2) that:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} f(x_i)\alpha_i \qquad (3)$$
- Thus we say that Gauss quadrature is accurate to order $2n-1$

Aside: Why is Guass quadrature so effective?

- To approximate $\int f(x)w(x)dx$ by an n-point rule, express $f(x)$ in terms of: nth order polynomial $p_n(x)$, a quotient $q(x)$ (degree $< n$), & a remainder $r(x)$ (degree $< n$):
  $$f(x) = q(x) * p_n(x) + r(x) \qquad (1)$$
- Integrate w.r.t. $w(x)$, and use orthogonality property of $p_n(x)$: we get $\int f(x)w(x)dx = \int r(x)w(x)dx$. Next, construct a $n-1$ degree polynomial from n points (Lagrange interpolation) and get:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} r(x_i) \int w(x)L_i(x)dx = \sum_{i=1}^{n} r(x_i)\alpha_i \qquad (2)$$
- Here's where the choice of points x_i is crucial: If x_i is a root of $p_n(x)$, then from (1) it follows that $f(x_i) = r(x_i)$ and from (2) that:
  $$\int f(x)w(x)dx = \sum_{i=1}^{n} f(x_i)\alpha_i \qquad (3)$$
- Thus we say that Gauss quadrature is accurate to order $2n-1$