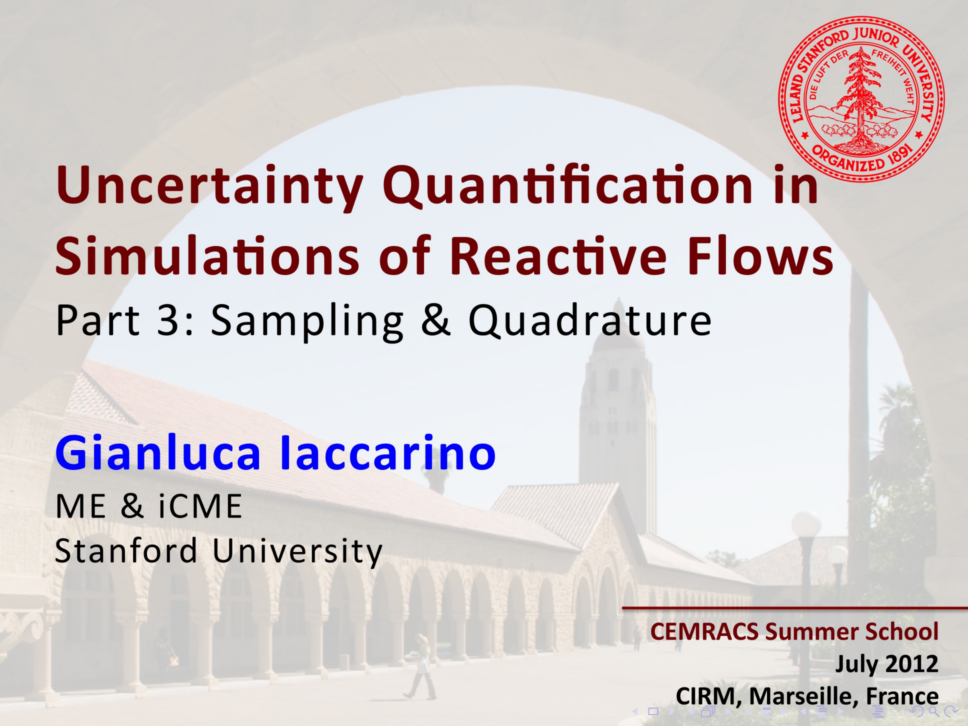


Uncertainty Quantification in Simulations of Reactive Flows

Part 3: Sampling & Quadrature

Gianluca Iaccarino

ME & iCME

Stanford University

CEMRACS Summer School

July 2012

CIRM, Marseille, France

Quadrature for Uncertainty Analysis

Stochastic Collocation

What does quadrature have to do with uncertainty?

Quadrature for Uncertainty Analysis

Stochastic Collocation

What does quadrature have to do with uncertainty?

Assume y is a uniform random variable describing the uncertainty in the input and Q is he quantity of interest

Quadrature for Uncertainty Analysis

Stochastic Collocation

What does quadrature have to do with uncertainty?

Assume y is a uniform random variable describing the uncertainty in the input and Q is he quantity of interest
The mean of Q is

$$\langle Q \rangle = \int_{-1}^{1} Q(y) f_y dy = \frac{1}{2} \int_{-1}^{1} Q(y) dy \quad \text{if y = U(-1,1)}$$

Similarly for the variance, etc.

Quadrature for Uncertainty Analysis

What does quadrature have to do with uncertainty?

Assume *y* is a uniform random variable describing the uncertainty in the input and *Q* is he quantity of interest
The mean of Q is

$$\langle Q \rangle = \int_{-1}^{1} Q(y) f_y dy = \frac{1}{2} \int_{-1}^{1} Q(y) dy \quad \text{if y = U(-1,1)}$$

Similarly for the variance, etc.

Moments of a quantity of interest are integrals in the probability space defined by the uncertain variables!

Stochastic Collocation

- For an input variables that is $U(-1,1)$
- Generate N values of the parameters $y^k \quad k = 1, \ldots, N$: abscissas (the zeros of the Legendre polynomial P_N)

Stochastic Collocation

- For an input variables that is $U(-1, 1)$
- Generate N values of the parameters $y^k \quad k = 1, \dots, N$: abscissas (the zeros of the Legendre polynomial P_N)
- Perform N simulations according to the selected abscissas and obtain $Q(y^k)$

Stochastic Collocation

- For an input variables that is $U(-1,1)$
- Generate N values of the parameters $y^k \quad k = 1, \ldots, N$: abscissas (the zeros of the Legendre polynomial P_N)
- Perform N simulations according to the selected abscissas and obtain $Q(y^k)$
- Compute statistics as weighted sums (the weights are integrals of Lagrange polynomials through the abscissas)

$$\langle Q \rangle = \int_{-1}^{1} Q(y)dy = \sum_{k=1}^{N} Q(y^k)w_k$$

Stochastic Collocation

- For an input variables that is $U(-1,1)$
- Generate N values of the parameters $y^k \quad k = 1, \dots, N$: abscissas (the zeros of the Legendre polynomial P_N)
- Perform N simulations according to the selected abscissas and obtain $Q(y^k)$
- Compute statistics as weighted sums (the weights are integrals of Lagrange polynomials through the abscissas)

$$\langle Q \rangle = \int_{-1}^{1} Q(y) dy = \sum_{k=1}^{N} Q(y^k) w_k$$

No randomness is introduced! but convergence is exponential (cfr. MC)

Non-intrusive Polynomial Chaos

- Quadrature (and sampling) can be used directly to evaluate the statistics of the quantity on interest.
- Another avenue is to use these methods in conjunction with polynomial chaos approaches

Non-intrusive Polynomial Chaos

- Quadrature (and sampling) can be used directly to evaluate the statistics of the quantity on interest.
- Another avenue is to use these methods in conjunction with polynomial chaos approaches
- Reminder: in polynomial chaos (stochastic Galerkin) the solution is expressed as a spectral expansion of the *uncertain* variable(s): $\xi \in \Omega$ as:

$$u(x,t,\xi) = \sum_{i=0}^{P} \underbrace{u_i(x,t)}_{\textit{deterministic}} \underbrace{\psi_i(\xi)}_{\textit{stochastic}}$$

and this expansion is inserted in the governing PDE!

Non-intrusive Polynomial Chaos

- idea apply the Galerkin procedure directly to the formula: $u(x,t,\xi) = \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)$

Non-intrusive Polynomial Chaos

- idea apply the Galerkin procedure directly to the formula: $u(x,t,\xi) = \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)$
- Steps:
  - multiply by $\psi_k(\xi)$
  - integrate over the probability space
  - repeat for each $k = 0, 1, ..., P$
- The result is

$$\int_\Omega u(x,t,\xi)\psi_k(\xi)d\xi = \int_\Omega \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)\psi_k(\xi)d\xi$$

Non-intrusive Polynomial Chaos

- idea apply the Galerkin procedure directly to the formula: $u(x,t,\xi) = \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)$
- Steps:
  - multiply by $\psi_k(\xi)$
  - integrate over the probability space
  - repeat for each $k = 0, 1, ..., P$
- The result is

$$\int_\Omega u(x,t,\xi)\psi_k(\xi)d\xi = \int_\Omega \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)\psi_k(\xi)d\xi$$

- The orthogonality condition $\langle \psi_i \psi_k \rangle = \delta_{ik} h_k$ leads to:

$$\int_\Omega u(x,t,\xi)\psi_k(\xi)d\xi = u_k(x,t)h_k$$

where h_k is a known constant

Non-intrusive Polynomial Chaos

- The conclusion is that we can compute the coefficients of the polynomial chaos expansion

$$u(x,t,\xi) = \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)$$

simply by computing a sequence of integrals

$$u_k(x,t) = \frac{1}{h_k}\int_{\Omega} u(x,t,\xi)\psi_k(\xi)d\xi \qquad k = 0, 1, ..., P$$

Non-intrusive Polynomial Chaos

- The conclusion is that we can compute the coefficients of the polynomial chaos expansion

$$u(x,t,\xi) = \sum_{i=0}^{P} u_i(x,t)\psi_i(\xi)$$

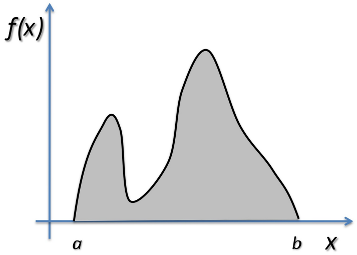
  simply by computing a sequence of integrals

$$u_k(x,t) = \frac{1}{h_k}\int_\Omega u(x,t,\xi)\psi_k(\xi)d\xi \qquad k = 0,1,...,P$$

- Every numerical integration method (Monte Carlo, LHS, quadrature) can be used and only require few (?) evaluations of the solution $u(x,t,\xi)$ of the original problem.

Numerical Integration

- Recall ordinary numerical integration:



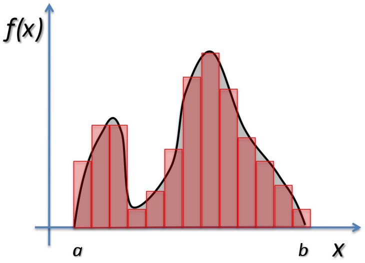
Problem: Compute the integral of $f(x)$ over the interval $[a : b]$.

Numerical Integration

- Evaluate the function at N regular interval $\Delta x = (b-a)/N$
- Midpoint rule (direct summation)

$$A = \sum_{i=1}^{N} f(x_i)\Delta x = \frac{b-a}{N}\sum_{i=1}^{n} f(x_i)$$

with $x_i = a + (i-0.5)\Delta x$ are the *abscissas*



Numerical Integration

d-dimensional case

- Function defined on a $d - \mathit{dimensional}$ interval $([a_1 : b_1], [a_2 : b_2], \ldots, [a_d : b_d])$
- The integral becomes

$$V^{d+1} = \frac{(b_1 - a_1)(b_2 - a_2) \cdots (b_d - a_d)}{N_1 N_2 \cdots N_d} \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_d=1}^{N_d} f(x_i)$$

with $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$

Numerical Integration

d-dimensional case

- Function defined on a $d - dimensional$ interval $([a_1 : b_1], [a_2 : b_2], \ldots, [a_d : b_d])$
- The integral becomes

$$V^{d+1} = \frac{(b_1 - a_1)(b_2 - a_2)\cdots(b_d - a_d)}{N_1 N_2 \cdots N_d} \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_d=1}^{N_d} f(x_i)$$

  with $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$
- We can write the integral more compactly:

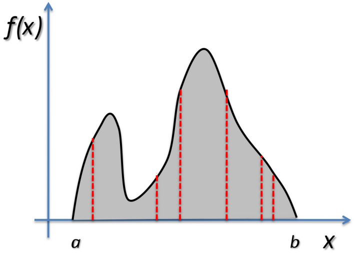
$$V^{d+1} = V^d \frac{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_d=1}^{N_d} f(x_i)}{N}$$

  with $N = N_1 N_2 \cdots N_d$ the total number of points where the function is evaluated

Monte Carlo Integration

- Pick N random d-dimensional vectors $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$ (the x_{i_j} are independent uniform random numbers)
- The desired volume is

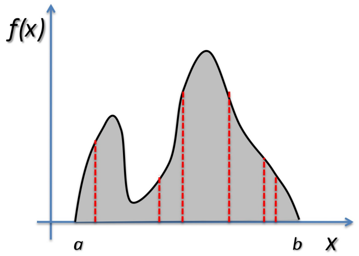
$$V^{d+1} = V^d \frac{\sum_{i=1}^{N} f(x_i)}{N}$$



Monte Carlo Integration

- Pick N random d-dimensional vectors $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$ (the x_{i_j} are independent uniform random numbers)
- The desired volume is

$$V^{d+1} = V^d \frac{\sum_{i=1}^{N} f(x_i)}{N}$$



Compare to: $V^{d+1} = V^d \frac{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_d=1}^{N_d} f(x_i)}{N}$

Monte Carlo Integration

- The difference between mid-point integration and MC is the replacement of *d* nested sums with one, and the random choice of the abscissas.

Monte Carlo Integration

- The difference between mid-point integration and MC is the replacement of d nested sums with one, and the random choice of the abscissas.
- In 1D there is not much difference and indeed using high-order integration (e.g. Simpson rule) the conventional integration can be quite accurate and efficient

Monte Carlo Integration

- The difference between mid-point integration and MC is the replacement of d nested sums with one, and the random choice of the abscissas.
- In 1D there is not much difference and indeed using high-order integration (e.g. Simpson rule) the conventional integration can be quite accurate and efficient
- In Multi D the conventional integration becomes cumbersome and expensive.
    - Assume $N_j = 5$ for all j (this is a low value!), for $d = 10$, we need 5^{10} points to get a reasonable answer

MC versus Conventional Integration

Classical example

Compute the volume of a (hyper-) sphere in d dimensions

- Conventional integration: $N_j = 20$ for all $j \rightarrow V_{NI}$
- MC: $N = 10^5 \rightarrow V_{MC}$

MC versus Conventional Integration

Classical example

Compute the volume of a (hyper-) sphere in d dimensions

- Conventional integration: $N_j = 20$ for all $j \rightarrow V_{NI}$
- MC: $N = 10^5 \rightarrow V_{MC}$

d	sec	V_{NI}/V_e	sec	V_{MC}/V_e
2	0.00	1.0034	0.01	1.0006
3	0.00	0.9964	0.07	1.0002
4	0.00	0.9934	0.08	0.9996
5	0.02	0.9951	0.10	1.0028
6	0.30	0.9956	0.13	1.0012
7	5.02	0.9885	0.15	0.9968
8	89.9	0.9755	0.17	0.9973
9	1320	1.0307	0.20	1.0062

Monte Carlo method

The MC integration can be rewritten as:

$$V^{d+1} = V^d \frac{\sum_{i=1}^{N} f(x_i)}{N} = V^d \langle f \rangle$$

Monte Carlo method

The MC integration can be rewritten as:

$$V^{d+1} = V^d \frac{\sum_{i=1}^{N} f(x_i)}{N} = V^d \langle f \rangle$$

The integration error can be related to the error of the average

$$S^2 = \frac{1}{N-1}\left[\left(\sum_{i=1}^{N} f(x_i)^2\right) - \langle f \rangle^2\right] \approx \frac{1}{N}\left(\langle f^2 \rangle - \langle f \rangle^2\right)$$

Monte Carlo method

The MC integration can be rewritten as:

$$V^{d+1} = V^d \frac{\sum_{i=1}^N f(x_i)}{N} = V^d \langle f \rangle$$

The integration error can be related to the error of the average

$$S^2 = \frac{1}{N-1}\left[\left(\sum_{i=1}^N f(x_i)^2\right) - \langle f \rangle^2\right] \approx \frac{1}{N}\left(\langle f^2 \rangle - \langle f \rangle^2\right)$$

Monte Carlo integration error is unbiased and can be estimated as:

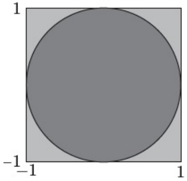
$$\int f dV = V\langle f \rangle \pm \alpha V \sqrt{\frac{1}{N}\left(\langle f^2 \rangle - \langle f \rangle^2\right)}$$

Larger α imply broader confidence that the *true* value is included in the error bar.

Monte Carlo method

Computing π

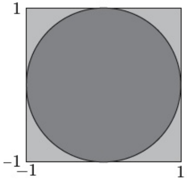
- Assume *uniform rain* on the square $[-1, 1] \times [-1 : 1] \rightarrow x, y \approx U[-1 : 1]$



Monte Carlo method

Computing π

- Assume *uniform rain* on the square $[-1,1] \times [-1:1] \rightarrow x, y \approx U[-1:1]$



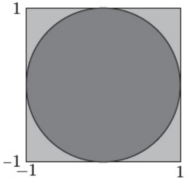
- The probability that a rain drop falls into the circle is $p \rightarrow$

$$P(\sqrt{x^2+y^2} < R) = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$$

Monte Carlo method

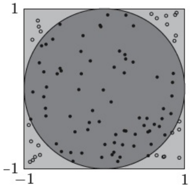
Computing π

- Assume *uniform rain* on the square $[-1, 1] \times [-1 : 1] \rightarrow x, y \approx U[-1 : 1]$



- The probability that a rain drop falls into the circle is $p \rightarrow$
$$P(\sqrt{x^2 + y^2} < R) = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$$

- Consider N independent rain drops and count the ones falling within the circle (rejection)



Monte Carlo method

Computing π

- $p = P(\sqrt{x^2 + y^2} < R) \approx \frac{N_{in}}{N}$ and $p = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$
- We can estimate $\bar{\pi} \approx 4\frac{N_{in}}{N}$
  - Assume $N = 100$
  - a result is $N_{in} = 77$
  - $\bar{\pi} = 4N_{in}/N = 3.08$ (a fairly bad estimate...)

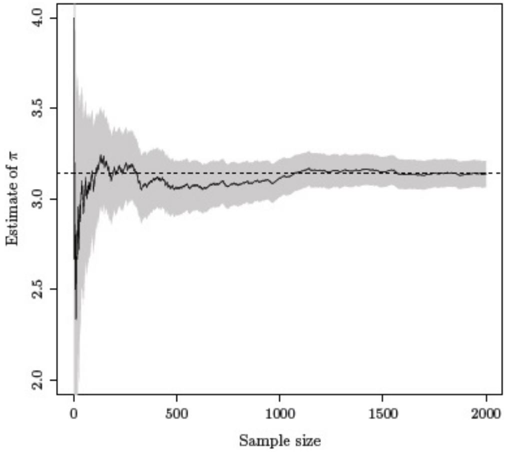
Monte Carlo method

Computing π

- $p = P(\sqrt{x^2 + y^2} < R) \approx \frac{N_{in}}{N}$ and $p = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$
- We can estimate $\bar{\pi} \approx 4\frac{N_{in}}{N}$
  - Assume $N = 100$
  - a result is $N_{in} = 77$
  - $\bar{\pi} = 4N_{in}/N = 3.08$ (a fairly bad estimate...)
  - The *Law of Large Numbers* guarantees that this estimate converges to π as $N \to \infty$

Monte Carlo method

Computing π - Convergence of the estimate



The Central Limit Theorem gives an estimate for the variance — and therefore of the error in the estimate

Monte Carlo method

Comments

- MC is simple, non-intrusive, parallel, and provides an error estimates

Monte Carlo method

Comments

- MC is simple, non-intrusive, parallel, and provides an error estimates
- The accuracy in MC increases as $1/\sqrt{N}$ independently on the number of dimensions d

Monte Carlo method

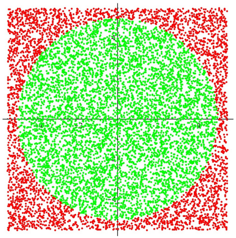
Comments

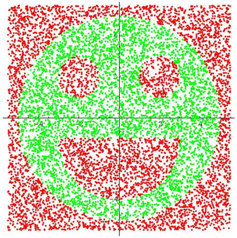
- MC is simple, non-intrusive, parallel, and provides an error estimates
- The accuracy in MC increases as $1/\sqrt{N}$ independently on the number of dimensions d
- It is easy to incorporate input variables covariances – if you know how to sample . . .

Monte Carlo method

Comments

- MC is simple, non-intrusive, parallel, and provides an error estimates
- The accuracy in MC increases as $1/\sqrt{N}$ independently on the number of dimensions d
- It is easy to incorporate input variables covariances – if you know how to sample . . .
- It is general





Beyond Monte Carlo

- History
  - MC estimation of π was suggested by Laplace in 1812
  - Monte Carlo was officially invented in 1940s by Von Neuman, Ulam and Metropolis (Manhattan Project)

Beyond Monte Carlo

- History
  - MC estimation of π was suggested by Laplace in 1812
  - Monte Carlo was officially invented in 1940s by Von Neuman, Ulam and Metropolis (Manhattan Project)
- Why do we want to do anything else?
  - Convergence speed

Beyond Monte Carlo

- History
  - MC estimation of π was suggested by Laplace in 1812
  - Monte Carlo was officially invented in 1940s by Von Neuman, Ulam and Metropolis (Manhattan Project)
- Why do we want to do anything else?
  - Convergence speed
- Need to cheat . . .
  - Importance sampling
  - Control variate
  - Latin Hypercube
  - Quasi Monte Carlo
  - . . .

Latin Hypecube Sampling, LHS

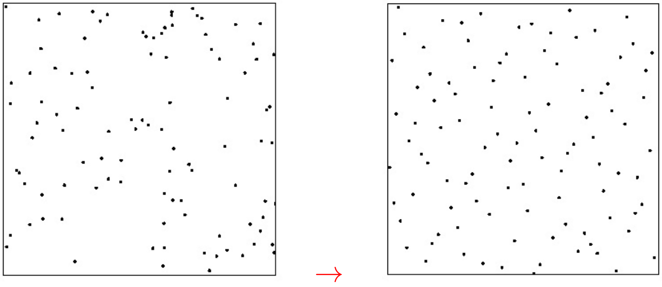
Also stratified MC or constrained MC

- Assume we have a $d-$dimensional input vector y
- In MC we pick N random d-dimensional vectors $y^i = (y_1^i, y_2^i, \ldots, y_d^i)$ for $i = 1, \ldots, N$

Latin Hypecube Sampling, LHS

Also *stratified MC* or *constrained MC*

- Assume we have a $d-$dimensional input vector y
- In MC we pick N random d-dimensional vectors $y^i = (y_1^i, y_2^i, \ldots, y_d^i)$ for $i = 1, \ldots, N$
- In LHS the realizations y^i are chosen in a different way...



LHS

Simple example

- Consider a 2D problem ($d = 2$) and assume we want to generate $N = 5$ LHS samples with y_1 a Gaussian r.v. and y_2 a Uniform r.v.

LHS

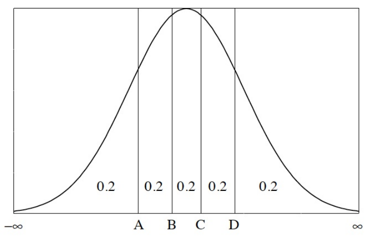
Simple example

- Consider a 2D problem ($d = 2$) and assume we want to generate $N = 5$ LHS samples with y_1 a Gaussian r.v. and y_2 a Uniform r.v.
- The first step is to build the *equi-probability* partitions

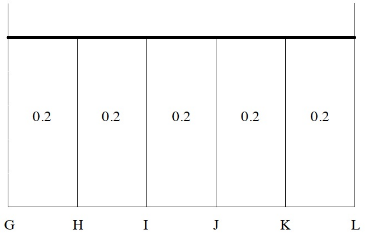
LHS

Simple example

- Consider a 2D problem ($d = 2$) and assume we want to generate $N = 5$ LHS samples with y_1 a Gaussian r.v. and y_2 a Uniform r.v.
- The first step is to build the *equi-probability* partitions



y_1



y_2

LHS

Simple example

- Sample *randomly* a value in each equi-probability partition
- We have now N values for y_1 and N values for y_2

LHS

Simple example

- Sample *randomly* a value in each equi-probability partition
- We have now N values for y_1 and N values for y_2
- The next step is the random pairing of the intervals: consider d random permutations of the first N integers and associate the result with each input variable interval.

Permutation #1: (3, 1, 5, 2, 4)
Permutation #2: (2, 4, 1, 3, 5)

LHS

Simple example

- Sample *randomly* a value in each equi-probability partition
- We have now N values for y_1 and N values for y_2
- The next step is the random pairing of the intervals: consider d random permutations of the first N integers and associate the result with each input variable interval.

Permutation #1: (3, 1, 5, 2, 4)
Permutation #2: (2, 4, 1, 3, 5)

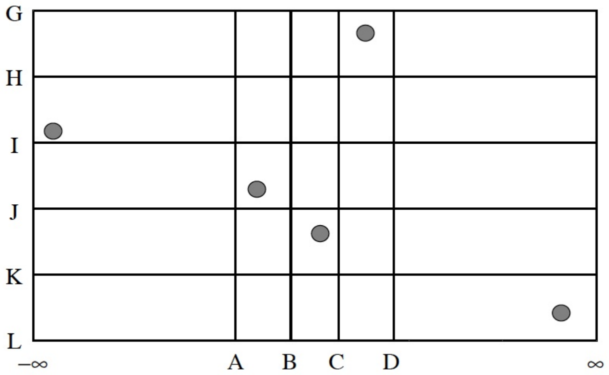
- The N input vectors y^i are then

Realization	y_1	y_2
1	3	2
2	1	4
3	5	1
4	2	3
5	4	5

LHS

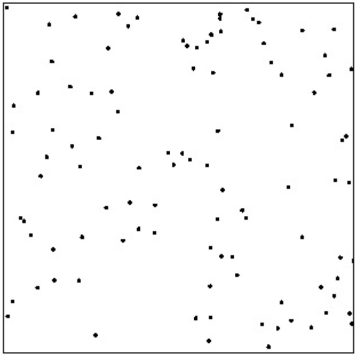
Simple example

These are the resulting realizations

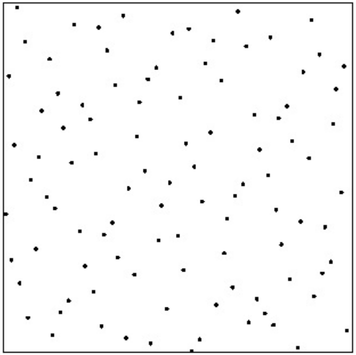


MC vs. LHS

Qualitative differences...suggestive of better coverage in LHS



Monte Carlo



LHS

LHS properties

- Advantages w.r.t. MC
  - Convergence is typically faster (lower variance of the estimate for equal N)
  - *Optimal* coverage of the marginals $\rightarrow$ equi-probability partitions

LHS properties

- Advantages w.r.t. MC
  - Convergence is typically faster (lower variance of the estimate for equal N)
  - *Optimal* coverage of the marginals $\rightarrow$ equi-probability partitions
- Disadvantages w.r.t. MC
  - LHS has a history
  - Need to run exactly N samples
  - It is possible (but not straightforward) to control the correlations between input variables by modifying the *pairing* step [Iman & Conover, 1992]

LHS properties

- Advantages w.r.t. MC
  - Convergence is typically faster (lower variance of the estimate for equal N)
  - *Optimal* coverage of the marginals $\rightarrow$ equi-probability partitions
- Disadvantages w.r.t. MC
  - LHS has a history
  - Need to run exactly N samples
  - It is possible (but not straightforward) to control the correlations between input variables by modifying the *pairing* step [Iman & Conover, 1992]

Remains the Method of choice for a number of engineering applications...

Concluding...

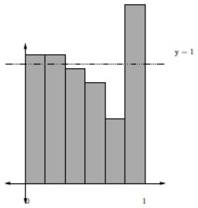
- In general MC methods are unaware of the problem (completely non-intrusive)...

Concluding...

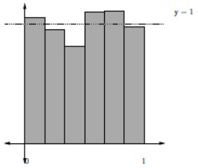
- In general MC methods are unaware of the problem (completely non-intrusive)...
- Consider the die-rolling example: probability of each outcome is 1/6th.
- what happens if we try MC?

Concluding...

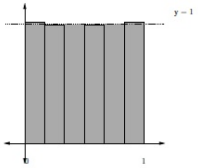
- In general MC methods are unaware of the problem (completely non-intrusive)...
- Consider the die-rolling example: probability of each outcome is 1/6th.
- what happens if we try MC?



$N = 100$



$N = 5000$



$N = 100000$

Sampling Methods

- Sample the random *input parameter vector* according to its probability distributions
- Perform a sequence of independent simulations
- Compute statistics of the quantity of interest

Sampling Methods

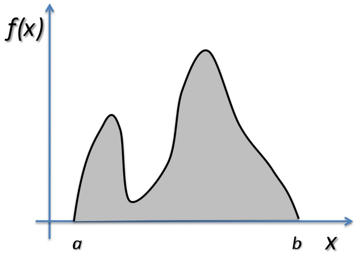
- Sample the random *input parameter vector* according to its probability distributions
- Perform a sequence of independent simulations
- Compute statistics of the quantity of interest

Now we will introduce an alternative way of computing the output statistics without random sampling!

Numerical Integration

The basic idea is to use *advanced* numerical integration techniques

- Recall numerical quadrature:



Problem: Compute the integral of $f(x)$ over the interval $[a : b]$.

Numerical integration

Express integrals as a *finite, weighted sum*

$$\int_a^b f(\xi)d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

- Examples: midpoint, trapezoidal, Simpson rules
- Remark: all use equispaced abscissas $\xi_i \in [a:b]$ (Newton-Cotes)

Numerical integration

Express integrals as a *finite, weighted sum*

$$\int_a^b f(\xi)d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

- Examples: midpoint, trapezoidal, Simpson rules
- Remark: all use equispaced abscissas $\xi_i \in [a:b]$ (Newton-Cotes)

We can do better: Gauss quadrature rules.

Observation: *We can always fit an N-1 degree polynomial to a set N points ($N = 2 \rightarrow$ line, $N - 3 \rightarrow$ parabola, etc.).*

Numerical integration

Express integrals as a *finite, weighted sum*

$$\int_a^b f(\xi)d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

- Examples: midpoint, trapezoidal, Simpson rules
- Remark: all use equispaced abscissas $\xi_i \in [a : b]$ (Newton-Cotes)

We can do better: Gauss quadrature rules.

Observation: *We can always fit an N-1 degree polynomial to a set N points ($N = 2 \rightarrow$ line, $N - 3 \rightarrow$ parabola, etc.). By carefully choosing the abscissas and weights (ξ_i, w_i), we can exactly evaluate the integral if $f(\xi)$ is $\leq (2N - 1)$ degree polynomial.*

Numerical integration

Express integrals as a *finite, weighted sum*

$$\int_a^b f(\xi) d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

- Examples: midpoint, trapezoidal, Simpson rules
- Remark: all use equispaced abscissas $\xi_i \in [a : b]$ (Newton-Cotes)

We can do better: Gauss quadrature rules.

Observation: *We can always fit an N-1 degree polynomial to a set N points ($N = 2 \rightarrow$ line, $N - 3 \rightarrow$ parabola, etc.).*
By carefully choosing the abscissas and weights (ξ_i, w_i), we can exactly evaluate the integral if $f(\xi)$ is $\leq (2N - 1)$ degree polynomial.
What are the abscissas ξ_i and the weights w_i?

Numerical quadrature

$$\int_a^b f(\xi) d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

Numerical quadrature

$$\int_a^b f(\xi)d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

The abscissas are the roots of orthogonal polynomials (Legendre)

$$\int_{-1}^{1} p_j(x)p_i(x)dx = C_i\delta_{ij}$$

Abscissas: impose $p_N(\xi) = 0 \rightarrow \xi_1, \ldots, \xi_N$

Numerical quadrature

$$\int_a^b f(\xi)d\xi \approx \sum_{i=1}^{N} w_i f(\xi_i)$$

The abscissas are the roots of orthogonal polynomials (Legendre)

$$\int_{-1}^{1} p_j(x)p_i(x)dx = C_i\delta_{ij}$$

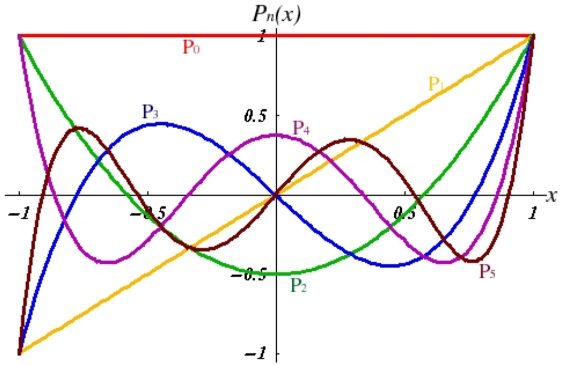
Abscissas: impose $p_N(\xi) = 0 \rightarrow \xi_1, \ldots, \xi_N$

The weights are the integrals of the Lagrange interpolating polynomials passing through the abscissas

$$w_i = \int_{-1}^{1} L_{i,N}(\xi)d\xi \quad \text{with} \quad L_{i,N}(\xi) = \prod_{\substack{k=1 \\ k\neq j}}^{N} \frac{\xi - \xi_k}{\xi_i - \xi_k}$$

Legendre-Gauss quadrature

Legendre Polynomials



$$(n+1)P_{n+1}(\xi) = (2n+1)\xi P_n(\xi) - nP_{n-1}(\xi)$$ Three-term recurrence

$$\int_{-1}^{1} P_j(x)P_i(x)dx = \frac{2}{2n+1}\delta_{ij}$$ Orthogonality

Legendre-Gauss quadrature

Both the abscissas and the weights are tabulated and can be computed in several ways

For example:

```
function I = gauss(f,n)                  % (n+1)-pt Gauss quadrature
beta = .5/sqrt(1-(2*(1:n))^(-2));        % 3-term recurrence coeffs
T = diag(beta,1) + diag(beta,-1);        % Jacobi matrix
[V,D] = eig(T);                          % eigenvalue decomposition
x = diag(D); [x,i] = sort(x);            % nodes (= Legendre points)
w = 2*V(1,i)^2;                          % weights
I = w*feval(f,x);                        % the integral
```

The command `gauss(cos,6)` yields 1.68294196961579 which is correct to double precision [Trefethen, 2008]

Advanced Concepts

Beyond Uniform rvs

Gaussian rvs

What do we do if the input variables are not distributed as uniform r.v.?

Beyond Uniform rvs

Gaussian rvs

What do we do if the input variables are not distributed as uniform r.v.?
As you probably know, numerical quadrature is more than just Legendre-Gauss quadrature!

Beyond Uniform rvs

Gaussian rvs

What do we do if the input variables are not distributed as uniform r.v.?
As you probably know, numerical quadrature is more than just Legendre-Gauss quadrature!

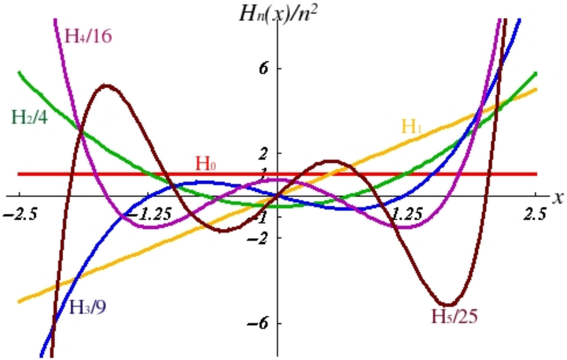
Consider y distributed as a $N(0,1)$, we can build orthogonal polynomials w.r.t. to a Gaussian measure as:

$$\int_{-\infty}^{\infty} p_j(x)p_i(x)e^{-x^2}dx = C_i\delta_{ij}$$

Hermite polynomials for normal r.v. play the same role as Legendre polynomials for uniform r.v.s!

Hermite-Gauss quadrature

Hermite Polynomials



$$H_{n+1}(\xi) = 2\xi H_n(\xi) - 2nH_{n-1}(\xi)$$ Three-term recurrence

$$\int_{-\infty}^{\infty} H_j(x)H_i(x)e^{-x^2}dx = 2^i i!\sqrt{\pi}\delta_{ij}$$ Orthogonality

Stochastic Collocation

Summary of the quadrature rules

We can use Legendre or Hermite polynomials, can we do even more?

Stochastic Collocation

Summary of the quadrature rules

We can use Legendre or Hermite polynomials, can we do even more?

Distribution	pdf	Polynomials	Weights	Support
Uniform	$1/2$	Legendre	1	$[-1:1]$
Gaussian	$(1/\sqrt{2\pi})e^{(-x^2/2)}$	Hermite	$e^{(-x^2/2)}$	$[-\infty:\infty]$
Exponential	e^{-x}	Laguerre	e^{-x}	$[0:\infty]$
Beta	$\frac{(1-x)^{\alpha}(1+x)^{\beta}}{B(\alpha,\beta)}$	Jacobi	$(1-x)^{\alpha}(1+x)^{\beta}$	$[-1:1]$

Table: Some polynomials in the Askey family

Stochastic Collocation

Summary of the quadrature rules

We can use Legendre or Hermite polynomials, can we do even more?

Distribution	pdf	Polynomials	Weights	Support
Uniform	$1/2$	Legendre	1	$[-1:1]$
Gaussian	$(1/\sqrt{2\pi})e^{(-x^2/2)}$	Hermite	$e^{(-x^2/2)}$	$[-\infty:\infty]$
Exponential	e^{-x}	Laguerre	e^{-x}	$[0:\infty]$
Beta	$\frac{(1-x)^{\alpha}(1+x)^{\beta}}{B(\alpha,\beta)}$	Jacobi	$(1-x)^{\alpha}(1+x)^{\beta}$	$[-1:1]$

Table: Some polynomials in the Askey family

What if the random variables are not distributed according to any of the above?

Stochastic Collocation

Summary of the quadrature rules

We can use Legendre or Hermite polynomials, can we do even more?

Distribution	pdf	Polynomials	Weights	Support
Uniform	$1/2$	Legendre	1	$[-1:1]$
Gaussian	$(1/\sqrt{2\pi})e^{(-x^2/2)}$	Hermite	$e^{(-x^2/2)}$	$[-\infty:\infty]$
Exponential	e^{-x}	Laguerre	e^{-x}	$[0:\infty]$
Beta	$\frac{(1-x)^{\alpha}(1+x)^{\beta}}{B(\alpha,\beta)}$	Jacobi	$(1-x)^{\alpha}(1+x)^{\beta}$	$[-1:1]$

Table: Some polynomials in the Askey family

What if the random variables are not distributed according to any of the above?

1. Szego (1939). Orthogonal Polynomials - American Mathematical Society.
2. Schoutens (2000). Stochastic Processes and Orthogonal Polynomial - Springer.
3. Gram-Schmidt Procedure

Nested rules

The choice of abscissas

The Gauss quadrature rules introduce different abscissas for each order N considered. They are not nested, no reuse of computed solutions for lower-order quadrature

Nested rules

The choice of abscissas

The Gauss quadrature rules introduce different abscissas for each order N considered. They are not nested, no reuse of computed solutions for lower-order quadrature
Two extensions are possible

- Gauss-Kronrod rules
- Clenshaw-Curtis rules: express the integrand using Chebyshev polynomials (lower polynomial exactness)

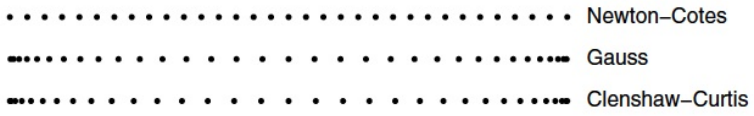


Figure: 32 abscissas in $[-1:1]$

Clenshaw-Curtis vs. Gauss

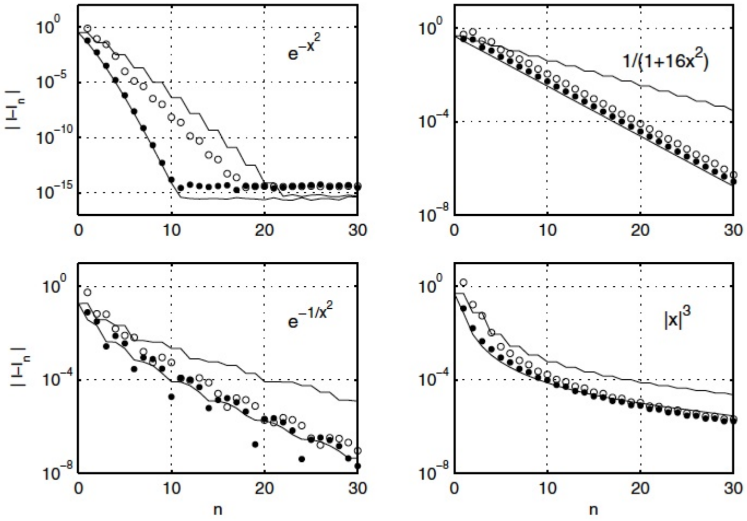


Figure: Black: Gauss, White: CC [Trefethen, 2008]

Why Nested rules?

Assume you have a budget of $N = 9$ computations.

- With 9 Gauss abscissas (Legendre), we can obtain an estimate of the statistics of the solution which would be exact if the solution is a polynomial of degree $\leq 2N - 1 = 17$

Why Nested rules?

Assume you have a budget of $N = 9$ computations.

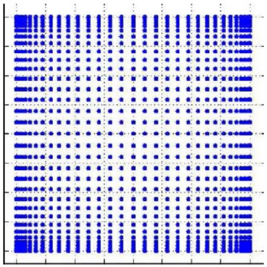
- With 9 Gauss abscissas (Legendre), we can obtain an estimate of the statistics of the solution which would be exact if the solution is a polynomial of degree $\leq 2N - 1 = 17$
- With Clenshaw-Curtis we can obtain again an estimate (only exact for polynomials of degree $\leq N = 9$).
  On the other hand, with the same computations ($N = 9$ abscissas) we can also estiamte the solution statistics corresponding to $N = 5$ and $N = 3$ $\rightarrow$ error estimate.

Multi-dimensional rules

Tensor Product

The extension of the previous 1D rules (Gauss or CC) is straightforward

- The abscissas are tensor products of the quadrature points in 1D
- The weights are the products of the 1D weights

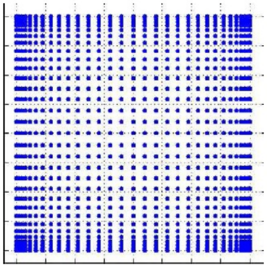


Multi-dimensional rules

Tensor Product

The extension of the previous 1D rules (Gauss or CC) is straightforward

- The abscissas are tensor products of the quadrature points in 1D
- The weights are the products of the 1D weights



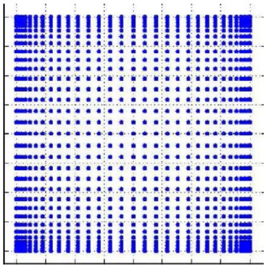
- The number of function evaluations increases as N^d $\rightarrow$ curse of dimensionality

Multi-dimensional rules

Tensor Product

The extension of the previous 1D rules (Gauss or CC) is straightforward

- The abscissas are tensor products of the quadrature points in 1D
- The weights are the products of the 1D weights



- The number of function evaluations increases as N^d $\rightarrow$ curse of dimensionality
- Remark: This is valid ONLY if the uncertain variables are independent (because the joint PDF becomes the product of the marginals)!

Extension of the Stochastic Collocation Methodology

- Stochastic Collocation is a very simple ad powerful alternative to MC sampling
- Several limitations remain:
  - High-Dimensionality
  - Non-Smooth Responses
  - General Correlated/Dependent Inputs

Extension of the Stochastic Collocation Methodology

- Stochastic Collocation is a very simple ad powerful alternative to MC sampling
- Several limitations remain:
  - High-Dimensionality
  - Non-Smooth Responses
  - General Correlated/Dependent Inputs
- Various extensions have attempted to address these issues:
  - Multi-dimensional constructions: Sparse Grids
  - Global vs. Local Basis: Multi-element methods and Simplex Stochastic Collocation
  - Adaptive Quadrature
  - Different Choice of Basis (non-polynomials): Wavelets, Pade’

Multi-dimensional Extensions

Sparse Grids - Smolyak Grids

Smolyak idea is to sparsify the construction of quadrature grids

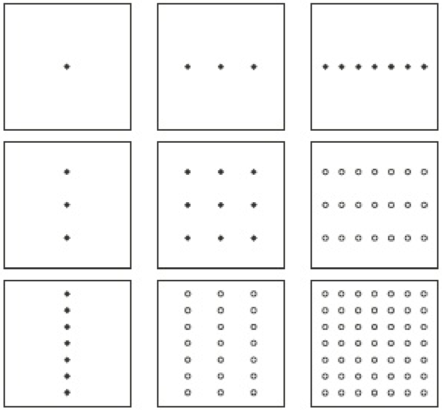


Figure: Sequence of grids used in 2D by a nested rule

Multi-dimensional Extensions

Sparse Grids - Smolyak Grids

The *nominal* accuracy can be preserved with much less points

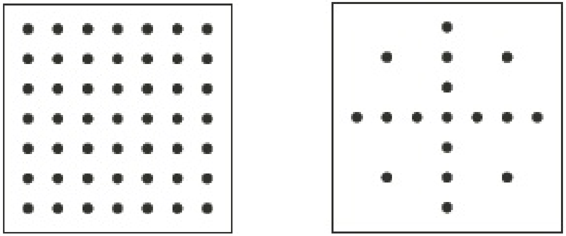


Figure: From Tensor grid to Sparse grid in 2D

Multi-dimensional Extensions

Sparse Grids - Smolyak Grids

The *nominal* accuracy can be preserved with much less points

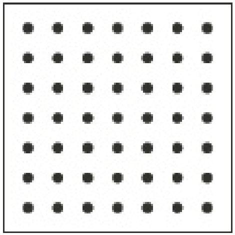
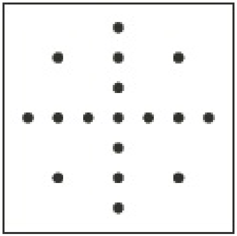



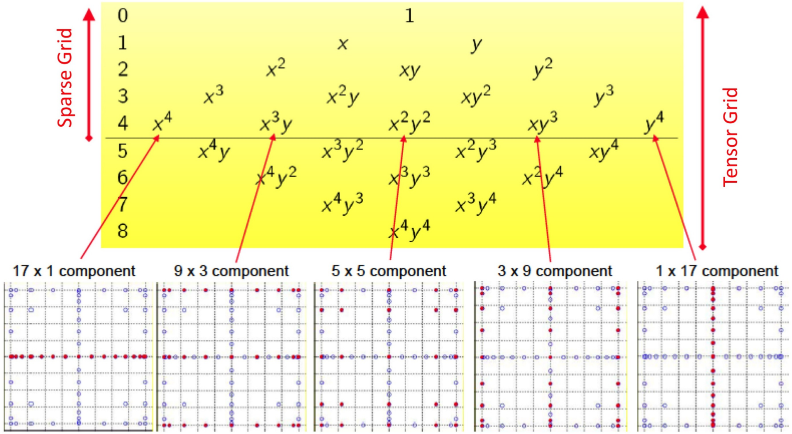
Figure: From Tensor grid to Sparse grid in 2D

The method is based on a linear combination of tensor products to build the actual sparse grid

Sparse Grids

Rationale

The key is to *reinterpret* the concept of "polynomial exactness"



Modified from Eldred 2009

Sparse Grids

Stochastic Collocation

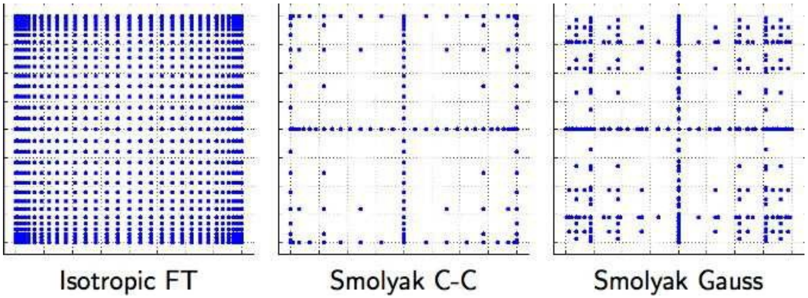


Table: Abscissas for $N = 5$ in each dimension

	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
Tensor Gauss	25	125	625	3125	15625	78125
Smolyak Gauss	17	31	49	71	97	127
Smolyak CC	13	25	41	61	85	113

Sparse Grids
Summary

- A fundamental advance in the development of stochastic collocation approaches in multiD
- Becoming more and more popular

Sparse Grids

Summary

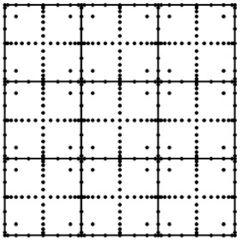
- A fundamental advance in the development of stochastic collocation approaches in multiD
- Becoming more and more popular

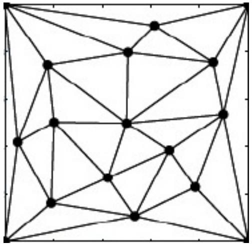
- Not perfect
  - Not straightforward to construct (implementation errors)
  - Does not *solve* the curse of dimensionality, although it is better than tensor grids
  - Not very flexible. Increasing the accuracy requires a large increase in number of solutions...

From Global to Local Basis

Multi-Element Methods

- The classical construction of the stochastic collocation method relies on polynomial basis defined over the entire domain spanned by the input uncertainties
- In many cases it is useful to compute the integrals over subdomains
  - Capture local features (including discontinuities)
  - Allow more control on the number of simulations to perform
  - ...


Multi-Element SC


Simplex SC

Basis Selection

Adaptivity & Anisotropy

- In multi-dimensional problem it is *unlikely* that all the input uncertainty have the same importance with respect to the quantity of interest
- How can we selectively increase the accuracy of the integration?

Basis Selection

Adaptivity & Anisotropy

- In multi-dimensional problem it is *unlikely* that all the input uncertainty have the same importance with respect to the quantity of interest
- How can we selectively increase the accuracy of the integration?
- Define a sensor based on
  - sensitivity
  - variance decomposition
  - error estimate

Basis Selection

Adaptivity & Anisotropy

- In multi-dimensional problem it is *unlikely* that all the input uncertainty have the same importance with respect to the quantity of interest
- How can we selectively increase the accuracy of the integration?
- Define a sensor based on
  - sensitivity
  - variance decomposition
  - error estimate
- Tailor the interpolation basis
  - Increase the polynomial order selectively (anisotropy)
  - Choose special basis (enrichment)
  - Increase the resolution locally (subdomain decomposition)

Discontinuous Surface Responses - Approaches

This is not a new problem....

- Multi-element approaches (Wan & Karniadakis, . . .)
- Wavelet-based polynomial chaos (LeMaitre et al.)
- Basis-enrichment (Ghosh & Ghanem, . . .)
- Polynomial Annihilation (Jakeman & Xiu)
- Simplex Element Collocation (Witteveen & Iaccarino)
- Kriging (Jouhaout & Libediu, . . .)
- . . .

Discontinuous Surface Responses - Approaches

This is not a new problem....

- Multi-element approaches (Wan & Karniadakis, . . .)
- Wavelet-based polynomial chaos (LeMaitre et al.)
- Basis-enrichment (Ghosh & Ghanem, . . .)
- Polynomial Annihilation (Jakeman & Xiu)
- Simplex Element Collocation (Witteveen & Iaccarino)
- Kriging (Jouhaout & Libediu, . . .)
- . . .

Why do we need another method?

Discontinuous Surface Responses - Approaches

This is not a new problem....

- Multi-element approaches (Wan & Karniadakis, . . .)
- Wavelet-based polynomial chaos (LeMaitre et al.)
- Basis-enrichment (Ghosh & Ghanem, . . .)
- Polynomial Annihilation (Jakeman & Xiu)
- Simplex Element Collocation (Witteveen & Iaccarino)
- Kriging (Jouhaout & Libediu, . . .)
- . . .

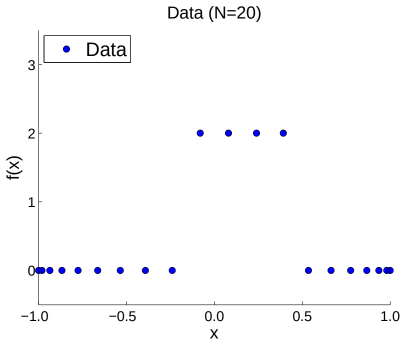
Why do we need another method?

- Prefer a global approach
- No prior knowledge of the location of discontinuity
- Avoid adaptivity
- Reuse/extend stochastic collocation framework

Padé-Legendre (PL) Method

- Consider $f(x) = \text{sign}(x + 0.2) - \text{sign}(x - 0.5)$.

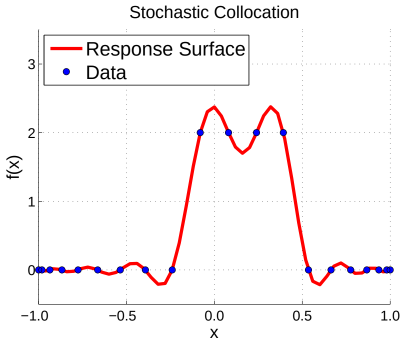
Data: Number of data points: $N = 20$



Padé-Legendre (PL) Method

- Consider $f(x) = \text{sign}(x + 0.2) - \text{sign}(x - 0.5)$.

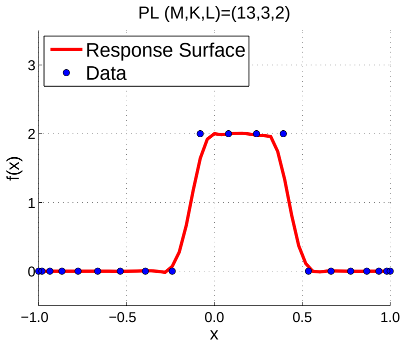
Stochastic Collocation solution



Padé-Legendre (PL) Method

- Consider $f(x) = \text{sign}(x + 0.2) - \text{sign}(x - 0.5)$.

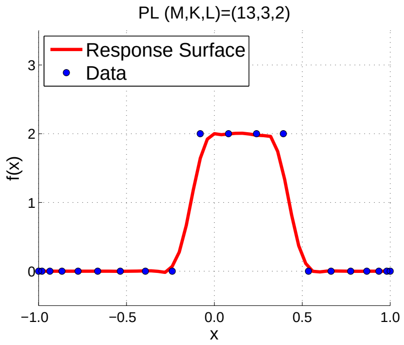
PL with "tuned" parameters (e.g. polynomial orders)



Padé-Legendre (PL) Method

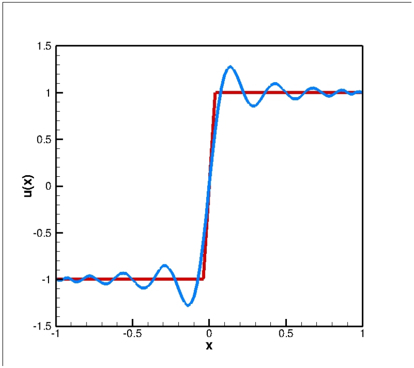
- Consider $f(x) = \text{sign}(x + 0.2) - \text{sign}(x - 0.5)$.

PL with "tuned" parameters (e.g. polynomial orders)



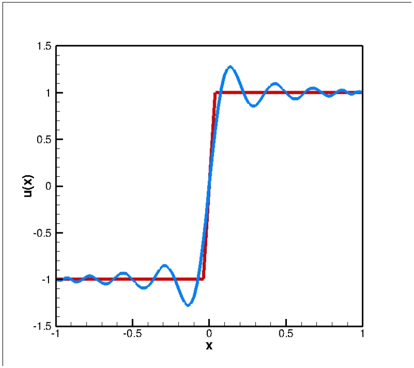
How to construct the PL approximant?
How to choose the tuning parameters?

Padé-Legendre (PL) Method

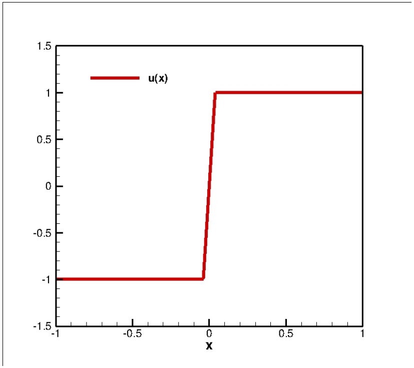


Gibbs phenomena. Polynomial interpolation

Padé-Legendre (PL) Method

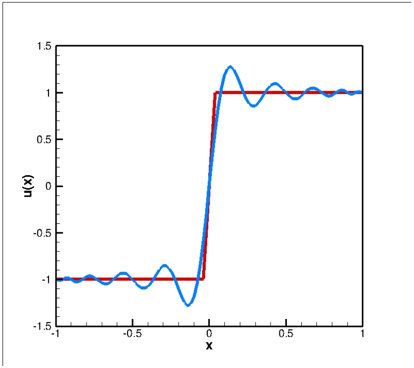


Gibbs phenomena. Polynomial interpolation

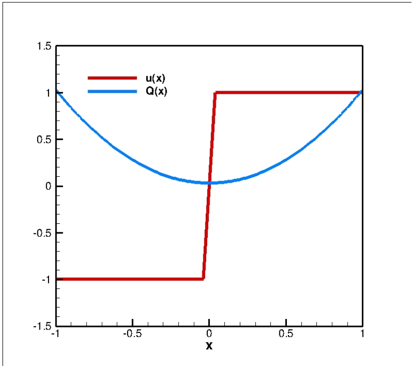


Data

Padé-Legendre (PL) Method

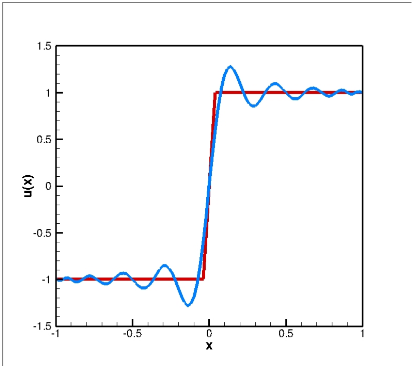


Gibbs phenomena. Polynomial interpolation

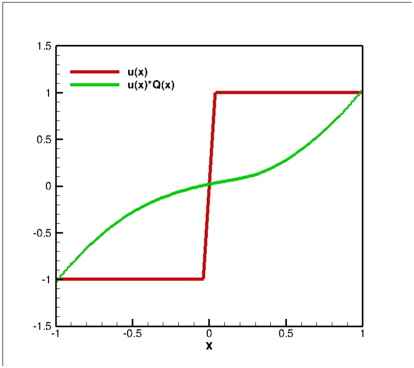


Auxiliary function $Q(x)$

Padé-Legendre (PL) Method



Gibbs phenomena. Polynomial interpolation



Preconditioned function $Q(x)u(x)$

PL Formulation (1-D)

- Given data $u(x_k)$, $k = 0, 1, \ldots, N$
- Find the approximation $R(x) \approx u(x)$ in the form of

$$R(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{j=0}^{M} \hat{p}_j \Psi_j(x)}{\sum_{j=0}^{L} \hat{q}_j \Psi_j(x)}, \tag{A1}$$

PL Formulation (1-D)

- Given data $u(x_k)$, $k = 0, 1, \ldots, N$
- Find the approximation $R(x) \approx u(x)$ in the form of

$$R(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{j=0}^{M} \hat{p}_j \Psi_j(x)}{\sum_{j=0}^{L} \hat{q}_j \Psi_j(x)}, \qquad \text{(A1)}$$

such that

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N, \qquad \text{(A2)}$$

where where Ψ_j's are the Legendre polynomial basis and $\langle \cdot, \cdot \rangle_N$ is the discrete inner product.

* J. Hesthaven, et al, Padé–Legendre interpolants for Gibbs reconstruction, J. Sci. Comput. 28 (2006) 337-359.

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate Q by using $\phi = \Psi \in \mathbb{P}_N \setminus \mathbb{P}_M$.

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate Q by using $\phi = \Psi \in \mathbb{P}_N \setminus \mathbb{P}_M$.
  But $P \in \mathbb{P}_M$, thus by orthogonality:

$$\langle Qu, \Psi_n \rangle_N = 0, \qquad n = M + 1, \ldots, N. \tag{1}$$

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate Q by using $\phi = \Psi \in \mathbb{P}_N \setminus \mathbb{P}_M$.
  But $P \in \mathbb{P}_M$, thus by orthogonality:

$$\langle Qu, \Psi_n \rangle_N = 0, \qquad n = M+1, \dots, N. \tag{1}$$

  Solve the following linear system for coefficients of Q:

$$\begin{bmatrix} \langle u\Psi_0, \Psi_{M+1} \rangle_N & \cdots & \langle u\Psi_L, \Psi_{M+1} \rangle_N \\ \vdots & \ddots & \vdots \\ \langle u\Psi_0, \Psi_{M+L} \rangle_N & \cdots & \langle u\Psi_L, \Psi_{M+L} \rangle_N \end{bmatrix} \begin{bmatrix} \hat{q}_0 \\ \vdots \\ \hat{q}_L \end{bmatrix} = \underline{0}.$$

  Matrix size: $L \times (L+1)$. Solve for nonzero Q.

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

$$\langle P, \Psi_n \rangle_N = \langle Qu, \Psi_n \rangle_N, \qquad n = 0, 1, \ldots, M \tag{2}$$

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

  $$\langle P, \Psi_n \rangle_N = \langle Qu, \Psi_n \rangle_N, \qquad n = 0, 1, \ldots, M \tag{2}$$

  The right hand side is known (Q has already been calculated).

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

$$\langle P, \Psi_n \rangle_N = \langle Qu, \Psi_n \rangle_N, \qquad n = 0, 1, \ldots, M \tag{2}$$

  The right hand side is known (Q has already been calculated).
  The left hand side is $\hat{p}_n \langle \Psi_n, \Psi_n \rangle_N$.

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

$$\langle P, \Psi_n \rangle_N = \langle Qu, \Psi_n \rangle_N, \qquad n = 0, 1, \ldots, M \tag{2}$$

The right hand side is known (Q has already been calculated).
The left hand side is $\hat{p}_n \langle \Psi_n, \Psi_n \rangle_N$.

$$\hat{p}_n = \frac{\langle P, \Psi_n \rangle_N}{\langle \Psi_n, \Psi_n \rangle_N} = \frac{\langle Qu, \Psi_n \rangle_N}{\langle \Psi_n, \Psi_n \rangle_N}, \qquad n = 0, 1, \ldots, M \tag{3}$$

PL Construction (1-D)

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_N \tag{A2}$$

- Calculate P by using $\phi = \Psi \in \mathbb{P}_M$.

$$\langle P, \Psi_n \rangle_N = \langle Qu, \Psi_n \rangle_N, \qquad n = 0, 1, \ldots, M \tag{2}$$

  The right hand side is known (Q has already been calculated).
  The left hand side is $\hat{p}_n \langle \Psi_n, \Psi_n \rangle_N$.

$$\hat{p}_n = \frac{\langle P, \Psi_n \rangle_N}{\langle \Psi_n, \Psi_n \rangle_N} = \frac{\langle Qu, \Psi_n \rangle_N}{\langle \Psi_n, \Psi_n \rangle_N}, \qquad n = 0, 1, \ldots, M \tag{3}$$

- $R = P/Q \approx u$.

1-D to n-D

One-dimensional PL

- There are $N + 1$ equations, one for each Ψ_n.
- We split the equations into M and L ($M + L = N + 1$).
- The last L equations are used to calculate Q.
- The first M equations are then used to calculate P.

Multi-dimensional PL

- Let d be the dimension.
- There are $c(N, d) = \frac{(N+d)!}{N!d!}$ equations.
- There are $c(L, d) = \frac{(L+d)!}{L!d!}$ coefficients in Q
- And $c(M, d) = \frac{(M+d)!}{M!d!}$ coefficients in P.
- It is *impossible* to split the equations into two groups to match the numbers of unknowns.

1-D to *n*-D

One-dimensional PL

- There are $N + 1$ equations, one for each Ψ_n.
- We split the equations into M and L ($M + L = N + 1$).
- The last L equations are used to calculate Q.
- The first M equations are then used to calculate P.

Multi-dimensional PL

- Let d be the dimension.
- There are $c(N, d) = \frac{(N+d)!}{N!d!}$ equations.
- There are $c(L, d) = \frac{(L+d)!}{L!d!}$ coefficients in Q
- And $c(M, d) = \frac{(M+d)!}{M!d!}$ coefficients in P.
- It is *impossible* to split the equations into two groups to match the numbers of unknowns.

PL Formulation (n-D)

- Given data $u(x_k, y_l)$, $k, l = 0, 1, \ldots, N$
- Find the approximation $R(x, y) \approx u(x, y)$ in the form of

$$R(x) = \frac{P(x,y)}{Q(x,y)} = \frac{\sum_{j=0}^{c(M)-1} \hat{p}_j \Psi_j(x,y)}{\sum_{j=0}^{c(L)-1} \hat{q}_j \Psi_j(x,y)}, \tag{4}$$

PL Formulation (n-D)

- Given data $u(x_k, y_l)$, $k, l = 0, 1, \dots, N$
- Find the approximation $R(x, y) \approx u(x, y)$ in the form of

$$R(x) = \frac{P(x,y)}{Q(x,y)} = \frac{\sum_{j=0}^{c(M)-1} \hat{p}_j \Psi_j(x,y)}{\sum_{j=0}^{c(L)-1} \hat{q}_j \Psi_j(x,y)}, \tag{4}$$

such that

$$\langle P - Qu, \phi \rangle_N = 0, \qquad \forall \phi \in \mathbb{P}_M, \tag{5}$$

and $\langle P - Qu, \phi \rangle_N$ is minimized for $\phi \in \mathbb{P}_{M+K}$.

PL Construction (*n*-D)

- Similar to 1-D, choose the Legendre basis: $\phi = \Psi$.
- Calculate Q *approximately* by using $\phi = \Psi \in \mathbb{P}_{M+K} \setminus \mathbb{P}_M$
- Matrix size: $L \times (K+1)$. Solve for nonzero Q using a least-square minimization

PL Construction (n-D)

- Similar to 1-D, choose the Legendre basis: $\phi = \Psi$.
- Calculate Q *approximately* by using $\phi = \Psi \in \mathbb{P}_{M+K} \setminus \mathbb{P}_M$
- Matrix size: $L \times (K+1)$. Solve for nonzero Q using a least-square minimization
- Calculate P *exactly* by using $\phi = \Psi \in \mathbb{P}_M$.

PL Construction (*n*-D)

- Similar to 1-D, choose the Legendre basis: $\phi = \Psi$.
- Calculate Q *approximately* by using $\phi = \Psi \in \mathbb{P}_{M+K} \setminus \mathbb{P}_M$
- Matrix size: $L \times (K+1)$. Solve for nonzero Q using a least-square minimization
- Calculate P *exactly* by using $\phi = \Psi \in \mathbb{P}_M$.

We have to specify L, M and K (N is usually given).

Automatic Parameter Selection

- Every triplet (L, M, K) gives a different response surface.
- We designed a strategy (called APS) to choose the “best” response surfaces among all the possible choices of (L, M, K)

Question: What do we mean by “best?”

Answer: According to 2 error measures.

Two Error Measures

- L_2-error (measure of accuracy w.r.t. data)

$$e_{L_2} = \frac{\|\tilde{u} - u\|_{L_2}}{\|u\|_{L_2}} = \left(\frac{\sum_{j=1}^{N_q} w_j (u(x_j) - \tilde{u}(x_j))^2}{\sum_{j=1}^{N_q} w_j u^2(x_j)} \right)^{\frac{1}{2}},$$

Two Error Measures

- L_2-error (measure of accuracy w.r.t. data)

$$e_{L_2} = \frac{\|\tilde{u} - u\|_{L_2}}{\|u\|_{L_2}} = \left(\frac{\sum_{j=1}^{N_q} w_j (u(x_j) - \tilde{u}(x_j))^2}{\sum_{j=1}^{N_q} w_j u^2(x_j)} \right)^{\frac{1}{2}},$$

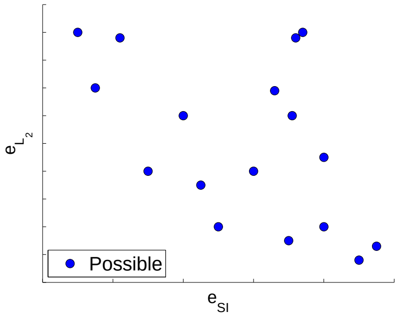
- Smoothness Indicator (measure of lack of spurious oscillations between data points)

$$e_{SI} = \frac{|\mathrm{SI}(\tilde{u}, G_F) - \mathrm{SI}(u, G_D)|}{\mathrm{SI}(u, G_D)},$$

where $SI(\cdot)$ is Total Variation, G_D is a grid consisting of the available data, and G_F is an additional highly refined grid.

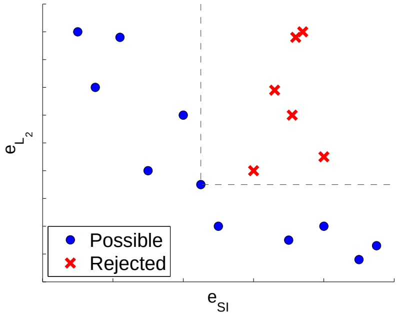
Pareto Front

Plot all response surfaces according to e_{L_2} and e_{SI}



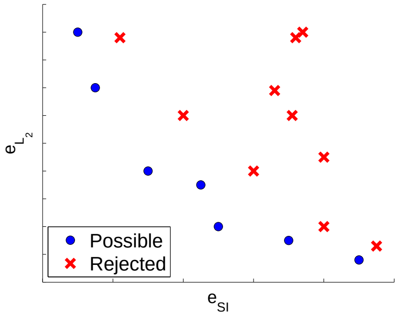
Pareto Front

Reject all the ones that cannot be best



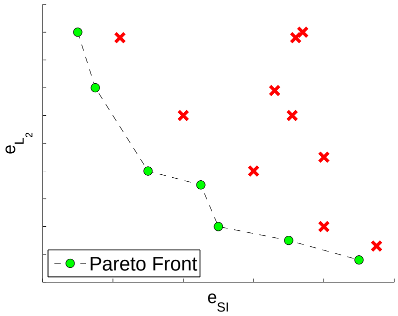
Pareto Front

Keep rejecting until we can no longer reject anymore



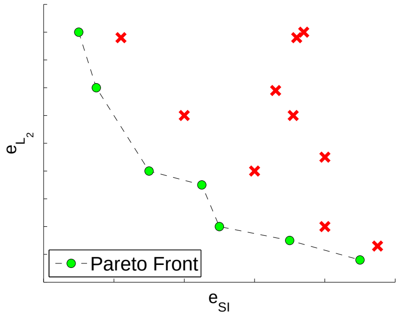
Pareto Front

The remaining PL surfaces constitute the Pareto front



Pareto Front

The remaining PL surfaces constitute the Pareto front



- Bottom-right: most data-accurate, but least smooth
- Top-left: most smooth, but least data-accurate

APS

- Any response surface in the Pareto front is logically acceptable.

A good trade-off between smoothness and data-accuracy depends on applications

- Data-accuracy is always good, but ...
- How accurate is the given data?
- Do we want to extract gradient information?
- Do we want to calculate extrema?

APS

- Any response surface in the Pareto front is logically acceptable.

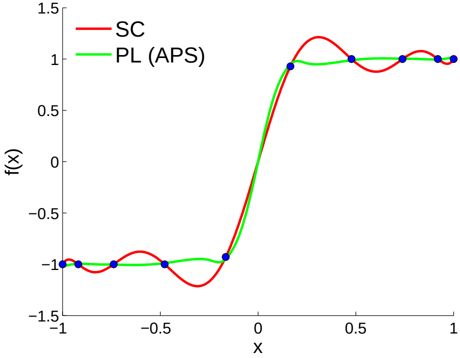
A good trade-off between smoothness and data-accuracy depends on applications

- Data-accuracy is always good, but ...
- How accurate is the given data?
- Do we want to extract gradient information?
- Do we want to calculate extrema?

Example

Underlying function: $f(x) = \tanh(10x), x \in [-1, 1]$
APS strategy: Most data-accurate.
Stochastic Collocation (SC) vs Padé-Legendre (PL) method

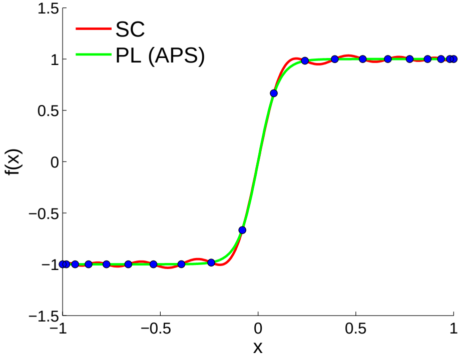
Number of data points: $N = 10$



Example

Underlying function: $f(x) = \tanh(10x), x \in [-1, 1]$
APS strategy: Most data-accurate.
Stochastic Collocation (SC) vs Padé-Legendre (PL) method

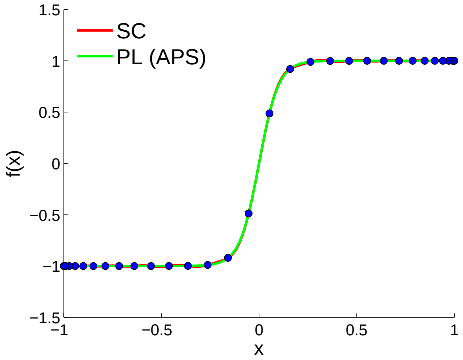
Number of data points: $N = 20$



Example

Underlying function: $f(x) = \tanh(10x), x \in [-1, 1]$
APS strategy: Most data-accurate.
Stochastic Collocation (SC) vs Padé-Legendre (PL) method

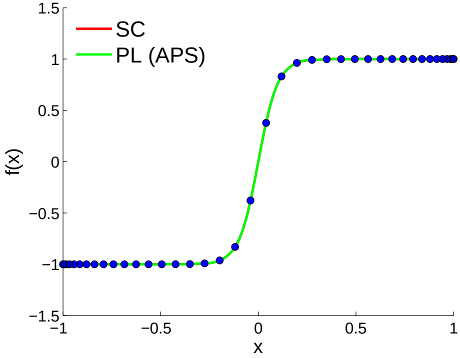
Number of data points: $N = 30$



Example

Underlying function: $f(x) = \tanh(10x), x \in [-1, 1]$
APS strategy: Most data-accurate.
Stochastic Collocation (SC) vs Padé-Legendre (PL) method

Number of data points: $N = 40$



Convergence to SC for Smooth Underlying Functions

Consider $f(x) = \tanh(x/\delta)$. Vary the number of data points, N. Observe L of the most data-accurate response surfaces.

	$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$	
N	$e_{SI}[SC]$	L	$e_{SI}[SC]$	L	$e_{SI}[SC]$	L
8	9.744e-1	2	2.852e-1	2	1.045e-1	2
10	5.882e-1	4	1.474e-1	4	2.627e-2	4
12	3.281e-1	4	6.224e-2	4	7.192e-3	4
14	2.141e-1	6	2.508e-2	6	2.414e-3	0
16	1.311e-1	6	8.718e-3	6	6.083e-4	0
18	7.265e-2	8	3.359e-3	0	2.535e-4	0
20	4.124e-2	8	1.069e-3	0	8.143e-5	0
22	2.352e-2	8	3.840e-4	0	2.603e-5	0
24	1.257e-2	9	1.656e-4	0	8.291e-6	0
26	6.967e-3	0	6.731e-5	0	2.596e-6	0
28	3.665e-3	0	2.839e-5	0	1.143e-6	0

Convergence to SC for Smooth Underlying Functions

Consider $f(x) = \tanh(x/\delta)$. Vary the number of data points, N. Observe L of the most data-accurate response surfaces.

	$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$	
N	$e_{SI}[SC]$	L	$e_{SI}[SC]$	L	$e_{SI}[SC]$	L
8	9.744e-1	2	2.852e-1	2	1.045e-1	2
10	5.882e-1	4	1.474e-1	4	2.627e-2	4
12	3.281e-1	4	6.224e-2	4	7.192e-3	4
14	2.141e-1	6	2.508e-2	6	2.414e-3	0
16	1.311e-1	6	8.718e-3	6	6.083e-4	0
18	7.265e-2	8	3.359e-3	0	2.535e-4	0
20	4.124e-2	8	1.069e-3	0	8.143e-5	0
22	2.352e-2	8	3.840e-4	0	2.603e-5	0
24	1.257e-2	9	1.656e-4	0	8.291e-6	0
26	6.967e-3	0	6.731e-5	0	2.596e-6	0
28	3.665e-3	0	2.839e-5	0	1.143e-6	0

CEMRACS Summer School

July 2012

CIRM, Marseille, France