AN INTRODUCTION TO ALGORITHMS FOR
UNCERTAINTY QUANTIFICATION FOR
PDES WITH RANDOM INPUTS

Max Gunzburger
Department of Scientific Computing, Florida State University
gunzburg@fsu.edu

Basque Center for Applied Mathematics and University of Valladolid
Bilbao, Spain, July 2016
I - UNCERTAINTY QUANTIFICATION IN THE PDE SETTING
• Physical, biological, social, economic, financial, etc. systems always involve uncertainties which should be accounted for in the mathematical models describing these systems

• Uncertainties are often classified into two categories

• Epistemic uncertainties are due to incomplete knowledge of a system
  – in principle if not in practice, can be reduced by additional measurements, improvements in measuring devices, and perhaps by other means
  – although the data are predictable, they may be too difficult or too costly to obtain
  – an example are media properties in oil reservoirs or aquifers

• Aleatoric uncertainties are intrinsic in a system so that they cannot be reduced through additional measurements, improvements in measuring devices, etc.
  – examples include wind gusts and rainfall amounts
• In addition to these two categories
  – uncertainty can be introduced into a deterministic system in cases in which not all scales in the data and/or solution can or should be resolved either because it is too difficult, or perhaps impossible, or too costly to do so in a computational simulation (as is the case for, e.g., turbulent flows) or
  they may not be of interest to the goals of the simulation (as is often the case for, e.g., surface roughness or hourly stock prices)

• From the point of view of numerical methods, there is no appreciable differences between the types and sources of uncertainty so that in these lectures we will not be concerned about how uncertainties in system specifications arise

• One should also note that two experiments run under the same conditions will yield different results,
  – i.e., data about a system obtained from experiments, even if they are thought of as a being deterministic, are always subject to uncertainty
Uncertainty quantification

- Uncertainty quantification (UQ)
  \[ \implies \text{the task of determining information about the uncertainty in the outputs of a system, given information about the uncertainty in the inputs} \]
  - of course, a system may have additional inputs that are not uncertain

- We consider systems governed by partial differential equations (PDEs) having random inputs
  - thus, now

  \[ \textit{UQ is the task of determining information about the uncertainties in the outputs of a PDE, given information about the uncertainties in its inputs} \]

  - the solution of the partial differential equation defines the mapping from the inputs to the solution of the PDE
• Often, the solution of the PDE is not the primary output of interest
  – outputs obtained by, in some manner, post-processing the solution of the PDE are of greater interest
  – of course, one still has to obtain a solution of the PDE to determine the output of interest so that, now
    \(UQ\) is the task of determining information about the uncertainty in an output of interest that depends on the solution of a PDE, given information about the uncertainty in the inputs of the PDE

• There are several approaches available for UQ, both deterministic and probabilistic
  - e.g., fuzzy sets, possibility theory, interval arithmetic, evidence theory, ...
  – we consider probabilistic approaches to UQ so that the uncertainties in the input of the PDE are described in terms of statistical quantities
    probability density functions, expected values, variances, covariance functions, higher statistical moments, ...
• Thus, finally, for us,

UQ is the task of determining statistical information about the uncertainty in an output of interest that depends on the solution of a PDE, given statistical information about the uncertainty in the inputs of the PDE

• We define

output of interest = O_{OI} = a quantity that depends on the solution of the PDE for which we want to know some statistical information

quantity of interest = Q_{OI} = the desired statistical information about the chosen output of interest
Types of uncertain inputs

- **Random parameters** $\leftarrow$ a finite number of random numbers
  - these numbers may be collected into a vector so that this type of input may be viewed as random vector input
  - if one thinks of “knobs” in an experimental setup, the value at which the knob is set is subject to uncertainty
  - each parameter may vary independently according to its own given one-dimensional PDF or, alternately, the parameters may vary according to a given joint multivariate PDF
  - we use the notation $\mathbf{y} \in \mathbb{R}^N$ to denote a random vector having components $y_1, y_2, \ldots, y_N$
  - examples include physical properties such as Young’s modulus, speed of sound, chemical reaction rates, interest rates, ...
Random fields defined over the spatial/temporal domain

- values that vary randomly from one point to another and/or from one time instant to another

- random fields are identically distributed which means that all values are determined by sampling from a single probability density function (PDF)

- we use the notation \( \eta(x, t; \omega) \) to denote random fields that depend on position \( x \) and time \( t \) with values drawn from a PDF \( \rho(\omega) \)
• **White noise** $\iff$ i.i.d. (independent and identically distributed) random fields
  - in addition to being identically distributed
    the value of the field at a point $x$ and time $t$ is independent
    from its value at all other spatial points and time instants
  - white noise random fields are necessarily **uncorrelated**
  - often, the terminology “white noise” is reserved for uncorrelated Gaussian
    random fields, i.e., $\rho(\omega)$ is a Gaussian PDF
  - examples of white noise include thermal fluctuations, surface roughness, ...

• **Colored or correlated** random fields $\iff$ identically but not independently
distributed
  - instead their random values over the space/time domain are determined
    according to a given (spatial/temporal) correlation structure
  - examples include rainfall amounts, bone densities, permeabilities of subsurface media, ...
• Spatially and/or time-dependent random fields
  whether they are correlated or not
  can also be expressed in terms of random parameters
  – however, because they are (infinite) stochastic processes
    an infinite number of random parameters is required

• However, for all three types of input noises, ultimately, on a computer one
  can only solve problems involving a finite number of random parameters
  – in the white and colored noise cases, one has to discretize the random fields
    so that they are approximated in terms of a finite number of parameters
It is important to note that white and colored random fields, although both are infinite stochastic processes, are, from both the theoretical and computational points of views, fundamentally different

- colored noise fields, because they are correlated and have finite variances, can in principle be approximated well, e.g., in a some mean-square sense, by a finite number of parameters

- white noise fields have infinite variances so that approximations, which out of computational necessity have to involve a finite number of parameters, converge in a much weaker sense

- as a result of the difference between the white and colored random fields, their discretization and use as inputs to a PDE have to be treated separately, i.e., they cannot be considered as special cases or generalizations of each other.
Output uncertainties

• Assume we are given random inputs parameterized by the random parameters \( \{y_n\}_{n=1}^N \), which for convenience we collect in the vector \( \mathbf{y} = (y_1, y_2, \ldots, y_N)^T \)

• The solution of the PDE is not only a function of the spatial position \( x \) and/or time \( t \), it is also a function of the random parameters \( \mathbf{y} \)

• A realization of a solution \( u(x, t; \mathbf{y}) \) of a PDE is a solution obtained for a specific choice for the random parameters \( \mathbf{y} \)
  – in general, there is no interest in individual realizations

• As already mentioned above
  one instead wants to obtain statistical information
  e.g., expected values, variances, standard deviations, higher statistical moments, probability density functions, etc
• In addition, as has also already been mentioned, one does not usually want such information about the solution of the PDE at all points \( x \) and times \( t \), but instead, one is interested in statistical information about an output of interest that depends on the solution of the PDE
  
  – in turn, the desired statistical information about an output of interest is referred to as a quantity of interest (QoI)

  – approximating quantities of interest is an important goal of computational simulations involving PDEs with random inputs

  – we will shortly discuss various types of outputs of interest and quantities of interest

• Often, the determination of a QoI requires the evaluation of a multi-dimensional integral with respect to the parameters \( y_1, \ldots, y_N \)
  
  – because in general, such integrals cannot be evaluated exactly, quadrature rules have to be employed

  – thus, a subject of substantial interest is the construction of useful multi-dimensional quadrature rules
Surrogates for the solution of the PDE or for outputs of interest

- Many, if not most, methods used in practice for determining statistical information about an output of interest require the evaluation of that output for many choices of the parameter vector $\mathbf{y}$
  - e.g., for each of the quadrature points used to approximate a parameter integral of an output of interest
  - because the output of interest depends on the solution of the PDE, one is thus required to obtain many solutions of the PDE

- To alleviate this burden, one can use relatively few solutions of the PDE to build a surrogate
  - e.g., an interpolant or a reduced-order model for the solution of the PDE
  - subsequently, when a solution of the PDE is needed to evaluate a QoI, the surrogate is queried instead of the PDE itself
• Alternately, one can directly build a surrogate for the output of interest which again requires relatively few solutions of the PDE

• Thus, another subject of substantial interest is the construction of surrogates
Stochastic inverse problems

- We have tacitly assumed, as is often done, that we know the PDFs of the input parameters
  - actually, in practice, one usually does not know much about the statistics of the input variables
  - one is lucky if one knows a range of values, e.g., maximum and minimum values, for an input parameter, in which case one often assumes that the parameter is uniformly distributed over that range
  - if one is luckier, one knows the mean and variance for an input parameter, in which case one often assumes that the parameter is normally distributed
  - of course, one may be completely wrong in assuming such simple probability distributions for a parameter $\Rightarrow$ the two PDFs in the sketch have the same mean and variance
• The lack of information about the input parameters leads to the need to solve stochastic model calibration problems
  – in our context, model calibration is the task of determining statistical information about the inputs of a system, given statistical information about the outputs that can be obtained, e.g., from laboratory experiments or field observations.
• Ideally, one would like to identify the PDFs of the input parameters.
  – in practice, one has to often settle for obtaining less rich information about the input parameters such as their expected values and variances
• Of course, the PDE still maps the inputs to the outputs, so that determining the statistical information about input parameters is an inverse problem that usually involves an iteration in which guesses for the inputs of interest are recursively updated
  – there are several ways to effect the update, e.g., Bayesian, optimization, etc.
Clearly, model calibration problems are a particular case of more general stochastic inverse problems that include parameter identification, optimal and feedback control, and optimization problems.

- we do not consider stochastic inverse problems in these lectures.
The curse of dimensionality

- In almost all cases, i.e., for almost all methods, the computational costs associated with UQ for systems governed by PDEs increases with the number of parameters used to model the uncertainty
  - of course, this is something the numerical PDE community is used to; after all, they are all used to having problems in three spatial dimensions costing more to solve than those in two and one dimension
  - what they are not used to is that the number of random parameters \( N \) can be much larger than 2 or 3
  - as a result, many familiar numerical methods used for spatial discretization, e.g., for quadrature, become, in the UQ setting, too costly to use for approximation with respect to the random parameters
  - in fact, almost of all of them become unviable even for a moderate number of parameters
– this is because the costs increase very, very quickly, even explosively, as the number of parameters increases

    “the curse of dimensionality” is a phrase in ubiquitous use to describe this phenomenon

• In one way or another, all work on UQ for systems governed by PDEs is directed at alleviating the curse

  – unfortunately, at this date, the curse cannot be completely removed

  – all one can try to do is to increase the value of $N$ at which the curse comes into play

    i.e., to increase the value of $N$ at which our computer resources become exhausted.
Quantities of interest

- Statistical quantities of interest (QoIs) are very often determined as follows:
  - from the solution $u(x, t; y)$ of the PDE, define an output of interest (OoI) $F(u)$
  - choose what statistical information about $F(u)$ is desired
  - use the choice made for the output of interest and the choice made for the desired statistical information to define an integrand $G(u)$
  - obtain the QoI by integrating $G(u)$ over the parameter domain $\Gamma$
For example

– suppose the output of interest is \( F(u) = |\nabla u| \), the magnitude of the spatial gradient of the solution \( u \) of the PDE

– further suppose the statistical information one wants is the second moment of \( F(u) \)

then, \( G(u) = (F(u))^2 = |\nabla u|^2 \)

the exponent 2 follows from the second moment choice

– suppose \( \rho(y) \) denotes the PDF of the input parameters

– then, the QoI is given by

\[
\text{QoI}(x, t) = \int_{\Gamma} G(u) \rho(y) \, dy = \int_{\Gamma} |\nabla u|^2 \rho(y) \, dy
\]

– note that the QoI is an integral over the parameter domain \( \Gamma \) and that this particular QoI is a function of \( x \) and \( t \)
• More generally, QoIs are often defined in terms of the parameter integral

\[ QoI = \int_{\Gamma} G(u(x, t; y); x, t, y) \rho(y) \, dy \]

– the integrand \( G(u(x, t; y); x, t, y) \) depends on the solution \( u(x, t; y) \) of the PDE and perhaps also depends explicitly on the physical variables \( x \) and \( t \) and the random parameters \( y \)

– the functional form of \( G(u(x, t; y); x, t, y) \) is determined from an output of interest \( F(u(x, t; y); x, t, y) \) and the particular statistical information one seeks about that output

– in the latter regard, if the statistical information is, for example, the expected value of \( F \), then \( G = F \); if it is the \( m^{th} \) moment of \( F \), then \( G = F^m \); and if it is the variance of \( F \), then
\[ G = (F - E[F])^2 \]

• We next discuss some possibilities for how outputs of interest \( F(u; x, t, y) \)
are connected to the solution \( u(x, t; y) \) of the PDE and how they are combined with a choice of desired statistical information to define a QoI

\[ E[F] = \int_{\Gamma} F(y) \rho(y) \, dy \] denotes the expected value or statistical mean of \( F(y) \)
Examples of outputs of interest and quantities of interest

• **Pointwise outputs of interest**

  the output of interest $F(u(x, t; y); x, t, y)$ could be pointwise related to the solution $u$ of the PDE

  - e.g., $F = u$, $F = u^2$, $F = \nabla u$, $F = |\nabla u|^2$, $F = \partial u/\partial t$, ..., for $x \in \mathcal{D}$ and $t \in (t_0, t_1)$

    - here, $\mathcal{D}$ denotes a spatial domain with boundary $\partial \mathcal{D}$
      and $(t_0, t_1)$ denote a time interval.

  - in such cases, the QoI is a function of $x$ and $t$

  - this type of output of interest $F$ arises, e.g., in determining QoIs such as the

    - statistical mean or expected value of the solution $u$ of the PDE

    $$\bar{u}(x, t) = E[u(x, t; y)] = \int_{\Gamma} u(x, t; y) \rho(y) \, dy$$

    for which we have $F = u$ and $G = F = u$
- the second moment of the solution $u$ of the PDE

$$M_2(x, t) = E\left[ (u(x, t; y))^2 \right] = \int_{\Gamma} (u(x, t; y))^2 \rho(y) \, dy$$

for which we have $F = u$ and $G = F^2 = u^2$

- the variance of the solution $u$ of the PDE

$$V(x, t) = E\left[ (u(x, t; y) - \bar{u}(x, t))^2 \right] = \int_{\Gamma} (u(x, t; y) - \bar{u}(x, t))^2 \rho(y) \, dy = M_2(x, t) - (\bar{u}(x, t))^2$$

for which we have $F = u$ and $G = (F - E[F])^2 = (u - E[u])^2$

one may define similar QoIs by substituting any quantity having a point-wise dependence on $u$ (e.g., $u^2$, $u^3$, $|u|$, $\nabla u$, $e^u$, etc.) wherever $u$ appears in the above expressions

- e.g., we could be interested in the third statistical moment of $e^u$, in which case we have the QoI

$$QoI(x, t) = \int_{\Gamma} e^{3u(x, t; y)} \rho(y) \, dy$$

for which we have $F = e^u$ and $G = F^3 = (e^u)^3 = e^{3u}$
Spatial and temporal integral outputs of interest

Of greater interest are QoIs involving outputs of interest $F$ that are spatial and/or temporal integrals of a quantity $H(u(x, t, y); x, t, y)$ that depends pointwise on the solution $u$ of the PDE

- suppose that the pointwise quantity $H(u(x, t, y); x, t, y)$ is derived from the solution $u(x, t, y)$ of the PDE, e.g., $H = u, H = u^4, H = |\nabla u|^2, H = \partial u/\partial t, H = e^u, \ldots$

- further suppose that the output of interest is the spatial/temporal average of $H$ so that

$$F(y) = \int_{t_0}^{t_1} \int_D H(u(x, t, y); x, t, y) \, dx \, dt$$

- suppose that the statistical information that we want about $F$ is its expected value so that $G(y) = F(y)$

---

The case of pointwise outputs of interest is merely the case $F = H$
we then have the QoI

\[ QoI = E[G(y)] = \int_G G(y) \rho(y) \, dy = \int_G F(y) \rho(y) \, dy \]

\[ = \int_G \int_{t_0}^{t_1} \int_D H(u(x, t, y); x, t, y) \rho(y) \, dx \, dt \, dy \]

i.e., the statistical expected value (or mean) of the spatial/temporal average of \( H \)

if instead, we are interested in the second moment of the spatial/temporal average of \( H \), we would have \( G(y) = (F(y))^2 \) and the QoI

\[ QoI = E[G(y)] = \int_G G(y) \rho(y) \, dy = \int_G (F(y))^2 \rho(y) \, dy \]

\[ = \int_G \left( \int_{t_0}^{t_1} \int_D H(u(x, t, y); x, t, y) \, dx \, dt \right)^2 \rho(y) \, dy \]

note that these two QoIs are just numbers, i.e., they are not functions of \( x, t, \) or \( y \).
there are other possible outputs of interest $F$ that involve spatial and/or temporal integrals of $H$, including, among many other possibilities,

$$F(t; y) = \int_{D} H(u; x, t, y) \, dx$$

\begin{cases}
- \text{spatial average of } H \\
- F \text{ and } G \text{ are functions of } t \text{ and } y \\
- \text{the QoI is a function of } t
\end{cases}

$$F(x; y) = \int_{t_0}^{t_1} H(u; x, t, y) \, dt$$

\begin{cases}
- \text{temporal average of } H \\
- F \text{ and } G \text{ are functions of } x \text{ and } y \\
- \text{the QoI is a function of } x
\end{cases}

$$F(t; y) = \int_{\partial D} H(u; x, t, y) \, dx$$

\begin{cases}
- \text{average of } H \text{ over boundary of } D \\
- F \text{ and } G \text{ are functions of } t \text{ and } y \\
- \text{the QoI is a function of } t
\end{cases}

$$F(t; y) = \int_{\tilde{D}} H(u; x, t, y) \, dx$$

\begin{cases}
- \text{average of } H \text{ over subset } \tilde{D} \subset D \\
- F \text{ and } G \text{ are functions of } t \text{ and } y \\
- \text{the QoI is a function of } t.
\end{cases}
— for example, suppose \( u(x, t; y) \) denotes a flow velocity field and \( \rho \) denotes the fluid density which we assume is constant and deterministic; then, the expected value of the total kinetic energy in the domain \( \mathcal{D} \) at time \( t \) is given by

\[
Q_{\text{ol}}(t) = \frac{\rho}{2} \left( \int_{\Gamma} \left( \int_{\mathcal{D}} u(x, t; y) \cdot u(x, t; y) \, dx \right) \rho(y) \, dy \right),
\]

where

\[
H(u; x, t, y) = \alpha u \cdot u \\
F(t; y) = \alpha \int_{\mathcal{D}} u(x, t; y) \cdot u(x, t; y) \, dx
\]

\[
G(y) = F(y)
\]

\footnote{The usage of \( \rho \) here to denote fluid density, which is in ubiquitous use, should not be confused with our usage of \( \rho \) elsewhere to denote PDFs.}
• **Extreme value outputs of interest**

A class of important outputs of interest that do not involve spatial or temporal integrals are extreme values

- we again have a quantity $H(u(x, t, y); x, t, y)$ that depends pointwise on the solution $u$ of the PDE

- we now have that the output of interest is the maximum value\(^4\) of $H$ over the spatial domain $\mathcal{D}$ and the time interval $(t_0, t_1)$ so that

\[
F(y) = \max_{x \in \mathcal{D}, t \in (t_0, t_1)} H(u; x, t, y)
\]

- suppose that the statistical information that we want about $F$ is its expected value so that $G(y) = F(y)$

\(^4\)Of course, in some applications one might instead be interested in the minimum value
we then have the QoI

\[
\text{QoI} = \mathbb{E}[G(y)] = \int_{\Gamma} G(y) \rho(y) \, dy = \int_{\Gamma} F(y) \rho(y) \, dy
\]

\[
= \int_{\Gamma} \left( \max_{x \in D, \, t \in (t_0, t_1)} H(u(x, t, y); x, t, y) \right) \rho(y) \, dy,
\]

i.e., the statistical expected value (or mean) of the maximum value of \( H \) over the spatial domain \( D \) and time interval \((t_0, t_1)\)

if instead, we are interested in the second moment of the maximum value of \( H \) over the spatial domain \( D \) and time interval \((t_0, t_1)\), we would have \( G(y) = (F(y))^2 \) and the QoI

\[
\text{QoI} = \mathbb{E}[G(y)] = \int_{\Gamma} G(y) \rho(y) \, dy = \int_{\Gamma} (F(y))^2 \rho(y) \, dy
\]

\[
= \int_{\Gamma} \left( \max_{x \in D, \, t \in (t_0, t_1)} H(u(x, t, y); x, t, y) \right)^2 \rho(y) \, dy.
\]
– of course, there are other possible outputs of interest and QoIs of this type

– for example, in addition to the maximum value of $H$ over the spatial domain $\mathcal{D}$ and time interval $(t_0, t_1)$, we have, among many other possibilities,

$$F(t, y) = \max_{x \in \mathcal{D}} H(u; x, t, y)$$

\hspace{1em} \{ F \text{ is the maximum of } H \text{ over the spatial domain } \mathcal{D} \}

$$F(x, y) = \max_{t \in (t_0, t_1)} H(u; x, t, y)$$

\hspace{1em} \{ F \text{ is the maximum of } H \text{ over the time interval } (t_0, t_1) \}

$$F(t, y) = \max_{x \in \partial \mathcal{D}} H(u; x, t, y)$$

\hspace{1em} \{ F \text{ is the maximum of } H \text{ over the boundary } \partial \mathcal{D} \text{ of the spatial domain } \mathcal{D} \}

$$F(t, y) = \max_{x \in \tilde{\mathcal{D}}} H(u; x, t, y)$$

\hspace{1em} \{ F \text{ is the maximum of } H \text{ over the subset } \tilde{\mathcal{D}} \text{ of the spatial domain } \mathcal{D} \}. 


– as an example of a QoI of this type, let $\mathbf{u}$ denote the displacement vector in a solid

- then, for a homogeneous, isotropic material, the stress tensor is given by $\mathbf{T} = \lambda \nabla \cdot \mathbf{E} + 2\mu \mathbf{E}$, where we have the Lamé constants $\lambda$ and $\mu$ and the strain tensor $\mathbf{E} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$

- note that $\mathbf{T} = \mathbf{T}(\mathbf{u})$

- as a measure of the size of the stress, we use the Forbenius norm $\|\mathbf{T}\|_{Frob}$

- now consider the expected value of the maximum of the Forbenius norm of the stress as the QoI

- we then have that

$$H(\mathbf{u}; \mathbf{x}, t, \mathbf{y}) = \|\mathbf{T}(\mathbf{u})\|_{Frob}$$

$$F(\mathbf{y}) = \max_{\mathbf{x} \in \mathcal{D}, \ t \in (t_0, t_1)} H(\mathbf{u}; \mathbf{x}, t, \mathbf{y}) = \max_{\mathbf{x} \in \mathcal{D}, \ t \in (t_0, t_1)} \|\mathbf{T}(\mathbf{u}(\mathbf{x}, t, \mathbf{y}))\|_{Frob}$$

$$G(\mathbf{y}) = F(\mathbf{y}) \quad \text{QoI} = \int_{\Gamma} \left( \max_{\mathbf{x} \in \mathcal{D}, \ t \in (t_0, t_1)} \|\mathbf{T}(\mathbf{u}(\mathbf{x}, t, \mathbf{y}))\|_{Frob} \right) \rho(\mathbf{y}) d\mathbf{y}$$

---

$^5$Again, the usage of $\lambda$ and $\mu$ here to denote the Lamé constants should not be confused with our usage of these symbols for other purposes elsewhere
- Event probability outputs of interest

- an important class of QoIs that arises in risk assessment, failure probability, and many other settings, requires the determination of the probability that a quantity derived from the solution of the PDE exceeds (or is less than) a given value

- often in this setting, the output of interest $F$ is a functional of $u$ so that $F(u(x, t; y))$ is scalar valued

- then, the problem at hand is to determine, given a threshold value $F_0$

$$\text{QoI} = \text{prob} \left[ F(u(x, t; y)) \geq F_0 \right]$$
this QoI does not seem, at first glance, to involve an integral over the parameter space \( \Gamma \), but, in fact, this QoI can also be expressed in this manner, i.e., we have

\[
QoI = \text{prob} \left[ F(u(x, t; y)) \geq F_0 \right] = \int_{\Gamma} \chi_{F \geq F_0} \rho(y) \, dy
\]

where the characteristic or indicator function is given by

\[
\chi_{F \geq F_0} = \begin{cases} 
1 & \text{if } F(u(x, t; y)) \geq F_0 \\
0 & \text{otherwise}.
\end{cases}
\]

– thus, with \( G(y) = \chi_{F \geq F_0} \), we again see that the QoI has the form

\[
QoI = \int_{\Gamma} G(y) \rho(y) \, dy
\]
returning to the example of the stress in a solid, we note that there is usually much more interest in knowing the probability that some measure of the stress exceeds a given threshold value (e.g., a value for which a structure may fall down or crack) compared to knowing statistics such as the expected value of that measure.

- thus, we now have, for some $\tau > 0$,

$$H(u; x, t, y) = \|T(u)\|_{Frob}$$

$$F(y) = \max_{x \in D, \ t \in (t_0, t_1)} H(u; x, t, y) = \max_{x \in D, \ t \in (t_0, t_1)} \|T(u(x, t, y))\|_{Frob}$$

$$G(y) = \chi_{F(y) \geq \tau} = \chi_{\max_{x \in D, \ t \in (t_0, t_1)} \|T(u(x, t, y))\|_{Frob} \geq \tau}$$

$$QoI = \int_{\Gamma} \chi_{\max_{x \in D, \ t \in (t_0, t_1)} \|T(u(x, t, y))\| \geq \tau} \rho(y) \, dy = \text{prob}[\|T(u)\|_{Frob} \geq \tau]$$
• **Determining PDFs**
  
  – the ideal QoI is the **PDF of the output of interest**

  – if the PDF could be determined, then obtaining statistics of the output of interest would be a simple matter

  – methods for determining approximate output PDFs include binning methods, moment methods, Bayesian methods, and deriving a partial differential equation for the PDF
Quadrature rules for parameter integrals

- The QoIs we have considered require the evaluation of the integral over the parameter domain $\Gamma$ of some integrand $G(\cdot)$
  - i.e., one must evaluate integrals of the type
  \[
  \int_\Gamma G(u(x, t; y); x, t, y) \rho(y) \, dy
  \]

- Such integrals cannot, in general, be evaluated exactly so that they have to be approximated using a quadrature rule
  - it makes a huge difference (when choosing a quadrature rule) whether or not the integrand is smooth or not
  - in particular, discontinuous integrands, as is the case if the characteristic function is involved, are especially troublesome
Quadrature rules have the general form

\[ Q_{\text{ol}} = \int_{\Gamma} G(u(x, t; y); x, t, y) \rho(y) \, dy \]

\[ \approx \sum_{q=1}^{Q} w_q G(u(x, t; y_q); x, t, y_q) \rho(y_q) \]

for some choice of

quadrature weights \( \{w_q\}_{q=1}^{Q} \) (real numbers)

and

quadrature points \( \{y_q\}_{q=1}^{Q} \) (points in the parameter domain \( \Gamma \))

– alternately, sometimes the probability density function is used in the determination of the quadrature points and weights so that instead one ends up with the approximation

\[ Q_{\text{ol}} = \int_{\Gamma} G(u(x, t; y); x, t, y) \rho(y) \, dy \]

\[ \approx \sum_{q=1}^{Q} w_q G(u(x, t; y_q); x, t, y_q) . \]
- in one dimension
  - examples of the first type of quadrature rule would be the trapezoidal and Simpson rules and their composite versions
  - examples of the second would be Gauss-Hermite or Gauss-Legendre rules

- Monte Carlo integration, the simplest rule, proceeds as follows:
  - randomly select $Q$ points in $\Gamma$ according to the PDF $\rho(y)$
  - evaluate the integrand at each of the sample points
  - average the values so obtained, i.e., for all $q$, $w_q = \frac{1}{Q}$.

- we will consider Monte Carlo and other quadrature rules in another lecture
Where is the probability?

- In a great many, if not most papers on UQ in the PDE setting, the reader will find a paragraph similar to

  Let $\mathcal{D} \subset \mathbb{R}^d$, $d = 1, 2, 3$, be a bounded, Lipschitz domain and let $\mathcal{F}$ be a complete probability space, where $\mathcal{F}$ denotes the set of outcomes, $\mathcal{F}$ denotes the $\sigma$-algebra of events, and $\mathbb{P} : \mathcal{F} \to [0, 1]$ is a complete probability measure. Given random fields $a, f : \mathcal{D} \times \Omega \to \mathbb{R}$, the model problem can be stated as follows: find $u : \overline{\mathcal{D}} \times \Omega \to \mathbb{R}$ such that almost surely,

  $\mathcal{L}(a)u = f$ for $x \in \mathcal{D}$

  - here $\mathcal{L}(a)u = f$ represents your favorite PDE with a random coefficient $a$ and forcing function $f$
• Then, in many instances, assumptions are made on $a$ and $f$, such as
  
  – the random fields $a$ and $f$ are approximated in terms of a finite number $N$ of random variables $\{y_1(\omega), \ldots, y_N(\omega)\}$.
  
  – the image $\Gamma_n := y_n(\Omega)$ of $y_n$ is bounded for all $n \in \{1, \ldots, N\}$ and, with $\Gamma = \prod_{n=1}^{N} \Gamma_n$, the random variables $\{y_1, \ldots, y_N\}$ have a joint probability density function $\rho \in L^\infty(\Gamma)$.

• After this, much less often it is stated that

  It follows from the Doob-Dynkin Lemma that the solution $u$ to the PDE can also be characterized in terms of the random variables $\{y_1, \ldots, y_N\}$. The solution $u(x, \omega)$ thus has a deterministic, parametric equivalent $u(x, y)$ and the probability space $(\Gamma, \mathcal{B}, \rho(y)\,dy)$ takes the place of $(\Omega, \mathcal{F}, \mathbb{P})$, where $\mathcal{B}$ denotes the Borel $\sigma$-algebra generated by the open subsets of $\Gamma$. Here, we have introduced the random vector $y = (y_1, \ldots, y_N)^T$. In what follows, we will therefore denote the solution of the PDE by $u(x, y)$ for $y \in \Gamma$ and $x \in D$. 
Such passages are often gratuitous in that none of $\Omega$, $\mathcal{F}$, or $\mathbb{P}$ are ever mentioned again.

In fact, in papers that only deal with algorithmic descriptions and testing, i.e., no rigorous analyses are involved, it is not necessary to do so, once one has made the assumptions stated above.

- In fact, everything that follows such passages would make perfect sense if one just starts with the last sentence of the last “quotation”

- with a few exceptions, this is what we do in here because these lectures only deal with algorithmic descriptions and testing

  - most of the topics considered in these lectures are also considered from a more rigorous approach in the review article

in fact, for the most part we assume that $a$ and $f$ as well as $u$ are functions of the random vector $y$ as well as of the spatial/temporal variables $x$ and $t$

we merely need to know the PDFs for the random parameters
We note here that a PDE often involves more than one input random field
  
  – e.g., we could have a random coefficient $a$ and a random forcing function $f$
  
  – in general, two such random fields are seldom related to each other so that it is reasonable to assume that they are defined in terms of two different random vectors

\[ \mathbf{y}_a \in \Gamma_a \subset \mathbb{R}^{N_a} \quad \text{and} \quad \mathbf{y}_f \in \Gamma_f \subset \mathbb{R}^{N_f} \]

with

\[ \mathbf{y}_a = (y_{a,1}, \ldots, y_{a,N_a})^T \quad \text{and} \quad \mathbf{y}_f = (y_{f,1}, \ldots, y_{f,N_f})^T \]

and with respect to two different PDFs

\[ \rho_a(\mathbf{y}_a) \quad \text{and} \quad \rho_f(\mathbf{y}_f) \]

– to economize notation in such cases, we will however just write $a(\mathbf{x}, t, \mathbf{y})$ and $f(\mathbf{x}, t, \mathbf{y})$ with $\mathbf{y} = (y_1, \ldots, y_N)^T$

- where it is understood that in most cases $N = N_a + N_f$, $y_n = y_{a,n}$ for $n = 1, \ldots, N_a$, and $y_n = y_{f,n-N_a}$ for $n = N_a + 1, \ldots, N$
• The value of a white noise random field $\eta_{\text{white}}(x, t; \omega)$ at every point in space and at every instant in time is independently chosen according to a Gaussian PDF.

• From an algorithmic point of view, the inclusion of $\omega$ in the argument of the random field indicates that:
  - the values of the random field at any point $x$ and any time instant $t$ are determined by first sampling $\omega$ according to a one-dimensional PDF $\rho_g(\omega)$
    here this is the standard normal distribution $\mathcal{N}(0, 1)$
  - then adjusting the sample value taken for the point $x$ and time instant $t$ so that it corresponds to a sample from the normal distribution having given mean $\mu_{\text{white}}(x, t)$ and variance $\sigma^2_{\text{white}}(x, t)$
  - as a result, the sample value used is given by $\mu_{\text{white}}(x, t) + \sigma_{\text{white}}(x, t)\omega$
  - often, $\sigma^2_{\text{white}}$ is chosen to be constant
The covariance function\(^6\) corresponding to a white noise random field \(\eta_{\text{white}}(\mathbf{x}, t; \omega)\) is given by

\[
\text{Cov}_{\text{white}}(\mathbf{x}, t, \mathbf{x}', t') = \sigma_{\text{white}}(\mathbf{x}, t) \sigma_{\text{white}}(\mathbf{x}', t') \delta(\mathbf{x} - \mathbf{x}') \delta(t - t')
\]

where \(\delta(\cdot)\) denotes the Dirac delta function

- thus, the variance \(V_{\text{white}}(\mathbf{x}, t)\) of white noise is infinite\(^7\) so that white noise cannot describe a real process

- notwithstanding this observation, white noise random fields are the most common random inputs used in the PDE setting

- later we comment on why numerical simulations involving white noise inputs do not “see” this infinity.

---

\(^6\)The mean of a random field \(\eta(\mathbf{x}, t; \omega)\) is defined by \(\mu_\eta(\mathbf{x}, t) = E[(\eta(\mathbf{x}, t; \cdot))\), where \(E\) denotes the expectation with respect to the PDF \(\rho_\eta(\omega)\). Its covariance is given by

\[
\text{Cov}_\eta(\mathbf{x}, t; \mathbf{x}', t') = E\left[(\eta(\mathbf{x}, t; \cdot) - \mu_\eta(\mathbf{x}, t))(\eta(\mathbf{x}', t'; \cdot) - \mu_\eta(\mathbf{x}', t'))\right]
\]

\[
= E\left[\eta(\mathbf{x}, t; \cdot)(\eta(\mathbf{x}', t'; \cdot))\right] - \eta_{\text{white}}(\mathbf{x}, t; \omega)^2,
\]

and its variance by \(V_\eta(\mathbf{x}, t) = \text{Cov}_\eta(\mathbf{x}, t; \mathbf{x}, t)\).

\(^7\)One should not confuse the variance of the white noise field \(\eta(\mathbf{x}, t; \cdot)\), which is infinite, with the variance \(\sigma_{\text{white}}^2(\mathbf{x}, t)\) of the Gaussian PDF from which sample values of the random field are drawn, which is finite.
• Obviously, in any computer simulation, one cannot sample the Gaussian distribution at every point of the spatial domain and at every instant of time so that white noise is replaced by **discretized white noise**

  – among the means available for discretizing white noise in the PDE setting, **grid-based methods** are the most popular

  – truncated expansions in terms of Hermite polynomials are another means used to discretized white noise; we don’t have time to discuss this approach
**Grid-based methods for discretizing white noise**

- For simplicity, we consider time-independent white noise random fields
- To define a single realization of grid-based discretized white noise
  - we first subdivide the spatial domain $\mathcal{D}$ into $N$ non-overlapping, covering subdomains $\{\mathcal{D}_n\}_{n=1}^N$
  - then, for each $n = 1, \ldots, N$, we let
    
    $|\mathcal{D}_n|$ denote the volume of the subdomain $\mathcal{D}_n$
    
    $\chi_n(x)$ denote the characteristic (or indicator) function corresponding to the subdomain $\mathcal{D}_n$
    
    $y_n(\omega)$ denotes an $\mathcal{N}(0, 1)$ i.i.d. random number

so that the value of $y_n(\omega)$ is independent of the value of $y_{n'}(\omega)$ for all $n' \neq n$ but both $y_n(\omega)$ and $y_{n'}(\omega)$ are drawn from the same PDF $\rho_g(\omega)$
Then, for some constants \( \{a_n\}_{n=1}^N \), we seek an approximation of the white noise random field \( \eta_{\text{white}}(x; \omega) \) of the form\(^8\)

\[
\eta^N_{\text{white}}(x; y) = \mu_{\text{white}}(x) + \sum_{n=1}^N a_n \chi_n(x) y_n(\omega) \approx \eta_{\text{white}}(x; \omega)
\]

where \( y \in \mathbb{R}^N \) denotes the vector having components \( y_n, n = 1, \ldots, N \)

- note that this is a piecewise constant approximation (with respect to the subdivision \( \{D_n\}_{n=1}^N \) of the spatial domain \( D \)) of the random field \( \eta_{\text{white}}(x; \omega) - \mu_{\text{white}}(x) \)

The covariance of the discrete random field \( \eta^N_{\text{white}}(x; \omega) \) is given by

\[
\text{Cov}^N_{\text{white}}(x, x') = \begin{cases} 
  a_n^2 & \text{if both } x, x' \in D_n \\
  0 & \text{otherwise}
\end{cases}
\]

\(^8\)Usually the mean function \( \mu_{\text{white}}(x) \) is also approximated by a function \( \mu^N_{\text{white}}(x) \) defined with respect to the grid. For example, one could use the piecewise constant approximation \( \mu^N_{\text{white}}(x) = \sum_{n=1}^N \mu(x_n^*) \chi_n(x) \), where \( x_n^* \) denotes the centroid of the subdomain \( D_n \).
Because pointwise values of the covariance
\[
\text{Cov}_{\text{white}}(\mathbf{x}, \mathbf{x}') = \sigma_{\text{white}}(\mathbf{x})\sigma_{\text{white}}(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')
\]
of a spatially dependent white noise random field are not defined, we determine, for \( n = 1, \ldots, N \), the coefficients \( a_n \) by matching the averages over \( D_n \) of the covariance \( \text{Cov}_{\text{white}}(\mathbf{x}, \mathbf{x}') \) and its approximation \( \text{Cov}_{\text{white}}^N(\mathbf{x}, \mathbf{x}') \)

- in this manner we obtain that
\[
a_n^2 = \frac{1}{|D_n|^2} \int_{D_n} \sigma^2_{\text{white}}(\mathbf{x}) \, d\mathbf{x}.
\]

- approximating \( \sigma_{\text{white}}(\mathbf{x}) \) for \( \mathbf{x} \in D_n \) by its value at the centroid \( \mathbf{x}_n^* \) of \( D_n \), we then have
\[
a_n = \frac{\sigma_{\text{white}}(\mathbf{x}_n^*)}{\sqrt{|D_n|}}
\]

- so that the discretized white noise random field is given by
\[
\eta^N_{\text{white}}(\mathbf{x}; \mathbf{y}) = \mu_{\text{white}}(\mathbf{x}) + \sum_{n=1}^{N} \frac{\sigma_{\text{white}}(\mathbf{x}_n^*)}{\sqrt{|D_n|}} \chi_n(\mathbf{x}) y_n(\omega) \approx \eta_{\text{white}}(\mathbf{x}; \omega)
\]

Thus, via discretization, white noise has been reduced to the case of \( N \) random parameters.
• If \( \sigma_{\text{white}}(x) = \sigma_{\text{white}} = \text{constant} \), as is often the case, then

\[
\eta_{\text{white}}^N(x; y) = \mu_{\text{white}}(x) + \sigma_{\text{white}} \sum_{n=1}^{N} \frac{1}{\sqrt{|D_n|}} \chi_n(x) y_n(\omega) \approx \eta_{\text{white}}(x; \omega)
\]

• In one dimension, for constant variance \( \sigma_{\text{white}}^2 \) and zero expected value and for a uniform grid of size \( h \), we have the well-known formula

\[
\eta_{\text{white}}^N(x; y) = \frac{\sigma_{\text{white}}}{\sqrt{h}} \sum_{n=1}^{N} \chi_n(x) y_n(\omega) \approx \eta_{\text{white}}(x; \omega)
\]

for approximating a white noise random field

• We have constructed an approximate white noise field based on sampling in subdomains
  
  – this is convenient for finite element spatial discretization schemes
  
  – for other spatial discretization schemes, e.g., finite difference methods, it may be more convenient to sample at grid points; the construction process we have discussed is easily amended to this case
• It is important to note that the variance of the discretized white noise field is finite

  – in fact, we have, for \( n = 1, \ldots, N \),

\[
V^N_{\text{white}}(x) = \text{Cov}^N_{\text{white}}(x; x) = \frac{\sigma^2_{\text{white}}(x^*_n)}{|D_n|} \quad x \in D_n
\]

  which clearly is finite for all \( x \in D \)

  – this is one reason why, in simulations, the fact that the variance of a white noise random field is infinite does not cause codes to fail

  – however, note also that as the spatial grid size goes to zero, i.e., as \(|D_n| \to 0\), that the variance of discretized white noise goes to infinity

• Time-dependent white noise fields can be treated in an entirely similar manner
• What about non-Gaussian white noise fields?
  
  – it seems an easy matter to sample according to a different PDF instead of the normal distribution; indeed, this is done in practice

  – however, there are issues that arise when doing so, not the least of which is that the Gaussian PDF is the only one which is completely defined by its mean and variance
Realizations of grid-based discretized white noise over the same time interval in a square subdivided into 2, 8, 32, 72, 238, 242, 338, and 512 triangles
Realizations of grid-based discretized white noise over two different time intervals in a square subdivided into the same number of triangles

- We clearly see that the discretized white noise is piecewise constant in space and time
- One also sees that it is much smoother than the white noise random field it approximates.
• We have seen that the white noise case has been reduced to a case of a large but finite number of parameters
  – if we refine the spatial grid and/or reduce the time step, the number of parameters increases

• White noise is not defined pointwise and its variance is unbounded
  – thus, one cannot expect discretized white noise, which is well-defined pointwise and has finite variance, to converge to white noise pointwise or even with respect to an $L^2$ norm
  – what can be shown is convergence of the covariance function, e.g., if the white noise field has zero mean, we have (for spatially-dependent white noise)

\[
\lim_{N \to \infty, |\mathcal{D}_n| \to 0} E\left[ \eta_{\text{white}}^N(x; y) \eta_{\text{white}}^{N_{\text{space}}, N_{\text{time}}}(x; y) \right] = E\left[ \eta_{\text{white}}(x; y) \eta_{\text{white}}(x; y) \right] = \sigma_{\text{white}}(x) \sigma_{\text{white}}(x') \delta(x - x')
\]
PDEs with white noise inputs

- Formally, we can write an evolution equation with white noise forcing as

\[
\frac{\partial u}{\partial t} = A(u; x, t) + f(x, t) + B(u; x, t)\eta_{\text{white}}(x, t; \omega) \quad \text{in } D \times (t_0, t_1]
\]

- \(A\) is a possibly nonlinear deterministic operator
- \(f\) is a deterministic forcing function
- \(B\) is a possibly nonlinear deterministic operator
- \(\eta_{\text{white}}\) is the input white noise random field
- \(f\) and \(B\) can possibly take care of cases with means \(\neq 0\) and variances \(\neq 1\)
• If $B$ does no depend on $u$, we have additive white noise:\[ \frac{\partial u}{\partial t} = A(u; x, t) + f(x, t) + b(x, t) \eta_{white}(x, t; \omega) \]

  – often in practice, $b$ is constant

  – if the values of $\eta$ are drawn from the standard normal distribution, then $b = \sigma =$ standard deviation

• If $B$ depends on $u$, we have multiplicative white noise

  – of particular interest is the case of $B$ linear in $u$:

\[ \frac{\partial u}{\partial t} = A(u; x, t) + f(x, t) + b(x, t) u \eta_{white}(x, t; \omega) \]

---

*In the stochastic processes community, the terminology *stochastic partial differential equation* (SPDE) is reserved for equations of this typ, i.e., for PDEs driven by additive Gaussian white noise. That term has been usurped by the computational community so that they refer to any PDE with any type of random input as an SPDE. This difference in the use of the terminology “SPDE” causes communications problems between the two communities, which is why in this talk, despite sometimes having to use somewhat cumbersome language, we refer to the more general case as PDEs with random inputs.*
• White noise inputs need not be restricted to forcing terms in the PDE
  – in practice, they can also appear in the coefficients of the PDEs and boundary and initial conditions, in the data in boundary and initial conditions, and even in the definition of the domain
  – in certain cases, one must be careful about choosing white noise coefficients, e.g., coefficient in elliptic equation

• It is natural to use the same grids in space and time to discretize the white noise as are used to effect spatial and temporal discretization of the PDE
  – thus, if one refines the spatial grid and the time step, one also refines the grid and time step for the white noise discretization
• Example: Consider the multiplicative white noise problem

\[
\begin{cases}
-\Delta u = f(x) + b(x)u \eta_{white}(x; \omega) & \text{in } D \\
\quad \text{in } D \\
\quad \text{in } \partial D
\end{cases}
\]

Then, an approximate realization \( u_N(x; y) \) is determined by sampling sampling the PDF \( N \) times to obtain the \( N \) values \( y_n, n = 1, \ldots, N \) and then solving

\[
\begin{cases}
-\Delta u_N + f(x) = b(x)u_N \eta_{white}^N(x, t; y) \\
\quad = b(x)u_N \sum_{n=1}^{N} \frac{\sigma(x^*)y_n}{\sqrt{|D_n|}} \chi_{ns}(x) & \text{in } D \\
\quad \text{in } \partial D
\end{cases}
\]

for the approximation \( u_N(x; y) \), where \( y \) is the vector with components \( y_n, n = 1, \ldots, N \).
A realization of the white noise input in one-dimension and two realizations of the solution of a simple two-point boundary value problem for a second-order ordinary differential equation with additive and linear multiplicative noise. The ranges of the ordinates in the three plots are $[-150, 150]$, $[-0.3, 0.2]$, and $[-0.1, -0.08]$, respectively, whereas the ranges of the abscissas, i.e., the spatial axis, are all the same.
Quantities of interest for white noise inputs

- We have the approximation of the QoI given by

\[
\text{QoI} = \int_{\Gamma} G(u(x; y); x, y) \rho(y) \, dy \approx \frac{1}{Q} \sum_{q=1}^{Q} G(u_N(x; y_q); x, y_q)
\]

where \( \{y_q\}_{q=1}^{Q} \) is a set of independent vectors in \( \mathbb{R}^N \), each one having independent components sampled from the same PDF \( \rho_g(y) \)

- To evaluate the integrand, one must solve the (discretized) PDE \( Q \) times, once for each \( y_q, q = 1, \ldots, Q \)

- So we are using random sampling to approximate the random field input and to do the quadrature needed to approximate the QoI

- This error in this approximation is proportional to \( 1/\sqrt{Q} \)
  - it is very slow!!
  -- however, this result is valid independent of the number \( N \) of parameters
Later on in these lectures we discuss other quadrature rules that, for smooth integrands, are more efficient, i.e., they have the same accuracy for a smaller number of quadrature points compared to Monte Carlo integration.

- However, for the case of white noise inputs, the integrand is not smooth so that one cannot take advantage of those other quadrature rules.

- In addition, unlike Monte Carlo sampling, the convergence of those rules deteriorate as the number of parameters $N$ increases (the curse of dimensionality) and here $N$, which is inversely related to the spatial grid size, is huge.
III - COLORED NOISE
Correlated random fields

• We next consider correlated random fields $\eta(x, t; \omega)$ for which at each point $x$ in a spatial domain $\mathcal{D}$ and at each instant $t$ in a time interval $[t_0, t_1]$, the value of $\eta$ is determined by a random variable $\omega$ whose values are drawn from a given probability density function $\rho(\omega)$

  however, unlike the white noise case, these draws are not independent

  $\iff$ the covariance function of a correlated random field $\eta(x, t; \omega)$ does not reduce to delta functions.

• In rare cases, a formula for the random field is “known”

  – more often, only the mean $\mu_\eta(x, t)$ and covariance function $\text{Cov}_\eta(x, t; x', t')$ are known at points $x$ and $x'$ in $\mathcal{D}$ and time instants $t$ and $t'$ in $[t_0, t_1]$

  – in this case, we do not have a formula for $\eta(x, t; \omega)$ so that we cannot evaluate $\eta(x, t; \omega)$ when we need to
– for example, if $\eta(x, t; \omega)$ is a coefficient or forcing function in a PDE, then to determine an approximate realization of the PDE we need to evaluate $\eta(x, t; \omega)$ at specific points $x$ (e.g., spatial quadrature points in a finite element assembly process) and specific instants of time $t$ (e.g., discrete times in a temporal finite difference quotient approximation to a time derivative) used to define the discretized PDE.

- Actually, we usually do not even know the covariance function so that, instead, it is often guessed; common guesses are

  exponential covariance: $\text{Cov}(x, t; x', t') = e^{-\frac{|x-x'|}{L} - \frac{|t-t'|}{T}}$

  (multivariate) Gaussian covariance: $\text{Cov}(x, t; x', t') = e^{-\frac{|x-x'|^2}{L^2} - \frac{|t-t'|^2}{T^2}}$

  (general) Gaussian covariance: $\text{Cov}(x; x') = e^{-(x-x')^T \Sigma^{-1} (x-x')}$,

where $L$ denotes the correlation length, $T$ the correlation time, and $\Sigma$ a symmetric positive definite matrix; large $L$ and $T$ correspond to long-range correlation whereas small $L$ and $T$ correspond to short-range correlation.
Note that, in general, covariance functions are symmetric and non-negative - for the sake of simplicity, we will assume they are positive.

- **Examples of known covariance functions**
  - one-dimensional Weiner process or Brownian motion $W_t$ for which
    \[ \mu(t) = 0 \quad \text{and} \quad \text{Cov}(t, t') = \min(t, t') \]

  - the Ornstein-Uhlenbeck (or mean-reverting) process defined by the stochastic ordinary differential equation
    \[ d\eta = \theta(\bar{\mu} - \eta)dt + \sigma dW_t \]
    where $W_t$ is a Wiener process so that we have that $dW_t$ is Gaussian white noise.
    \[ \theta = \text{the deterministic speed of reversion} \]
    \[ \bar{\mu} = \text{the long-run equilibrium level or mean respectively} \]
    \[ \sigma^2 = \text{the variance} \]
- the solution of this stochastic ODE is given by

\[ \eta(t; \omega) = \eta_0 e^{-\theta t} + \bar{\mu}(1 - e^{-\theta t}) + \sigma e^{-\theta t} \int_0^t e^{\theta s} dW_s \]

- \( \eta_0 \) denotes the deterministic initial condition

- the corresponding mean is given by

\[ \mu_\eta(t) = \mathbb{E}(\eta(t; \cdot)) = \eta_0 e^{-\theta t} + \bar{\mu}(1 - e^{-\theta t}) \]

\[ \implies \text{hence the process is "mean reverting"} \]

\[ \text{as } t \to \infty, \mu_\eta(t) \to \bar{\mu} \text{ for any initial data } \eta_0 \]

- the covariance function is given by

\[ \text{Cov}_\eta(t; t') = \sigma^2 e^{-\theta(t+t')} \mathbb{E} \left( \int_0^t e^{\theta s} dW_s \right) \left( \int_0^{t'} e^{\theta u} dW_u \right) \]

\[ = \frac{\sigma^2}{2\theta} e^{-\theta(t+t')} \left( e^{2\theta \min(t,t')} - 1 \right) \]
– the **Matérn** covariance

$$\text{Cov}(\mathbf{x}; \mathbf{x}') = \sigma^2 \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu \frac{|\mathbf{x} - \mathbf{x}'|}{L}}\right)^\nu K_\nu \left(\sqrt{2\nu \frac{|\mathbf{x} - \mathbf{x}'|}{L}}\right)$$

\(\sigma^2 = \text{the variance}\)
\(\Gamma(\cdot) = \text{the Gamma function}\)
\(K_\nu(\cdot) = \text{the modified Bessel function of the second kind}\)
\(\nu = \text{a parameter} \quad \nu = \frac{1}{2} \longrightarrow \text{Matérn reduces to exponential}\)

• In other cases, Fourier spectral information about a correlated random field is known
  – an example is pink noise that has, in one dimension, a \(1/f\) power spectrum, in contrast to the \(1/f^0\) and \(1/f^2\) power spectra for white noise and Brownian noise, respectively; here \(f\) denotes the frequency
  – again, we do not have a formula for the random field that we can evaluate at spatial and temporal points
Here, we consider the case for which the mean and covariance functions of a random field are known or guessed

- in this case, we would like to find a simple formula depending on only a few parameters whose mean and covariance functions are approximately the same as the given mean and covariance functions
- expansions in terms of orthogonal polynomials provide a means of approximating correlated random fields
- there also exist grid-based methods for this purpose
- we will mostly consider perhaps the most popular approach: the Karhunen-Loève (KL) expansion of a correlated random field $\eta(x, t; \omega)$

Given the mean and covariance functions for a random field $\eta(x, t; \omega)$, the KL expansion provides a simple formula for the random field in terms of an infinite number of uncorrelated random parameters $\{y_n\}_{n=1}^\infty$

- in principle, a truncated KL expansion can be used whenever one needs a value of $\eta(x, t; \omega)$ at a point $x$ and a time $t$
To keep things simple, we discuss KL expansions for the case of time-independent random fields; extension to the case of time-dependent fields is straightforward.

Let $\eta(x; \omega)$ denote a Gaussian random field, i.e., at each point $x$, we draw a sample from a standard Gaussian PDF.

Given the mean $\mu_{\eta}(x)$ and covariance $\text{Cov}_{\eta}(x, x')$ of a random field $\eta(x; \omega)$, the eigenpairs $\{\lambda_n, b_n(x)\}_{n=1}^{\infty}$ satisfy the eigenvalue problem

$$\int_{\mathcal{D}} \text{Cov}_{\eta}(x, x') b(x') \, dx' = \lambda b(x) \quad \forall x \in \mathcal{D}. $$

Due to the symmetry of $\text{Cov}_{\eta}(\cdot; \cdot)$, the eigenvalues $\lambda_n$ are real and the eigenfunctions $b_n(x)$ can be chosen to be real and orthonormal, i.e.,

$$\int_{\mathcal{D}} b_n(x) b_{n'}(x) \, dx = \delta_{nn'}. $$
due to the positivity\textsuperscript{10} of $\text{Cov}_\eta(\cdot; \cdot)$, the eigenvalues are all positive - without loss of generality, they may be ordered in non-increasing order $\lambda_1 \geq \lambda_2 \geq \cdots$

- Then, the random field $\eta(x; \omega)$ admits the KL expansion

$$\eta(x; \omega) = \mu_\eta(x) + \sum_{n=1}^{\infty} \sqrt{\lambda_n} b_n(x) y_n(\omega)$$

where $\{y_n(\omega)\}_{n=1}^{\infty}$ are sampled independently from the standard normal distribution

\textsuperscript{10}By non-negativity we mean that

$$\int_D \int_D \text{Cov}_\eta(x, x') b(x) b(x') dx' dx \geq 0$$

for all suitable functions $b(x)$. In general, covariance functions are non-negative, but, for the sake of simplicity, we assume positivity, i.e., the above relation holds strictly.
• Thus, the KL expansion accomplishes two important things
  – it provides a formula (albeit one involving and infinite number of parameters) for the correlated random field \( \eta(x; \omega) \) in terms of random parameters
  – it expresses the correlated random field in terms of uncorrelated parameters

• As an example of KL expansions, consider Brownian motion which has the KL eigenpairs

\[
\lambda_n = \frac{1}{(n - \frac{1}{2})^2 \pi^2} \quad \text{and} \quad b_n(t) = \sqrt{2} \sin \left( (n - \frac{1}{2}) \pi t \right)
\]

and the KL expansion

\[
W_t = \sqrt{2} \sum_{n=1}^{\infty} \frac{\sin \left( (n - \frac{1}{2}) \pi t \right)}{(n - \frac{1}{2}) \pi} y_n(\omega)
\]
The usefulness of the KL expansion results from the fact that the eigenvalues \( \{\lambda_n\}_{n=1}^{\infty} \) decay as \( n \) increases.

- how fast they decay depends on the smoothness of the covariance function \( \text{Cov}_{\eta}(x, x') \) and on the correlation length \( L \)

*Peaked covariance function with very small correlation length*

*Slowly decaying covariance function with large correlation length*

*Corresponding KL eigenvalues*

*Corresponding KL eigenvalues*
• The decay of the eigenvalues implies that truncated KL expansions

$$\eta_N(x; y) = \mu(x) + \sum_{n=1}^{N} \sqrt{\lambda_n} b_n(x) y_n(\omega)$$

can be accurate approximations to the exact expansions

- $y \in \mathbb{R}^N$ is the vector with components $y_n$, $n = 1, \ldots, N$
- if one wishes for the relative error of the truncated expansion to be less than a prescribed tolerance $\varepsilon$, i.e., if one wants\textsuperscript{11}

$$\frac{E[\|\eta_N - \eta\|]}{E[\|\eta\|]} \leq \varepsilon,$$

where here $\| \cdot \|$ denotes the $L^2(D)$ norm, one should choose $N$ to be the

\textsuperscript{11}For the sake of simplicity, for the time being we ignore the mean function $\mu(x)$; being a deterministic quantity, its role in error estimation is straightforward to analyze.
smallest integer such that

\[
\sum_{n=N+1}^{\infty} \lambda_n \leq \varepsilon^2 \quad \text{or, equivalently,} \quad \frac{\sum_{n=1}^{\infty} \lambda_n}{\sum_{n=1}^{N} \lambda_n} \geq 1 - \varepsilon^2
\]

– these estimates easily follow because the orthonormality of the eigenfunctions \( \{b_n\} \) imply that, e.g., \( \|\eta\|^2 = \sum_{n=1}^{\infty} \lambda_n \)

– of course, in practice, the infinite sums appearing in these estimates would themselves have to be truncated, with the number of retained terms being considerably larger than \( N \)

– this requires computing more eigenpairs than the \( N \) one ends up using in the truncated KL expansion
It is easily seen that the covariance function for the truncated KL expansion is given by

\[
\text{Cov}_\eta^N(x, x') = \sum_{n=1}^{N} \lambda_n b_n(x) b_n(x')
\]

then, a theorem due to Mercer shows the convergence of the covariance function

– specifically, if \( \text{Cov}_\eta(x, x') \) is continuous, symmetric, non-negative definite, and square integrable on \( \mathcal{D} \times \mathcal{D} \), we then have

\[
\lim_{N \to \infty} \max_{x, x' \in \mathcal{D}} | \text{Cov}_\eta(x, x') - \text{Cov}_\eta^N(x, x') | = 0
\]
• There are also estimates available that relate the decay of the eigenvalues \( \lambda_n \) to the smoothness of the covariance function
  
  for example, if \( \text{Cov}_\eta(x, x') \in L^2(D \times D) \) is piecewise analytic on \( D \times D \), then there exist constants \( c_1 \) and \( c_2 \) independent of \( n \) such that
  
  \[
  0 \leq \lambda_n \leq c_1 e^{-c_2 n^{1/d}} \quad \text{for all integers } n \geq 1
  \]
  
  i.e., we have exponential decay in the eigenvalues; here, \( d \) denotes the spatial dimension

  if instead \( \text{Cov}_\eta(x, x') \) is merely piecewise \( H_0^k(D \times D) \), where \( H_0^k(D \times D) \) denotes the Sobolev space of functions having square integrable derivatives of order up to \( k \) that also vanish on the boundary of \( D \times D \), we have the algebraic decay estimate

  \[
  0 \leq \lambda_n \leq c_2 n^{-k/d} \quad \text{for all integers } n \geq 1
  \]
To summarize

- we approximate a Gaussian random field \( \eta(x; \omega) \) by its \( N \)-term truncated KL expansion
- the parameters \( \{y_n\}_{n=1}^N \) are uncorrelated
- because we are considering multivariate Gaussian random variables, the parameters are also independent
- thus, we now have a formula for an approximation to a correlated Gaussian random field that involves a finite number of independent random parameters
- we can then use any of the methods to be discussed for problems involving a finite number of random parameters to solve problems defined in terms of Gaussian random fields
One important issue is the well posedness of a PDE when using KL representations of random fields.

Suppose the coefficient $a(x; \omega)$ of an elliptic PDE is a random field.

- it cannot be a Gaussian random field because, regardless of how positive is the mean of the random field, there is a non-zero probability for admitting negative values, i.e., for sign changes in $a(x; \omega)$; this is not allowable.
• One way to get around this is to truncate (and rescale) the Gaussian PDF so that the samples of the random variables $y_n$ are always large enough so that the approximate KL expansion is greater than some given value $a_{min} > 0$ for all $x \in \bar{D}$

• Another commonly used means for avoiding negative values of the coefficient $a(x; \omega)$ is to use log-normal random fields, i.e., we assume the coefficient has the form

$$a(x; \omega) \approx a_{min} + e^{\mu(x) + \sum_{n=1}^{N} \sqrt{\lambda_n} b_n(x) y_n(\omega)}$$

where $\{y_n(\omega)\}_{n=1}^{N}$ are uncorrelated, and therefore independent, multivariate Gaussian random variables

  - clearly, we now have that $a(x; \omega) \geq a_{min} > 0$

  - one issue with this approach that is seldom addressed is that if one uses the given mean and covariance functions to determine the truncated KL expansion of the Gaussian random field appearing in the exponent, then the resulting log-normal random field will have different mean and covariance functions
If one wants the log-normally distributed coefficient $a(x; \omega)$ to correspond to a given mean function $\tilde{\nu}(x)$ and covariance function $\text{Cov}(x, x')$, then the mean $\mu(x)$ and covariance function $\text{Cov}(x, x')$ of the Gaussian field whose KL expansion is the exponent should be chosen to be

$$\mu(x) = \ln (\tilde{\nu}(x')) - \frac{1}{2} \ln \left(1 + \frac{\text{Cov}(x, x)}{(\tilde{\nu}(x))^2}\right)$$

and

$$\text{Cov}(x, x') = \ln \left(1 + \frac{\tilde{\text{Cov}}(x, x')}{\tilde{\nu}(x)\tilde{\nu}(x')}\right)$$
In many applications, random field inputs are assumed to be not normally distributed.

We can still use the KL expansion to express the field in terms of uncorrelated random parameters \( \{y_n\}_{n=1}^{\infty} \).

- formally, one only need draw samples \( y_n, n = 1, 2, \ldots \), from a desired non-Gaussian PDF.

There is a problem with doing this, which is usually ignored in the PDEs with random input literature.

- although the independence of random variables implies that they are uncorrelated, the converse is not in general true, i.e., uncorrelated does not imply independence.
a classic example is to

- let $y_1 \in [-1, 1]$ denote a uniformly distributed random variable
- then let $y_2 = y_1^2$
- clearly, $y_1$ and $y_2$ are not independent
- however, it is easy to show that $\text{Cov}(y_1, y_2) = 0$
  i.e., $y_1$ and $y_2$ are uncorrelated

• In fact, uncorrelated implies independence only if the components of $\mathbf{y}$ are multivariate Gaussian variables, i.e., if the components of $\mathbf{y}$ are jointly Gaussian

• So, for general non-Gaussian random fields, our only recourses are to
  – assume that the KL parameters are independent (what is usually done), in which case we simply express the random field in terms of its KL expansion
  or, as we already did log-normal random fields
  – using CDFs and inverse CDFs, express the non-Gaussian random field in terms of a Gaussian random field and then use the KL expansion for the latter
Spatial approximations of Karhunen-Loève expansions

- Consider a truncated KL expansion

\[ \eta_N(x; y) = \mu_\eta(x) + \sum_{n=1}^{N} \sqrt{\lambda_n} b_n(x) y_n(\omega). \]

- In practice, one uses spatially approximate random fields, e.g., if \( \{\Phi_j\}_{j=1}^{\hat{J}} \) denotes a finite element basis, we use approximations of the form

\[ \mu_\eta(x) \approx \mu^h_\eta(x) = \sum_{j=1}^{\hat{J}} \mu_j \Phi_j(x) \]

and

\[ b_n(x) \approx b^h_n(x) = \sum_{j=1}^{\hat{J}} b_{jn} \Phi_j(x) \]
• We then have that the truncated KL expansion approximation \( \eta_N(x; \omega) \) of the random field \( \eta(x; \omega) \) is itself spatially approximated as

\[
\eta(x; \omega) \approx \eta_N(x; y) \approx \eta^h_N(x; y) = \mu^h(x) + \sum_{n=1}^{N} \sqrt{\lambda_n^h} b_n^h(x) y_n(\omega)
\]

\[
= \sum_{j=1}^{\hat{J}} \mu_j \Phi_j(x) + \sum_{n=1}^{N} \sqrt{\lambda_n^h} y_n(\omega) \sum_{j=1}^{\hat{J}} b_{jn} \Phi_j(x).
\]

• In general, the finite element basis used to spatially discretize the truncated KL expansion need not be the same as that used to effect the spatial discretization of the PD

– however, it may make sense to define both bases in terms of the same grid
• We still want the random variables \( \{y_n\}_{n=1}^N \) to have zero mean and to be uncorrelated, i.e., \( E[y_n] = 0 \) and \( E[y_n y_{n'}] = \delta_{nn'} \).

• The coefficients \( \{\hat{\mu}_j\}_{j=1}^{\hat{J}} \) are found from the linear system
  \[
  \mathbf{M}\bar{\mu} = \bar{a}
  \]
where, for \( j, j' = 1, \ldots, \hat{J} \), we have the Gram matrix
  \[
  \mathbf{M}_{j',j} = \int_D \Phi_j(x)\Phi_{j'}(x) \, dx
  \]
and the vectors
  \[
  (\bar{\mu})_j = \mu_j, \quad \text{and} \quad (\bar{a})_j = \int_D \mu_\eta(x)\Phi_{j'}(x) \, dx
  \]

• Next, consider the (generalized) matrix eigenvalue problem
  \[
  \mathbf{K}\vec{b}^h = \lambda^h \mathbf{M}\vec{b}^h,
  \]
where
  \[
  \mathbf{K}_{j',j} = \int_D \int_D \text{Cov}_\eta(x; x')\Phi_{j'}(x')\Phi_j(x) \, dx' \, dx
  \]
and \( (\vec{b}^h)_j = b_j^h \).
We select the first $N$ eigenpairs $\{\lambda_n^h, b_n^h\}_{n=1}^N$ and then set $b_{jn} = (\bar{b}_n)_j$ for $j = 1, \ldots, \hat{J}$.

we then obtain the truncated, spatially discretized approximation to the random field $\eta(x; y)$ given by

$$\eta_N^h(x; y) = \sum_{j=1}^{\hat{J}} \mu_j \Phi_j(x) + \sum_{n=1}^N \sqrt{\lambda_n^h} \ y_n(\omega) \sum_{j=1}^{\hat{J}} b_{jn} \Phi_j(x)$$

the $y_n$’s are uncorrelated random variables
Example: piecewise constant approximations

- choose the spatial basis \( \{ \Phi_j(x) \}_{j=1}^{\hat{J}} \) to be piecewise constants with respect to the triangulation \( \mathcal{D} = \bigcup_{j=1}^{\hat{J}} \mathcal{D}_j \), i.e., \( \Phi_j(x) = 1 \) if \( x \in \mathcal{D}_j \) and \( \Phi_j(x) = 0 \) otherwise

- let \( |\mathcal{D}_j| \) denote the volume of \( \mathcal{D}_j \)

- we then have that \( \mathbf{M} = \mathbf{D} = \text{diag}\{ |\mathcal{D}_j| \} \)

- suppose we use a piecewise constant quadrature rule to approximate the spatial integrals in the definition of \( \mathbf{K} \)

  - we use the quadrature weights \( \{|\mathcal{D}_j|\}_{j=1}^{\hat{J}} \) and

    the quadrature points are the centers of masses \( \{x^*_j\}_{j=1}^{\hat{J}} \) of the corresponding elements \( \mathcal{D}_j \)
we then have that

\[ K = DCD \]

where

\[ C_{jj'} = \text{Cov}_\eta(x^*_j, x^*_j') \]

and the spatially discretized KL-eigenvalue problem\(^{12}\) is given by \(^{13,14}\)

\[ DCD \vec{b} = \lambda^h D \vec{b}. \]

note that if

\[ \hat{\vec{b}} = D^{1/2} \vec{b} \]

one can transform the generalized eigenvalue problem

\[ DCD \vec{b} = \lambda^h D \vec{b} \]

to the standard eigenvalue problem

\[ D^{1/2} CD^{1/2} \hat{\vec{b}} = \lambda^h \hat{\vec{b}}. \]

\(^{12}\)For the time being, we assume that \( N = \hat{J} \), we are using the whole eigenspace in the spatially discretized KL expansion.

\(^{13}\)It is perhaps tempting to use the simpler and more direct discretized eigenvalue problem \( C \vec{b} = \lambda^h \vec{b} \); however, this problem does not properly scale the eigenproblem; the equally simple eigenproblem \( C \vec{b} = h^{-d} \lambda^h \vec{b} \), where \( h \) is a measure of the grid size, is properly scaled but results in a loss of accuracy unless one has a uniform grid; note that on a uniform grid we have that \( |D_j| = \alpha h^{-d} \) for all \( j \) and for some \( \alpha \) so that \( D \) is a scalar matrix and the eigenvalue problem then reduces to \( C \vec{b} = \alpha^{-1} h^{-d} \lambda^h \vec{b} \).

\(^{14}\)The same formulas hold the white noise case as well; we essentially end up with the grid-based approximation obtained previously for white noise. In this case, we have that \( K = D \) so that the eigenvalues \( \lambda^h_n \) do not decay and the eigenvectors \( \vec{b}_n \) are the unit vectors. For this reason, one must choose \( N = \hat{J} \), i.e., one must keep all the terms in the approximation of the random field; this means that as the grid is refined, the number of random parameters increases proportionally to the increase in the number of grid cells.
the eigenvalues \( \{ \lambda_n^h \}_{n=1}^J \) are real and positive and the eigenvectors \( \{ \hat{b}_n \}_{n=1}^J \)
may be chosen to be orthonormal, i.e, \( \hat{b}_n^T \hat{b}_{n'} = \delta_{nn'} \)

this implies that the generalized eigenvectors \( \vec{b} \) are \( \mathbb{D} \)-orthonormal, i.e.,
\( \vec{b}_n^T \mathbb{D} \vec{b}_{n'} = \delta_{nn'} \)

letting \( B \) and \( \hat{B} \) denote matrices whose columns are the vectors \( \vec{b}_n \) and \( \hat{b}_n \), respectively, we have that
\[
\hat{B} = \mathbb{D}^{1/2} B, \quad \mathbb{D}^{1/2} \mathbb{C} \mathbb{D}^{1/2} = \hat{B} \Lambda \hat{B}^T, \quad \text{and} \quad \mathbb{C} = B \Lambda B^T,
\]
where \( \Lambda = \text{diag}\{ \lambda_n^h \} \). Note that \( \hat{B}^T \hat{B} = I \) and that \( B^T \mathbb{D} B = I \).
Matrix form of approximate KL expansions

- It is perhaps instructive to examine spatially discretized KL expansions in matrix notation

- To this end, let

\[ \eta_{N,j} = \mu_j + \sum_{n=1}^{N} \sqrt{\lambda_n^h} b_{jn} y_n \]

so that

\[ \eta^h_N(x; \omega) = \sum_{j=1}^{\hat{j}} \eta_{N,j} \Phi_j(x) \]

- Then, letting \((y)_n = y_n, (\eta_N)_j = \eta_{N,j},\) and \((\mu)_j = \mu_j,\) we have that

\[ \eta_N = \mu + B \Lambda^{1/2} y = \mu + D^{-1/2} \hat{B} \Lambda^{1/2} y, \]

where \(E(y) = \overrightarrow{0}\) and \(E(yy^T) = I,\) i.e., the components of \(y\) have zero mean and are uncorrelated. It is easily verified that \(E(\eta_N) = \mu\) and \(E((\eta_N - \mu)(\eta_N - \mu)^T) = C\)
• For correlated random fields, the eigenvalues decay so that we may choose $N < \widehat{J}$ and truncate the eigenvalues
  
  i.e., set $\Lambda \approx \Lambda_N = \text{diag}\{\lambda_1^h, \ldots, \lambda_N^h, 0, \ldots, 0\}$
  
  and then approximate $\eta$ by
  
  $$\eta \approx \eta_N = \mu + B \Lambda_N^{1/2} y = \mu + D^{-1/2} \widehat{B} \Lambda_N^{1/2} y.$$ 

• Let $B_N$ denote the $\widehat{J} \times N$ matrix consisting of the first $N$ columns of $B$
  
  i.e., $B_N = B \begin{pmatrix} I_N \\ 0 \end{pmatrix}$

  where $I_N$ is the $N$-dimensional identity matrix, and similarly for $\widehat{B}_N$

• We then have that
  
  $$\eta \approx \eta_N = \mu + B_N \Lambda_N^{1/2} y_N = \mu + D^{-1/2} \widehat{B}_N \Lambda_N^{1/2} y_N,$$

  where $y_N$ is the $N$-vector consisting of the first $N$ components of $y$ and now $\Lambda_N = \text{diag}\{\lambda_1^h, \ldots, \lambda_N^h\}$
• Note that

\[ E(\eta_N) = \mu \quad \text{and} \quad E((\eta_N - \mu)(\eta_N^T - \mu)^T) = \mathbb{B}_N \Lambda_N \mathbb{B}_N^T. \]

Thus, we have an explicit approximation

\[ (\mu)_{\hat{j}} + \sum_{j=1}^{\hat{j}} (\tilde{\eta}_N)_j \Phi_j(\mathbf{x}) \]

of the random field \( \eta(\mathbf{x}; \omega) \) in terms of \( N \) uncorrelated random parameters.
IV - ABSTRACT SETTING FOR
STOCHASTIC FINITE ELEMENT METHODS
Having reduced input random fields to the case of a finite number of random parameters, from now on we consider the case in which the inputs in a PDE model, e.g., coefficients, forcing terms, initial data, etc., all depend on a finite number of random parameters. Such inputs could also depend on space and time and on deterministic parameters. The random parameters could arise from, e.g., a grid-based approximation of a white noise field or from a KL expansion of a correlated random field. They could also appear naturally in the definition of an input.

Ideally, we would know the PDF for each parameter although, as has already been mentioned, in practice, we often know very little about the statistics of input parameters. Here, however, we will assume that we know the PDFs for all the random input parameters.
• The finite number of parameters \( \{y_n\}_{n=1}^N \) we deal with can take on values anywhere in a Euclidean domain \( \Gamma \in \mathbb{R}^N \).

  – in general, these parameters come with a joint probability distribution \( \rho(y) \), where \( y = (y_1, y_2, \ldots, y_n) \) is the vector of parameters.

  – however, if the set \( \{y_n\}_{n=1}^N \) is independent, we necessarily have that

    \[ \Gamma = \Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_N \quad \text{and} \quad \rho(y) = \rho_1(y_1)\rho_2(y_2)\cdots\rho_N(y_N) \]

    where, for \( n = 1, \ldots, N \), \( \rho_n(y_n) \) denotes the PDF associated with \( y_n \).

  – in this case, \( \Gamma \) is a hyper-rectangle in \( \mathbb{R}^N \).

  – the extent of the hyper-rectangle in any direction \( y_n \) could be infinite, semi-infinite, or bounded depending on the domain of definition of the associated PDF \( \rho_n(y_n) \).

  – in case the parameters are also identically distributed, as is the case if they arise from a discretization of a random field, then all the one-dimensional PDFs are the same.
Example: a nonlinear parabolic equation

just to show the ubiquity of where random parameters can appear in a PDE model, consider the following problem that involves $N$ parameters divided among the many possible inputs in the model:

\[
\begin{cases}
    c(x, t; y_{Nb} + 1, \ldots, y_{Nc}) \frac{\partial u}{\partial t} - \nabla \cdot \left( a(x, t; y_1, \ldots, y_{Na}) \nabla u \right) + b(x, t; y_{Na+1}, \ldots, y_{Nb})u^3 \\
    = f(x, t; y_{Nc+1}, \ldots, y_{Nf}) \quad \text{on} \quad \mathcal{D}(y_{Ni+1}, \ldots, y_{Ng}; y_{Ng+1}, \ldots, y_N) \times (0, T) \\
    u = f_{dir}(x, t; y_{Nd+1}, \ldots, y_{Nd}) \quad \text{on} \quad \partial\mathcal{D}_{dir}(y_{Ni+1}, \ldots, y_{Ng}) \times (0, T) \\
    a(x, t; y_1, \ldots, y_{Na}) \frac{\partial u}{\partial n} = f_{neu}(x, t; y_{Nd+1}, \ldots, y_{Ne}) \quad \text{on} \quad \partial\mathcal{D}_{neu}(y_{Ng+1}, \ldots, y_N) \times (0, T) \\
    u = f_0(x; y_{Ne+1}, \ldots, y_{Ni}) \quad \text{on} \quad \mathcal{D}(y_{Ni+1}, \ldots, y_{Ng}; y_{Ng+1}, \ldots, y_N) \text{ for } t \in (0, T),
\end{cases}
\]

- we have the integers
  \[ 0 \leq N_a \leq N_b \leq N_c \leq N_f \leq N_d \leq N_e \leq N_i \leq N_g \leq N \]
- \( \{ y_n \}_{n=1}^{N} \) is the set of random parameters
- each of $a, b, c, f, f_{dir}, f_{neu}$, and $f_0$ is a given function of $x, t$, and a subset of the random parameters
- for example, \( f_{dir} \) is a function of \( x, t \), and the parameters \( y_n, n = N_f + 1, \ldots, N_d \)

- the Dirichlet boundary segment \( \partial D_{did} \) is parameterized by using the random parameters \( y_n, n = N_i + 1, \ldots, N_g \)

- similarly for the Neumann boundary segment \( \partial D_{neu} \)

- of course, \( \nabla \) and \( \nabla \cdot \) are operators involving spatial derivatives

- In order to avoid cumbersome notations, we will write
e.g., \( c(x, t; y), a(x, t; y), b(x, t; y) \), etc.

where \( y = (y_1, \ldots, y_N) \)

- that is, we will write the various inputs as functions of all the parameters, with it being understood that usually they are functions of different subsets, often disjoint subsets, of the parameters
Concrete example: an elliptic PDE with five random input parameters.

- as an illustration, consider the following admittedly completely artificial example:

\[
\begin{cases}
\nabla \cdot \left( a(x; y_1, y_2) \nabla u \right) = f(x; y_3, y_4) \quad \text{on } D(y_5) \\
u = 0 \quad \text{on } \partial D(y_5)
\end{cases}
\]

where

\[
a(x; y_1, y_2) = 3 + |x|(y_1^2 + \sin(y_2))
\]

\[
f(x; y_3, y_4) = y_3 e^{-y_4|x|^2}
\]

\[
D(y_5) = (0, 1) \times (0, 1 + 0.3y_5)
\]

with the PDFs specified for all the input parameters as

\[
\rho_1(y_1) = \mathcal{N}(0; 1) \\
\rho_2(y_2) = \mathcal{U}(0, 0.5\pi) \\
\rho_3(y_3) = \mathcal{N}(0; 2)
\]

\[
\rho_4(y_4) = \mathcal{U}(0, 1) \\
\rho_5(y_5) = \mathcal{U}(-1, 1)
\]

- here we have that \( \Gamma = (-\infty, \infty) \times (0, 0.5\pi) \times (-\infty, \infty) \times (0, 1) \times (-1, 1) \)

- \( \mathcal{U}(a, b) \) denotes the uniform distribution over the interval \((a, b)\)
Abstract setting for algorithmic discussions

- **Stochastic finite element methods (SFEMs)** refer to methods for which spatial discretization is effected using finite element methods.\(^\text{15}\)

- **Stochastic Galerkin methods (SGMs)** are a particular class of SFEMs for which discretization with respect to the random parameters is also effected using a Galerkin method
  - polynomial chaos methods are particular type of SGMs

- Another class of SFEMs are **stochastic sampling methods (SSMs)** for which points in the parameter domain \(\Gamma\) are sampled, then used as inputs for the PDE, and then ensemble averages of outputs of interest are computed
  - Monte Carlo (MC) methods provide the most straightforward example of SSMs
  - **stochastic collocation methods (SCMs)** are also SSMs

\(^{15}\)Throughout, we assume that spatial discretization is effected using finite element methods; most of what we say also holds for other spatial discretization approaches, e.g., finite difference, finite volume, spectral, etc.
Because we focus on finite element methods to effect spatial discretization, we use a weak formulation of an abstract PDE problem as a setting for discussions of computational algorithms.

We have the spatial domain $D \subset \mathbb{R}^d$ with boundary $\partial D$, where $d = 1, 2, \text{ or } 3$ denotes the spatial dimension; $x \in D$ denotes the spatial variable.

Let $u(x; y) \in X \times Z$ denote the solution of the PDE with random inputs.

The spatial function space $X$ is often a Sobolev space such as $H^1_0(D)$.

\footnote{For the sake of simplicity, we now consider stationary problems; almost all we have to say holds equally well for time-dependent problems.}
Generally, $Z = L^q_\rho(\Gamma)$, the space of functions of $N$ variables whose $q$-th power is integrable with respect to the joint PDF (the weight function) $\rho(\cdot)$, i.e., those functions $g(y)$ for which $\int_\Gamma |g(y)|^q \rho(y) \, dy < \infty$.

$q$ is chosen according to how many statistical moments one wants to have well defined.

The most common choice is $q = 2$ so that up to the second moments (and therefore variances) are well defined.

If $\{y_n\}_{n=1}^N$ is independent, we have that

$$L^q_\rho(\Gamma) = L^q_{\rho_1}(\Gamma_1) \times L^q_{\rho_2}(\Gamma_2) \times \cdots \times L^q_{\rho_N}(\Gamma_N),$$

where $L^q_{\rho_n}(\Gamma_n)$ denotes the space of functions that have integrable $q$-th powers with respect to the PDF $\rho_n(y_n)$. 
The abstract setting we use is then defined as follows:

for any $y \in \Gamma$, we seek a realization of the solution $u(x; y) \in X \times Z$ satisfying

$$
\int_D S(u; x, y) T(v) \, dx = \int_D v f(x, y) \, dx \quad \forall v(x) \in X
$$

- $S(\cdot; x, y)$ denotes a possibly nonlinear operator
- $T(\cdot)$ denotes a linear operator
- $f(x, y)$ denotes a given function
- in general, both $S$ and $T$ could involve derivatives with respect to $x$ but they do not involve derivatives with respect to $y$
• In general, we would have a sum of such terms, i.e., we would have that

$$\sum_{m=1}^{M} \int_{D} S_m(u; \mathbf{x}, \mathbf{y}) T_m(v) \rho(y) \, d\mathbf{x} = \int_{D} v f(x, y) \, d\mathbf{x} \quad \forall v(x) \in X$$

– however, without loss of generality, it suffices for our purposes to consider the simpler single-term form
**Example.** Suppose the PDE problem is given by

\[-\nabla \cdot \left( a(y) \nabla u \right) + c(y) u^3 = f(y) \quad \text{in } \mathcal{D} \quad \text{and} \quad u = 0 \quad \text{in } \partial \mathcal{D}.\]

- usually, \( a, c, \) and \( f \) depend on disjoint subsets of the parameters \( \{y_n\}_{n=1}^N \)
- of course, \( a, c, \) and \( f \) could also depend on \( x \), but, for the sake of notational simplicity, we suppress explicit reference to such dependences
- we have that \( X = H^1_0(\mathcal{D}) \) and the weak formulation: for any \( y \in \Gamma \), seek \( u(x; y) \in H^1_0(\mathcal{D}) \) such that

\[
\int_{\mathcal{D}} \left( a(y) \nabla u \right) \cdot \nabla v \, dx + \int_{\mathcal{D}} (c(y) u^3) v \, dx = \int_{\mathcal{D}} f(y) v \, dx \quad \forall \, v \in H^1_0(\mathcal{D}).
\]

- in the first term, we have that \( S(u, y) = a(y) \nabla u \) and \( T = \nabla v \)
- in the second term, we have that \( S(u, y) = c(y) u^3 \) and \( T = v \).
• We assume that all methods considered in this work use the same finite element method to effect discretization with respect to the spatial variables.

• Thus, one chooses a finite element space $X_J \subset X$ and then, for any $y \in \Gamma$, a realization of the finite element approximation $u_J(x; y) \in X_J$ of the solution $u(x; y) \in X$ is found by solving the discrete problem

$$
\int_D S(u_J; x, y) T(v_J) \, dx = \int_D v_J f(x, y) \, dx \quad \forall v_J(x) \in X_J.
$$

• Letting $\{\phi_j(x)\}_{j=1}^J$ denote a basis for the finite element space $X_J$
  
  – we have that $J$ denotes the dimension of the finite element space and also denotes the number of spatial degrees of freedom
  
  – we then have that, for any $y \in \Gamma$, the finite element approximation $u_J(x; y) \in X_J$ is given by

$$
u_J(x; y) = \sum_{j=1}^J \alpha_j(y) \phi_j(x)$$
– for any chosen $y \in \Gamma$, the coefficients $\{\alpha_j(y)\}_{j=1}^J$ are determined by solving the discretized system

$$
\int_{D} S \left( \sum_{j=1}^{J} \alpha_j(y) \phi_j(x); x, y \right) T (\phi_i(x)) \, dx
$$

$$
= \int_{D} \phi_i(x) f(x, y) \, dx \quad \forall \, i = 1, \ldots, J
$$

of $J$ equations in the $J$ unknowns $\{\alpha_j(y)\}_{j=1}^J$

• In practice, further discretization is required by applying a spatial quadrature rule to approximate the spatial integrals

– however, because we assume that the same finite element method is used for all methods discussed, the particular choice of spatial quadrature rule used does not affect the discussions, so that, here, we ignore the need for applying such rules
V - STOCHASTIC GALERKIN METHODS
General stochastic Galerkin methods

- Stochastic Galerkin methods (SGMs) are stochastic finite element methods for which discretization with respect to parameter space is also effected using a Galerkin approach

- It is entirely natural to then treat the solution $u(x; y) \in X \times Z$ of the PDE with random inputs as function of $d + N$ variables
  - i.e., of the $d$ spatial variables and of $N$ random parameters

- We now seek $u(x; y) \in X \times Z$ such that

$$\int_{\Gamma} \int_{\mathcal{D}} S(u; y) T(v) \rho(y) \, dx dy = \int_{\Gamma} \int_{\mathcal{D}} v f(y) \rho(y) \, dx dy \quad \forall v(x; y) \in X \times Z$$

\[17\] Of course, if $E[\cdot]$ denotes the expected value, we may write instead

$$E\left[\int_{\mathcal{D}} S(u; y) T(v) \rho(y) \, dx - \int_{\mathcal{D}} v f(y) \rho(y) \, dx\right] = 0$$

Also, again for simplicity, we consider stationary problems and again suppress explicit references to the possible dependences on the spatial variable $x$. Furthermore, we note again that the operators $S$ and $T$ do not involve derivatives with respect to the random parameters $y$. 
Using the concrete example previously introduced, we now have $X = H^1_0(D)$, $Z = L^2(\Gamma)$, and the weak formulation: seek $u(x; y) \in H^1_0(D) \times L^2(\Gamma)$ such that

\[
\int_D \int_\Gamma (a(y) \nabla u) \cdot \nabla v \rho(y) \, dy \, dx + \int_D \int_\Gamma (c(y)u^3) v \rho(y) \, dy \, dx = \int_D \int_\Gamma f(y) v \rho(y) \, dy \, dx \quad \forall v \in H^1_0(D) \times L^2(\Gamma)
\]

Functions of the parameters $y$ have to be discretized in much the same way functions of the (finite number of) spatial variables have to be discretized — thus, for discretization, we choose a $K$-dimensional subspace $Z_K \subset Z$ a $J$-dimensional subspace $X_J \subset X$

and then seek $u_{JK}(x; y) \in X_J \times Z_K$ that satisfies

\[
\int_\Gamma \int_D S(u_{JK}; y)T(v_{JK})\rho(y) \, dxdy = \int_\Gamma \int_D v_{JK}f(y)\rho(y) \, dxdy \quad \forall v_{JK}(x; y) \in X_J \times Z_K
\]
• Let \( \{ \psi_k(y) \}_{k=1}^K \) denote a basis for the parameter approximating space \( Z_K \)
\( \{ \phi_j(y) \}_{j=1}^J \) denote a basis for the spatial approximating space \( X_J \)

• Due to the product nature of the domain \( \mathcal{D} \times \Gamma \) and of the space \( X \times Z \),
  it is natural to seek an approximate solution \( u_{JK}(x; y) \in X_J \times Z_K \) of the form

\[
  u_{JK} = \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk} \phi_j(x) \psi_k(y)
\]

for some constants \( c_{jk}, j = 1, \ldots, J \) and \( k = 1, \ldots, K \)

• The coefficients \( c_{jk} \), and thus \( u_{JK} \), are determined by solving the (possibly nonlinear) system of \( JK \) equation in \( JK \) unknowns

\[
  \int_{\mathcal{D}} \int_{\Gamma} \rho(y) S \left( \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk} \phi_j(x) \psi_k(y), y \right) T \left( \phi_{j'}(x) \right) \psi_{k'}(y) \ dx \ dy
  = \int_{\mathcal{D}} \int_{\Gamma} \rho(y) \phi_{j'}(x) \psi_{k'}(y) f(y) \ dx \ dy
\]

for \( j' \in \{1, \ldots, J\} \) and \( k' \in \{1, \ldots, K\} \)
– of course, although the coefficients $c_{jk}$ depend on the choice of basis, the approximate solution does not

• In general, the integrals in the discrete system cannot be evaluated exactly so that quadrature rules must be invoked to effect approximate evaluations$^{18}$

– thus, we are led to the further discretized system

$$
\sum_{r=1}^{R} \hat{w}_r \rho(\hat{y}_r) \psi_{k'}(\hat{y}_r) \int_D S \left( \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk} \phi_j(x) \psi_k(\hat{y}_r), \hat{y}_r \right) T \left( \phi_{j'}(x) \right) dx
$$

$$
= \sum_{r=1}^{R} \hat{w}_r \rho(\hat{y}_r) \psi_{k'}(\hat{y}_r) \int_D \phi_{j'}(x) f(\hat{y}_r) dx
$$

for $j' \in \{1, \ldots, J\}$ and $k' \in \{1, \ldots, K\}$

$\Leftarrow$ we have quadrature weights $\{\hat{w}_r\}_{r=1}^{R}$

quadrature points $\{\hat{y}_r\}_{r=1}^{R}$ in $\Gamma$

$^{18}$Integrals with respect to the spatial domain $D$ must also be approximated using quadrature rules; we do not need to consider this issue because we assume that all methods discussed treat all aspects of the spatial discretization in the same manner
this quadrature rule need not be the same as the quadrature rule \( \{w_q, y_q\}^Q_{r=1} \)
used to obtain the approximation of a quantity of interest

• The discrete problem is a **coupled** (in physical and parameter spaces) \( JK \times JK \) system
  - i.e., a system \( K \) times the size of the \( J \times J \) finite element system
  - for example, if one needs \( J = 10^6 \) spatial degrees of freedom and \( K = 10^3 \)
    parameter degrees of freedom to obtain and acceptably accurate approximation, then the discrete system involves \( 10^9 \) equations in \( 10^9 \) unknowns

• On the other hand, one can solve for the approximate dependence of the solution \( u(x, y) \)
  with respect to both the spatial coordinates \( x \) and the random parameters \( y \)
  by solving a **single deterministic problem** of size \( JK \)
  - in particular, one does not have to explicitly sample the random parameters \( y \)
    and one does not have to determine multiple solutions of the PDE
• After the $c_{jk}$'s are determined by solving the single system, one has obtained the \textbf{explicit formula} for the approximate solution of the PDE with random inputs that can be evaluated at any point $x$ in the spatial domain $\mathcal{D}$ and at any point $y$ of the parameter domain $\Gamma$.

• In particular, one can determine, just by evaluation, $u_{JK}(x, y_q)$ at any quadrature point $y_q$ appearing in a quadrature rule approximation of a quantity of interest

– thus, we obtain the \textbf{stochastic Galerkin approximation} of the quantity of interest

$$Q_{oI} = \int_{\Gamma} G(u(x; y)) \rho(y) \, dy \approx \sum_{q=1}^{Q} w_q \rho(y_q) G(u(x; y_q))$$

$$\approx \sum_{q=1}^{Q} w_q \rho(y_q) G(u_{JK}(x; y_q))$$

$$= \sum_{q=1}^{Q} w_q \rho(y_q) G\left( \sum_{j=1}^{J} \sum_{k=1}^{K} c_{jk} \phi_j(x) \psi_k(y_q) \right) = Q_{oI, SGM}$$
To complete the description of the problem actually solved on a computer, one has to make specific choices for\(^{19}\)

- an approximating subspace \(Z_K \subset Z\)
- a basis \(\{\psi_k(\vec{y})\}_k^{K}\) for \(Z_K\)
- a quadrature rule \(\{\hat{w}_r, \hat{y}_r\}_R\) to approximate the parameter integrals in the discretized PDE
- another quadrature rule \(\{w_q, y_q\}_Q\) to approximate the parameter integrals in the discretized quantity of interest

For the parameter approximating space \(Z_K\), one could use, e.g., either of

- piecewise polynomial spaces with locally-supported basis functions, e.g., a finite element-type method
- polynomial spaces with globally-supported basis functions, e.g., a spectral-type method

\(\Rightarrow\) we now discuss methods of the second type

\(^{19}\)We assume that the approximating subspace \(S_J \subset S\), a basis \(\{\phi_j(\vec{x})\}_j^{J}\), and a quadrature rule for approximating spatial integrals used for spatial discretization have been already chosen.
Global polynomial approximating spaces

- Let $P_r$ be the set of all polynomials in $\mathbb{R}$ of degree less than or equal to $r$
  - let $\{\Theta_i(y)\}_{i=0}^r$ denote a basis for $P_r$
  - of course, there are an infinite number of possible bases, the simplest being the monomial basis for which $\Theta_i(y) = y^i$ for $i = 0, 1, \ldots, r$

- Let $p = (p_1, p_2, \ldots, p_N)$ denote a multi-index
  i.e., an $N$-vector whose components are non-negative integers
  and let $|p| = \sum_{n=1}^N p_n$

- For each parameter $y_n$, we use
  - polynomials of degree $M_n$
  - a basis $\{\Theta_{n,i_n}(y_n)\}_{i_n=1}^{K_n}$
  - for the sake of simplicity, we assume that $M_n = M$ for all $n$; there may be good reasons for sometimes choosing different degree polynomials for each parameter
• For a given integer $M \geq 0$, let $\{\psi_k(y)\}_{k=1}^K$ denote the set of distinct total degree multivariate polynomials for which

$$\left\{\psi_k(y)\right\}_{k=1}^K = \left\{ \prod_{n=1}^N \Theta_{n,i_n}(y_n) \right\} \text{ with } \Theta_{n,i_n}(y_n) \in P_M \text{ and } |p| \leq M$$

- the highest degree term in any of the multivariate polynomials is $M$
  - i.e., the sum of the exponents in any term in the polynomial is less than or equal to $M$
- thus, for example, if $N = 2$ and $M = 3$, we have terms like $y_1y_2$ and $y_1^3$ and $y_1^2y_2$ but not terms like $y_1y_2$

• The number of parameter degrees of freedom is given by

$$K = \frac{(N + M)!}{N! M!}$$

where $N$ denotes the number of random parameters and $M$ the maximal degree of any of the $N$-dimensional global polynomials used
for example, if $N = 2$ and $M = 3$, we have

$$|p| = p_1 + p_2 \leq M = 3$$

$$K = \frac{(N+M)!}{N!M!} = \frac{(2+3)!}{2!3!} = 10$$

and the set of 10 basis functions

$$\{\psi_1(y_1, y_2), \ldots, \psi_{10}(y_1, y_2)\} = \left\{ \Theta_{1,0}(y_1) \Theta_{2,0}(y_2), \Theta_{1,1}(y_1) \Theta_{2,0}(y_2), \Theta_{1,0}(y_1) \Theta_{2,1}(y_2), \Theta_{1,1}(y_1) \Theta_{2,1}(y_2), \Theta_{1,2}(y_1) \Theta_{2,0}(y_2), \Theta_{1,0}(y_1) \Theta_{2,2}(y_2), \Theta_{1,2}(y_1) \Theta_{2,1}(y_2), \Theta_{1,1}(y_1) \Theta_{2,2}(y_2), \Theta_{1,3}(y_1) \Theta_{2,0}(y_2), \Theta_{1,0}(y_1) \Theta_{2,3}(y_2) \right\}$$
Alternately, one could use the tensor product basis with now $p_n \leq M$ for all $n$

- now the highest degree term in any of the basis functions is $M$ in each $y_n$
- thus, if $N = 2$ and $M = 3$, we have not only have terms like $y_1^3$ and $y_1^2 y_2$, but we also have terms like $y_1^3 y_2$ and $y_1^3 y_2^3$
- the number of parameter degrees of freedom is now given by

\[ K = (M + 1)^N \]

where $N$ denotes the number of random parameters and $M$ the maximal degree in any variable $y_n$ of any of the $N$-dimensional global polynomials used
for example, if \( N = 2 \) and \( M = 3 \), we now have \( K = (M + 1)^N = (3 + 1)^2 = 16 \) and the set of 16 basis functions

\[
\left\{ \psi_1(y_1, y_2), \ldots, \psi_{16}(y_1, y_2) \right\} = \left\{ \begin{array}{c}
\Theta_{1,0}(y_1) \Theta_{2,0}(y_2) \\
\Theta_{1,1}(y_1) \Theta_{2,0}(y_2) \\
\Theta_{1,0}(y_1) \Theta_{2,1}(y_2) \\
\Theta_{1,1}(y_1) \Theta_{2,1}(y_2) \\
\Theta_{1,2}(y_1) \Theta_{2,0}(y_2) \\
\Theta_{1,0}(y_1) \Theta_{2,2}(y_2) \\
\Theta_{1,2}(y_1) \Theta_{2,1}(y_2) \\
\Theta_{1,1}(y_1) \Theta_{2,2}(y_2) \\
\Theta_{1,3}(y_1) \Theta_{2,0}(y_2) \\
\Theta_{1,0}(y_1) \Theta_{2,3}(y_2) \\
\Theta_{1,1}(y_1) \Theta_{2,3}(y_2) \\
\Theta_{1,2}(y_1) \Theta_{2,3}(y_2) \\
\Theta_{1,3}(y_1) \Theta_{2,3}(y_2) \\
\Theta_{1,2}(y_1) \Theta_{2,2}(y_2) \\
\Theta_{1,3}(y_1) \Theta_{2,1}(y_2) \\
\Theta_{1,3}(y_1) \Theta_{2,2}(y_2) \\
\Theta_{1,3}(y_1) \Theta_{2,2}(y_2)
\end{array} \right\}
\]

– note the 6 additional basis functions compared to the total degree case
The difference in the number of degrees of freedom $N$ for the two cases explodes as the number of random parameters increases.

- Increasing the maximal degree of the polynomials also causes explosive growth, especially for larger parameter dimensions.

<table>
<thead>
<tr>
<th>$N$ = no. random parameters</th>
<th>$M$ = maximal degree of polynomials</th>
<th>$K$ = no. of parameter degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>using total degree basis</td>
<td>using tensor product basis</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>216</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>1,024</td>
</tr>
<tr>
<td></td>
<td>252</td>
<td>7,776</td>
</tr>
<tr>
<td>10</td>
<td>286</td>
<td>1,048,576</td>
</tr>
<tr>
<td></td>
<td>3,003</td>
<td>60,046,176</td>
</tr>
<tr>
<td>20</td>
<td>1,771</td>
<td>$&gt; 1 \times 10^{12}$</td>
</tr>
<tr>
<td></td>
<td>53,130</td>
<td>$&gt; 3 \times 10^{15}$</td>
</tr>
<tr>
<td>100</td>
<td>176,851</td>
<td>$&gt; 1 \times 10^{60}$</td>
</tr>
<tr>
<td></td>
<td>96,560,646</td>
<td>$&gt; 6 \times 10^{77}$</td>
</tr>
</tbody>
</table>
• Recall that the discrete stochastic Galerkin system one has to solve involves $JK$ equations and unknowns

- figures for $K$ in the table have to be multiplied by the number of finite element degrees of freedom $J$

  thus, using tensor product bases is a really bad idea, except if one has very few parameters and is willing to use low-degree polynomials

• But, note that even for total degree global polynomial approximation, there is explosive growth in the number of parameter degrees of freedom as $N$ or $M$ increases

  $\iff$ the curse of dimensionality
Global orthogonal polynomial approximation

- For \( n = 1, \ldots, N \), let \( \{H_{n,m_n}(y_n)\}_{m_n=0}^{M} \) denote the set of polynomials in \( \mathbb{R} \) of degree less than or equal to \( M \) that are orthonormal with respect to the function \( \rho_n(y_n) \), i.e., we have that

\[
\int_{I_n} H_{n,m_n}(y_n) H_{n,m'_n}(y_n) \rho_n(y_n) \, dy_n = \delta_{m_m'\prime} \quad \text{for } m_n, m'_n \in \{0, \ldots, M\}
\]

- note that the set \( \{H_{n,m_n}(y_n)\}_{m_n=0}^{M} \) is hierarchical in the sense that \( \text{degree}(H_{n,m_n}) = m_n \)

- The multivariate basis functions \( \Psi_k(y) \)'s are products of one-dimensional orthonormal polynomials and have total degree less than or equal to \( M \)

\[
\Psi_k(y) = \prod_{n=1}^{N} H_{n,m_n}(y_n) \quad \text{for all } m_n \in \{0, \ldots, M\} \text{ such that } \sum_{n=1}^{N} m_n \leq M
\]

- we then have that \( k \in \{1, \ldots, K_{PC} = \frac{(N+M)!}{N!M!}\} \)
— for example, if $M = 1$ and $M = 3$ we have the $K_{PC} = 4$ basis functions$^{20}$

$$H_{1,0}(y_1)H_{2,0}(y_2)H_{3,0}(y_3)$$

$$H_{1,1}(y_1)H_{2,0}(y_2)H_{3,0}(y_3)$$

$$H_{1,0}(y_1)H_{2,1}(y_2)H_{3,0}(y_3)$$

$$H_{1,0}(y_1)H_{2,0}(y_2)H_{3,1}(y_3)$$

whereas for if $M = 2$ and $N = 3$ we have the $K_{PC} = 10$ basis functions (suppressing noting the explicit dependences on the $y_n$’s)

$$H_{1,0}H_{2,0}H_{3,0}$$

$$H_{1,1}H_{2,0}H_{3,0}$$

$$H_{1,0}H_{2,1}H_{3,0}$$

$$H_{1,0}H_{2,0}H_{3,1}$$

$$H_{1,0}H_{2,2}H_{3,0}$$

$$H_{1,1}H_{2,1}H_{3,1}$$

$$H_{1,1}H_{2,2}H_{3,1}$$

$$H_{1,0}H_{2,0}H_{3,2}$$

$^{20}$It is convenient and instructive to write the $N$-dimensional polynomials so that each row contains the polynomials of the same total degree $\sum_{n=1}^{N} m_n$; thus the first row contains all possible products of the $N$ one-dimensional polynomials of total degree 0, the second row has total degree 1, etc.
• We have that

\[
\int_{\Gamma} \Psi_k(y) \Psi'_k(y) \rho(y) \, dy = \int_{\Gamma} \Psi_k(y) \Psi'_k(y) \prod_{n=1}^{N} \rho_n(y_n) \, dy = \prod_{n=1}^{N} \int_{\mathcal{I}_n} H_{n,m_n}(y_n) H_{n,m'_n}(y_n) \rho_n(y_n) \, dy_n = \delta_{kk'}
\]

so that the multivariate polynomials \( \{ \Psi_k(y) \}_{k=1}^{K} \) are also orthonormal

– note that to obtain this results we need to assume that \( \rho(y) = \prod_{n=1}^{N} \rho_n(y_n) \), i.e., it is a product of one-dimensional PDFs, so that we know what \( H_{n,m}(\cdot) \) is orthonormal with respect to

– thus, we are restricted to independent random variables and to parameter domains \( \Gamma \) that are (possibly infinite) hyper-rectangles

• It is easy to see that the set \( \{ \Psi_k \}_{k=1}^{K_{PC}} \) of \( N \)-dimensional polynomials is a basis for the complete polynomial space of degree \( M \)

– i.e.,

\[
\text{span}\{\Psi_k\}_{k=1}^{K_{PC}} = \text{all } N\text{-dimensional polynomials of total degree } \leq M
\]
• The **stochastic Galerkin-global orthogonal polynomial approximation** of the solution of the SPDE is then defined by setting

\[ Z_{PC} = \text{span}\{\Psi_k\}_{k=1}^{K_{PC}} \]

so that

\[ u_{PC}(x, y) = \sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \phi_j(x) \Psi_k(y). \]

• Stochastic Galerkin-global orthogonal polynomial approximation with respect to parameters is better known, in the computational random PDE community, under the name polynomial chaos\textsuperscript{21,22}

\footnote{\textsuperscript{21}“Polynomial chaos” is a term coined by Weiner to describe representations of Gaussian white noise random fields in term of Hermite polynomials. It is unfortunate that this term was adopted to describe expansions of colored noise fields as well. A more elucidating terminology for describing such approximations is simply “global orthogonal approximations.” However, because the “polynomial chaos” has come into general use, we do use it in this work.

\textsuperscript{22}Paying some homage to Weiner, in the case of colored noise fields, polynomial chaos approximations usually refer to the case for which, for all \( n \), \( \rho_n(y_n) \) is a Gaussian PDF so that, for all \( n \), \( \{H_{n,m}(y_n)\}_{m=0}^{M} \) are sets of Hermite polynomials; for other PDFs, the polynomial chaos approximation is usually referred to as a “generalized polynomial chaos approximation;” here we do not differentiate between the two and refer to all cases as polynomial chaos approximations.}
Polynomial chaos approximations of quantities of interest

- The stochastic Galerkin-polynomial chaos approximation of a QoI is defined by

$$Q_oI = \int_\Gamma G(u(x; y)) \rho(y) dy \approx \sum_{q=1}^{Q} w_q \rho(y_q) G(u_{PC}(x; y_q)) = QoI_{PC}$$

where $u_{PC}(x; y_q)$, $q = 1, \ldots, Q$, is obtained by evaluation

- i.e., we have $u_{PC}(x, y_q) = \sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \phi_j(x) \Psi_k(y_q)$ for $q = 1, \ldots, Q$

- Thus, the polynomial chaos approximation of a QoI can be determined by

  1. first solving a single $JK_{PC} \times JK_{PC}$ system of equations to determine the coefficients $c_{jk}$, $j = 1, \ldots, J$ and $k = 1, \ldots, K_{PC}$
  2. then evaluating $\{y_q\}_{q=1}^{Q}$ for each the $Q$ quadrature points
  3. substituting the results into the approximation of the quantity of interest

- The cost of obtaining a stochastic Galerkin-polynomial chaos approximation of a quantity of interest is dominated by the first step
Economies in polynomial chaos methods for linear problems

• Suppose that the PDE with random inputs is linear in the solution $u$
  
  – for example, consider the case for which one has, using a polynomial chaos method, the discretization of the PDE given by\(^\text{23}\)

$$
\int_D \int_\Gamma \rho(y) a(x; y) S \left( \sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \phi_j(x) \Psi_k(y) \right) T \left( \phi_{j'}(x) \right) \Psi_{k'}(y) \, dx \, dy
$$

$$
= \int_D \int_\Gamma \rho(y) \phi_{j'}(x) \Psi_{k'}(y) f(x; y) \, dx \, dy
$$

where now both $S(\cdot)$ and $T(\cdot)$ are linear operators

\(^{23}\)Here, it is useful to follow the explicit dependences of the data functions $a$ and $f$ on the spatial variable $x$
because $S(\cdot)$ is now linear and does not involve derivatives with respect to the components of $y$, we have that

$$\sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \int_{D} S(\phi_j(x)) T(\phi_j'(x)) \int_{\Gamma} a(x; y) \rho(y) \Psi_k(y) \Psi_k'(y) \, dy \, dx$$

$$= \int_{D} \phi_j'(x) \int_{\Gamma} f(x; y) \rho(y) \Psi_k'(y) \, dy \, dx$$

- In this linear PDE case, there are two economies possible in the implementation of polynomial chaos methods.
• Polynomial chaos expansions of correlated random input fields

– we approximate the input correlated random fields $a$ and $f$ in the same way one approximates the solution, i.e., using polynomial chaos expansions

– thus, we assume we have in hand the approximations

$$a(x; y) \approx \sum_{k=1}^{K_{PC}} a_k(x) \Psi_k(y) \quad \text{and} \quad f(x; y) \approx \sum_{k=1}^{K_{PC}} f_k(x) \Psi_k(y)$$

– if the random fields are given explicitly as functions of $y$, these expansions can be determined via projection onto the span of $\{\Psi_k(y)\}_{k=1}^{K_{PC}}$

– however, in most cases, correlated random fields are characterized only through their mean and covariance functions so that determining their PC approximations cannot be determined by standard approximation techniques, e.g., interpolation or least-squares projection, in the the span of $\{\Psi_k(y)\}_{k=1}^{K_{PC}}$
– for this purpose, one could first obtain the KL approximations of $a$ and $f$; because these are explicit formulas for the approximate random fields in terms of $y$, we can proceed to obtain the polynomial expansions by standard techniques

– alternately, and more consistent, one can determine PC expansions of the given mean and correlation functions $\mu(x)$ and $\text{Cov}(x, x')$, respectively; then, the coefficients in the PC expansions of $a$ and/or $f$ would be determined by matching the mean and covariance functions of those approximations to the given mean and covariance functions

– supposing that we have obtained the PC approximations of $f$ and $a$, we substitute them into discretized PC system to obtain

$$
\sum_{k''=1}^{K_{PC}} \sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \left( \int_{D} a_{k''}(x)S(\phi_{j}(x))T(\phi_{j'}(x)) \, dx \right) \left( \int_{\Gamma} \rho(y)\Psi_{k}(y)\Psi_{k'}(y)\Psi_{k''}(y) \, dy \right)
$$

$$
= \sum_{k''=1}^{K_{PC}} \left( \int_{D} f_{k''}(x)\phi_{j'}(x) \, dx \right) \left( \int_{\Gamma} \rho(y)\Psi_{k'}(y)\Psi_{k''}(y) \, dy \right) = \int_{D} f_{k}(x)\phi_{j'}(x) \, dx
$$
orthogonality results in some sparsity in the left-hand side that may be taken advantage of when using iterative linear system solution methods. For example, whenever $k + k' \neq k''$ (and for similar situations involving permutations of indices), the summand on the left-hand side vanishes.
determining the polynomial chaos approximations of the input functions $a$ and $f$ may be costly because one has to determine a different expansion for every spatial quadrature point used in the finite element spatial discretization

- of course, if the data is independent of $x$, then only one expansion for each input function is needed

we again point out that, in the global orthogonal approximation setting, the economies resulting from the use of polynomial chaos expansions of input functions are realizable only for linear SPDEs
• Karhunen-Loève expansions of correlated random input fields

supposing that the input functions $a$ and $f$ are correlated random fields, we may determine the approximate KL-expansions

$$a(x; y) \approx a_{min}(x) + \sum_{n=1}^{N} \sqrt{\lambda_n} a_n(x) y_n$$

and

$$f(x; y) \approx \sum_{n=1}^{N} \sqrt{\theta_n} f_n(x) y_n$$

where $\{\lambda_n, a_n(x)\}_{n=1}^{\infty}$ and $\{\theta_n, f_n(x)\}_{n=1}^{\infty}$ are the eigenpairs of the covariance functions for $a$ and $f$, respectively

- substituting into the discrete PC system results in

$$\sum_{j=1}^{J} \sum_{k=1}^{K_{PC}} c_{jk} \sum_{n=1}^{N} \sqrt{\lambda_n} \left( \int_{D} a_n(x) S(\phi_j(x)) T(\phi_{j'}(x)) \, dx \right) \left( \int_{\Gamma} y_n \rho(y) \Psi_k(y) \Psi_{k'}(y) \, dy \right)$$

$$= \sum_{n=1}^{N} \sqrt{\theta_n} \left( \int_{D} f_n(x) \phi_{j'}(x) \, dx \right) \left( \int_{\Gamma} y_n \rho(y) \Psi_{k'}(y) \, dy \right)$$
– doubly orthogonal polynomials can be constructed\textsuperscript{24} such that
\[
\int_{\Gamma} \Psi_k(y) \Psi_{k'}(y) \rho(y) \, dy = 0 \quad \text{and} \quad \int_{\Gamma} y \Psi_k(y) \Psi_{k'}(y) \rho(y) \, dy = 0
\]
whenever \( k \neq k' \)

– as a result, the parameter and spatial degrees of freedom uncouple so that one can solve for the \( c_{jk} \)’s by solving \( K_{PC} \) deterministic finite element problems of size \( J \) instead of the single problem of size \( JK_{PC} \)

– we again point out that, in the PC setting, the economies resulting from the use of KL-expansions of the data random fields are realizable only for linear PDEs

– furthermore, the KL expansion has to be done for the coefficient \( a \) and not for an exponent in a log-normal version of \( a \); thus, \( a \) cannot be Gaussian which raises the issue of whether or not the random variable are independent; recall that that is a requirement for PC methods

\textsuperscript{24}The construction involves solving an eigenvalue problem for each polynomial
General stochastic sampling methods for approximating QoIs

- Recall that QoIs often require the evaluation of parameter integrals of functions of the solutions
  - these integrals often have to be approximated using quadrature rules
  - to use such a rule, one needs the solution $u(x, y)$ of the SPDE at each of the quadrature points $y_q$, $q = 1, \ldots, Q$, in the parameter domain $\Gamma$
  - for this purpose, one can use a stochastic Galerkin method, e.g., a polynomial chaos method, to obtain an approximation to the solution $u_{PC}(x, y)$ and then evaluate that approximation at the quadrature points

- However, once a quadrature rule is chosen to approximate a quantity of interest, in particular, once the quadrature points $\{y_q\}_{q=1}^Q$ are known, the simplest and most direct means of determining $u(x, y_q)$ is to simply solve the PDE $Q$ times, once for each quadrature point $y_q$
  - this approach is referred to as the stochastic sampling method (SSM) for PDEs with random inputs or for QoIs that depend on those solutions
In an SSM, to determine an approximation to a QoI, one proceeds as follows:

- choose a quadrature rule for the probabilistic integral in the QoI
  i.e., one chooses quadrature weights and points \( \{w_q, y_q\}_{q=1}^Q \)

- choose a finite element method, (i.e., a finite element space and a basis \( \{\phi_j\}_{j=1}^J \) for that space) and, for each \( q \), one defines the finite element approximation of the solution of the PDE at the parameter space quadrature point \( y_q \) by

\[
  u^h_q(x) = \sum_{j=1}^J b_{jq} \phi_j(x) \quad \text{for } q = 1, \ldots, Q
\]

- determine \( b_{jq} \) for \( j = 1, \ldots, J \) and \( q = 1, \ldots, Q \), by separately, and if desired, in parallel, solving the \( Q \) deterministic problems of size \( J \times J \), i.e., for \( q = 1, \ldots, Q \), one solves

\[
  \int_D S \left( \sum_{j=1}^J b_{jq} \phi_j, y_q \right) T(\phi_{j'}) \, dx = \int_D \phi_{j'} f(y_q) \, dx \quad \text{for } j' = 1, \ldots, J
\]
– substitute \( u_q^h(x) \) into the quadrature rule

\[ \text{QoI}_{SSM} = \sum_{q=1}^{Q} w_q G(u_q^h(x); y_q) \rho(y_q) \approx \int_{\Gamma} G(u(x, y); y) \rho(y) \, dy = \text{QoI} \]

or

\[ \text{QoI}_{SSM} = \sum_{q=1}^{Q} w_q G(u_q^h(x); y_q) \approx \int_{\Gamma} G(u(x, y); y) \rho(y) \, dy = \text{QoI} \]

depending on how the sample points \( y_q \) are chosen

\[ \text{i.e., either uniformly or according to the PDF } \rho(y), \text{ respectively} \]

• Each \( u_q^h(x) \) can be determined using a legacy code as black box

\[ \text{i.e., without changing a single line of code} \]

\[ \text{one just uses the legacy code } Q \text{ times} \]
The cost of determining an approximation to a quantity of interest using the SSM approach is dominated by the cost to determine the $Q$ finite element solutions, each of size $J$

- this should be compared to the cost of using general SGM approaches for the same purpose that are dominated by the cost needed to determine the solution of a single system of size $JK$

- which approach wins

  - i.e., which one yields a desired accuracy in the statistics of quantities of interest for the lowest computational cost

depends on

- the value of $N$ (the number of parameters)
- the specific quadrature rule used in the SSM method
- the particular SGM method used
- the value of $Q$ (the number of quadrature points in SSM approaches)
- the value of $K$ (the number of probabilistic terms in the SGM approximation to the solution)
- the cost of solving the systems of discrete equations encountered and many implementation issues
– in most cases, and certainly due to some recent developments, SSMs win over SGMs which is why global orthogonal polynomial approximation methods are now often used in an SSM framework

• We only consider the case for which the parameter domain $\Gamma$ is a hyper-rectangle in $\mathbb{R}^N$

– by translating and scaling, we need only consider hypercubes which may be finite or infinite in extent

• One is tempted to use well-known quadrature rules to define the sample points for SSMs but, if one is experienced in using quadrature rules in 1, or 2, or 3 dimensions, one will find that many popular quadrature rules in those settings become unviable for even moderate values of $N$
• We will discuss two classes of quadrature rules for the $N$-dimensional hyper-cube
  – sampling + simple averaging rules; the canonical example is Monte Carlo integration
  – weighted quadrature rules based on standard one-dimensional rules

• Ultimately, we consider stochastic collocation methods based on sparse grid quadrature rules

• It is important to always keep in mind that in the SSM framework, the quadrature points are also the points used to sample solutions of the SPDE so that one seeks quadrature rules that achieve a desired accuracy using a few sample points as possible
Monte Carlo methods and other sampling + simple averaging quadrature rules

- Let $u_h(x, y)$ denote a spatial approximation of the exact solution $u(x, y)$ of an elliptic PDE with random input parameters $y \in \Gamma \subseteq \mathbb{R}^N$
  - for the sake of concreteness, we consider the case for which the random parameters $y$ appear in the coefficient $a(x, y)$ of the PDE

- Let $G(u(x, y))$ denote an integrand determined from an output of interest depending on the solution of the PDE and the desired statistical information about that output
  - for simplicity, we assume that $G = G(y)$
    e.g., we have integrated out the spatial dependence or have evaluated at a spatial point
• The parameter integral

\[
Q_{\text{ol}} = \int_{\Gamma} G(u(x, y)) \rho(y) \, dy = \mathbb{E}[G]
\]

denotes the quantity of interest

• We let \( G_h(y) = G(u_h(x, y)) \) denote the approximate integrand output of interest determined from the approximate solution \( u_h \) of the PDE for any \( y \in \Gamma \)

  – the corresponding approximation of the quantity of interest is given by

\[
Q_{\text{ol}}(h) = \int_{\Gamma} G(u_h(x, y)) \rho(y) \, dy = \mathbb{E}[G_h] \approx \mathbb{E}[G] = Q_{\text{ol}}
\]
The Monte Carlo method

- The Monte Carlo (MC) method then proceeds as follows
  - select $Q$ random samples $\{y_q\}_{q=1}^Q$ of the parameter vector; we do this by selecting i.i.d. samples for the components of each $y_q$
  - determine the $Q$ corresponding approximate solutions $u_h(x, y_q)$, $q = 1, \ldots, Q$
  - determine the $Q$ corresponding approximate integrands $G_{h,q} = G_h(y_q) = G(u_h(x, y_q))$, $q = 1, \ldots, Q$
  - then, the Monte Carlo approximation of the QoI is given by

$$QoI_{MC} = \frac{1}{Q} \sum_{i=1}^Q G_{h,q} \approx QoI_h = E[G_h] \approx QoI = E[G]$$

- There are two sources of error in the MC approximation of the QoI:
  
  \[ QoI \Rightarrow QoI_h \text{ (spatial approximation)} \]

  \[ QoI_h \Rightarrow QoI_{MC} \text{ (parameter approximation)} \]
• The error in the MC approximation of a QoI is estimated by
\[ |QoI_{MC} - QoI| \leq |QoI_h - QoI| + |QoI_{MC} - QoI_h|. \]
\ - spatial error 
\ - probabilistic error

we assume that

\ - the spatial error is of \( O(h^\alpha) \)
\ - the probabilistic error is of \( O(1/\sqrt{Q}) \)
\ - the cost of a single PDE solve is of \( O(h^{-\gamma}) \)

then, the cost incurred by the MC method to achieve \( O(\varepsilon) \) accuracy is given by

\[ C_{h,Q}(\varepsilon) = O(Qh^{-\gamma}) = O(\varepsilon^{-2 - \frac{\gamma}{\alpha}}) \]

• Example in 3D

<table>
<thead>
<tr>
<th>(\varepsilon)</th>
<th>(\gamma)</th>
<th>(\alpha)</th>
<th>(Q)</th>
<th>(C_{h,Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>3</td>
<td>1</td>
<td>(O(10^4))</td>
<td>(O(10^{10}))</td>
</tr>
<tr>
<td>0.01</td>
<td>6</td>
<td>1</td>
<td>(O(10^4))</td>
<td>(O(10^{16}))</td>
</tr>
<tr>
<td>0.01</td>
<td>3</td>
<td>2</td>
<td>(O(10^4))</td>
<td>(O(10^7))</td>
</tr>
<tr>
<td>0.001</td>
<td>3</td>
<td>1</td>
<td>(O(10^6))</td>
<td>(O(10^{15}))</td>
</tr>
</tbody>
</table>
• MC methods have a very great virtue (in addition to its simplicity)
  their convergence behavior is independent of the dimension $N$
  i.e., of the number of parameters

• They also have a great fault
  their convergence behavior is slow
  i.e. the error decays, in expectation, as $1/\sqrt{Q}$
  where $Q$ denotes the number of sample points.

• Another great strength of MC methods are that
  they are almost universally applicable
  e.g., they hardly differentiate between integrands that are smooth
  and integrands that are discontinuous

• But this is also one of their great weaknesses
  they do not take advantage of any smoothness in the integrand
  i.e., that it has some number of continuous derivatives
  with respect to the parameters $y_1, \ldots, y_N$
- The table illustrates both MC strengths and weaknesses
  - the slow (and not monotone) convergence of MC methods is evident
  - to obtain one more digit of accuracy than that of the last entry of the table, one would expect to need 100 times as many points, i.e., \( \approx 3 \) billion evaluations of the integrand
  - on the other hand, convergence for the approximate integration of a discontinuous integrand is also evident

<table>
<thead>
<tr>
<th>( Q )</th>
<th>MC estimate</th>
<th>MC error</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.000000</td>
<td>5.16771</td>
</tr>
<tr>
<td>1,024</td>
<td>6.000000</td>
<td>0.83228</td>
</tr>
<tr>
<td>32,768</td>
<td>4.81250</td>
<td>0.35521</td>
</tr>
<tr>
<td>1,048,576</td>
<td>5.39063</td>
<td>0.22291</td>
</tr>
<tr>
<td>33,554,432</td>
<td>5.18042</td>
<td>0.01271</td>
</tr>
<tr>
<td>exact</td>
<td>5.16771</td>
<td>–</td>
</tr>
</tbody>
</table>

Monte Carlo approximations of an integral of a discontinuous function
• MC’s strengths, especially its simplicity, account for its predominant use in practice

• Its weaknesses account for the huge effort that has been and continues to be expended in the development of “improvements”
Other sampling + simple averaging quadrature rules

• We consider sampling + simple averaging-based quadrature rules that are based on determining a set of quadrature points \( \{y_q\}_{q=1}^{Q} \) and then approximating integrals of a function \( G(y) \) by an equal weight rule

\[
\int_{\Gamma} G(y) \rho(y) \, dy \approx \frac{1}{Q} \sum_{q=1}^{Q} G(y_q)
\]

if one samples the points according to the PDF \( \rho(y) \)

or by

\[
\int_{\Gamma} G(y) \rho(y) \, dy \approx \frac{1}{Q} \sum_{q=1}^{Q} \rho(y_q) G(y_q)
\]

if one samples the points uniformly.

• The second approach seems simpler, but is wasteful because the density of points is the same in regions where \( \rho(\cdot) \) is small as where it is large

  – unfortunately, some sampling methods can only be used to sample uniformly or have difficulty, i.e., they are much less efficient, when sampling nonuniformly
– note that the weights do not depend on the position of the points \( \{y_q\}_{q=1}^Q \)
or on other geometric quantities, e.g., the volumes associated with the sample points

• There have been many sampling + simple averaging-based quadrature rules proposed as replacements for MC quadrature, including
  
  many variance reduction MC methods
  quasi-Monte Carlo methods (Halton, Sobol, Faure, Hammersley, \ldots)
  stratified sampling
  Latin hypercube sampling and its many “improved” versions
  orthogonal arrays
  lattice rules
  importance sampling
  etc.

– in general, these “improved” rules have, in theory, improved rates of convergence, at least for not too large \( N \)
– for example, for quasi-MC methods, the best theoretical result for the error is that it is of $O((\ln Q)^N/Q)$
  - note the dependence on $N$
  - for large $N$, the $(\ln Q)^N$ term dominates
  - the curse of dimensionality is still with us

– however, if one is careful when using them, “improved” sampling + simple averaging methods can sometimes indeed improve on MC sampling, at least for moderate parameter dimension $N$

• Multilevel and multi-index MC methods are another sampling approach towards making MC methods more efficient

• Non-intrusive global orthogonal polynomial approximations are another sampling approach
Errors and costs incurred by stochastic sampling methods

- As for MC methods, we can determine an estimate for the costs associated with SSMs

- We make the same assumptions as we did for MC methods and add that the error due to sampling is of \( O(Q^{-\mu}) \)
  - for MC methods, \( \mu = \frac{1}{2} \)

- Then, the cost incurred by an SSM to guarantee an error of \( O(\varepsilon) \) satisfies

\[
C_{SSM} = O(\varepsilon^{-\frac{1}{\mu} - \frac{\gamma}{\alpha}})
\]

- thus, any SSM such that \( \mu > \frac{1}{2} \) “beats” MC
VI - STOCHASTIC COLLOCATION METHODS
What is stochastic collocation?

- Stochastic collocation methods (SCMs), in their most general sense, are synonymous with stochastic sampling methods
  - i.e., to approximate the parameter integral appearing in a quantity of interest
    - one chooses a $Q$-point quadrature rule for the parameter hypercube $\Gamma \in \mathbb{R}^N$
    - then one solves the PDE $Q$ times, once for each parameter vector defined by one of the quadrature points
    - then one uses those solutions to evaluate the summands in the rule

- However, most often, SCMs refer to SSMs for which the quadrature rule $\{w_q, y_q\}_{q=1}^Q$ has greater precision, at least for smooth integrands, than that of sampling + simple averaging rules
  - in particular, SCMs often refer to SSMs that use sparse grid quadrature rules
One is familiar with many quadrature rules in one dimension

- on the hypercube, one can easily define multidimensional integration rules as tensor products of one-dimensional rules
- the quadrature points are tensor products of the quadrature points of the one-dimensional rules
- the quadrature weights are products of the weights of the one-dimensional rules
- on the other hand, as we have already seen, tensor products suffer greatly from the curse of dimensionality
  - there is explosive growth in the number of quadrature points as the dimension $N$ of the parameter space increases
A tensor product set of quadrature points in two dimensions

- Tensor product rules integrate tensor products of one-dimensional polynomials exactly
• Just as is the case for interpolation and approximation, one can get the same rate of convergence using quadrature rules that integrate total degree polynomials exactly
  – we have seen that for the same degree polynomial and the same number of parameters, the number of points needed for total degree approximations is much smaller that that for tensor product approximations

• However, even in moderate dimensions, quadrature rules based on total degree polynomials are not easy to define
  – determining a good set of quadrature points and the corresponding quadrature weights is difficult
  – these difficulties partially motivated interest in orthogonal polynomial methods
• However, there is available an intermediate means of defining quadrature rules having, for the same nominal accuracy,
  - many fewer points than that for tensor product rules
  - but a somewhat greater number than that for total degree polynomial rules

• These rules are constructed through judicious sparsifications of tensor product rules
  – they are known as Smolyak or sparse grid quadrature rules
  – the table provides a comparison of the degrees of freedom needed for total degree, sparse grid, and tensor product interpolations that interpolate all $N$-variate polynomials of degree less than or equal to $M$ exactly
  – the situation for interpolatory quadrature is entirely similar, where now, of course, “degrees of freedom” translates to the number of solves of the discretized PDE

<table>
<thead>
<tr>
<th>degrees of freedom</th>
<th>total degree</th>
<th>sparse grid</th>
<th>tensor product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\frac{(N + M)!}{N!M!}$</td>
<td>$O(M(\ln M)^{N-1})$</td>
<td>$(M + 1)^N$</td>
</tr>
</tbody>
</table>
• We note that the cost of obtaining the SCM approximation of a quantity of interest
  being an SSM approximation
is dominated by the cost of obtaining solutions of \( K \) systems of size \( J \)
where \( K \) denotes the degrees of freedoms in the approximation
with respect to the parameters

• Sparse grid quadrature rules, also referred to as Smolyak quadrature, as mentioned before, are related to tensor product quadrature
  – thus, the construction of sparse grid quadrature rules are also based on one-dimensional quadrature rules
  – thus, we begin with a discussion of some of the one-dimensional quadrature rules used in sparse-grid construction
One-dimensional quadrature rules relevant to sparse grids

- We discuss some one-dimensional rules that are germane to our discussion about sparse grids
  - for simplicity, we only consider rules over bounded intervals
  - then, without loss of generality, we restrict attention to the interval $[-1, 1]$
  - general bounded intervals can then be handled through translations and dilations

- We first establish some nomenclature
  - the order of a one-dimensional quadrature rule is the number of quadrature points
  - if a quadrature rule integrates all polynomials of total degree less than or equal to, say $p$, it is said to have precision $p$
    - Monte Carlo quadrature has precision zero; only constant functions are integrated exactly
– in addition, the members of a family of quadrature rules are indexed by the non-negative integers

– we denote a family of one-dimensional quadrature rules indexed by the non-negative integer $i$ as $\{Q(i)\}$ for $i = 0, 1, 2, \ldots$

• The $Q$-point Clenshaw-Curtis (CC) rule

- a CC rule of order $Q$, features unequally spaced points determined by the extrema of the Chebyshev polynomial of degree $Q - 1$, including the $Q - 2$ interior stationary points and the two end-point extrema

One-dimensional Clenshaw-Curtis quadrature points
the $Q$ point CC rule can integrate all polynomials of degree $Q$ exactly - i.e., the one-dimensional CC rule of order $Q$ has precision $Q$

- unlike rules that use equally spaced quadrature points such as the Newton-Cotes rules, CC rules remain stable for large number of quadrature points

- **Gauss rules** also feature unequally spaced points

  - a $Q$ point Gauss quadrature rule has precision $2Q - 1$, nearly twice that of a $Q$ point CC rule

  - Gauss rules can handle finite intervals (e.g., Gauss-Legendre rules), semi-infinite integrals (e.g., Gauss-Laguerre rules), and infinite intervals (e.g., Gauss-Hermite rules)

  - standard CC rules only apply to finite intervals although variants have been developed for both semi-infinite and infinite intervals

- A comparison of the errors resulting from the application of the different types of rule for smooth integrands is given in the table
<table>
<thead>
<tr>
<th>Q</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MC</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>0.598</td>
</tr>
<tr>
<td>2</td>
<td>1.892</td>
</tr>
<tr>
<td>3</td>
<td>1.208</td>
</tr>
<tr>
<td>4</td>
<td>1.312</td>
</tr>
<tr>
<td>5</td>
<td>1.316</td>
</tr>
<tr>
<td>6</td>
<td>0.618</td>
</tr>
<tr>
<td>7</td>
<td>0.195</td>
</tr>
<tr>
<td>8</td>
<td>0.552</td>
</tr>
<tr>
<td>9</td>
<td>0.157</td>
</tr>
<tr>
<td>10</td>
<td>1.061</td>
</tr>
<tr>
<td>11</td>
<td>0.117</td>
</tr>
</tbody>
</table>

Comparison of Q point Monte Carlo, Newton-Cotes, Clenshaw-Curtis, and Gauss-Legendre quadrature rules for approximating the integral of a smooth univariate function.
• The table illustrates why Gauss rules are extremely popular for integration of smooth functions in one dimension and why tensor product of Gauss rules are likewise popular for integration over cubes in two and three dimension
  – if one could use tensor product rules in higher dimensions, one would want to use Gauss rules for this purpose, but, as we already observed several times, even in moderate dimensions and for moderate precision, tensor product rules cannot be contemplated

• We have also mentioned that one instead uses sparse grids which are judicious sparsifications of tensor product rules
  – to this end, one would also naturally believe that one should sparsify tensor product Gauss rules
  – it turns out that this is not the usual choice
  – the most common choice of one-dimensional quadrature rules used for building sparse grid rules is a nested subset of the CC family
• A **nested** family of quadrature rules is a sequence of rules of increasing precision, with the property that each rule includes all the quadrature points of the previous one.

• Sparse grid rules are built using a sum involving rules of increasing order, so that reusing the quadrature points in the rules is important for efficiency.\(^{25}\)

• Gauss rules are not nested: members of the family do not share any quadrature points with other members except for the center of the integration interval for the rules with an odd number of points.

• Subsets of the Clenshaw-Curtis family have nested quadrature points as illustrated in the figure.

\(^{25}\)Recall again that an approximation of a quantity of interest requires the solution of the PDE at each parameter quadrature point.
The first few members of the nested subset of the Clenshaw-Curtis family of quadrature points

- The table provides the order (the number of quadrature points) and precision of the nested Clenshaw-Curtis family

<table>
<thead>
<tr>
<th>index</th>
<th>$\ell$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>$Q$</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>33</td>
<td>65</td>
<td>129</td>
<td>257</td>
<td>513</td>
</tr>
<tr>
<td>1D precision</td>
<td>$p$</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>33</td>
<td>65</td>
<td>129</td>
<td>257</td>
<td>513</td>
</tr>
</tbody>
</table>

The order (number of points) and precision (degree polynomial integrated exactly) of the first few members of the nested family of one-dimensional Clenshaw-Curtis quadrature rules.
Construction of two-dimensional nested CC sparse grid quadrature rules

- Because of the ability to use visual explanations, we first illustrate the construction of sparse grids (Smolyak) quadrature rules by considering specific examples of CC sparse grid quadrature in two dimensions, i.e., for $N = 2$.

- The construction process involves the use of two-dimensional tensor product rules

  - thus, for a multi-index $\mathbf{l} = (\ell_1, \ell_2)$ with $\ell_1, \ell_2 \in \mathbb{N}_0$, we denote by $Q(\ell_1, \ell_2) = Q(\ell_1) \times Q(\ell_2)$ the tensor product of the $Q(\ell_1)$ rule in the $y_1$ direction and the $Q(\ell_2)$ rule in the $y_2$ direction

  - in this case, the $Q(\ell_1, \ell_2)$ has precision $p_i$ in with respect to $y_i$, $i = 1, 2$

  - we let $|\mathbf{l}| = \ell_1 + \ell_2$ denote the sum of the multi-indices
• We use the multi-index $\mathbf{p} = (p_1, p_2)$ to denote the precisions in the two directions
  
  – for example, for the one-dimensional nested CC quadrature rules, we have that $Q(3, 2)$ would have
    
    precision 9 with respect to $y_1$
    precision 5 with respect to $y_2$
    
    so that $\mathbf{p} = (9, 5)$

  – note also that $Q(3, 2)$ is a $9 \times 5 = 45$ point rule

• To illustrate the construction of sparse grids, we use both precision diagrams and point diagrams
• Precision diagrams illustrate the precision of a two-dimensional quadrature rule
  – for example, for the $Q(3, 2)$ rule constructed from the one-dimensional nested CC rules, we have the precision diagram

![Precision diagram](image)

The precision diagram for the $Q(2, 3)$ tensor product rule constructed from the nested family of Clenshaw-Curtis rules. This rule has precision 9 and 5 in the $y_1$ and $y_2$ directions.

– each square indicates a specific monomial $y_1^{\alpha_1}y_2^{\alpha_2}$ that is integrated exactly by the product rule

– for example, the square in the fifth column and third row corresponds to the monomial $y_1^4y_2^2$
• Similarly, we have point diagrams which indicate the point pattern of a rule
  – for $Q(3, 2)$ we have the point diagram is

![Point diagram](image)

The point diagrams for the $Q(2, 3)$ tensor product rule constructed from the nested family of Clenshaw-Curtis rules. This rule uses 9 and 5 points in the $y_1$ and $y_2$ directions, respectively.

– note that for the CC rules, the point diagram can be obtained by deleting the first row and column of the precision diagram

– the relation between precision and point diagrams is not always so simple; e.g., consider Gauss quadrature rules
We define the notion of the level \( L \in \mathbb{N}_0 \) of a quadrature rule as the highest index of the one-dimensional rules used in its construction\(^{26}\)

- in general, we denote a level \( L \) quadrature rule in \( N \) dimensions by \( \mathcal{S}(L, N) \)
- for example, the level \( L \) tensor product rule in \( N = 2 \) dimensions is given by \( \mathcal{S}_{TP}(L, 2) = \mathcal{Q}(L, L) = \mathcal{Q}(L) \times \mathcal{Q}(L) \)
  has precision \( p = (\ell_L, \ell_L) \) and \( (\ell_L + 1)^2 \) quadrature points

Once \( L \) is chosen, the goal is to construct a quadrature rule whose quadrature points consist of (a hopefully) sparse subset of the \( (\ell_L + 1)^2 \) quadrature points of the tensor product rule \( \mathcal{Q}(L, L) \)

- we note at the outset that we will not be able to achieve the same precision as that of the parent tensor product rule
- what we achieve is a great enough sparsification of the points of the latter rule so that the sparse grid rule will still be more efficient

\(^{26}\)One could have, and often wants to have, a different level for each parameter direction \( y_n \) but, for simplicity, here we assume the same level for all directions.
• The sparse grid quadrature rule construction process thus begins by selecting a level $L \in \mathbb{N}_0$
  
  – it helps if at this point we focus on a specific level, say $L = 4$

• Naturally, we would want the one-dimensional monomials that are integrated exactly by the one-dimensional CC rule with index $\ell_4$ to still be integrated exactly by the two-dimensional rule
  
  – to this end, we start with the sum $Q(4, 0) + Q(0, 4)$ as depicted in the precision diagram provided in plot (a)

  – to this sum we cannot add the rules $Q(4, 1)$ and $Q(1, 4)$ because that would merely lead us on a path towards the full tensor product rule $Q(4, 4)$ which, of course, we want to avoid

  – so, we do the next best thing, which is to add the $Q(3, 1)$ and $Q(1, 3)$ rules to obtain the sum

     $$Q(4, 0) + Q(3, 1) + Q(1, 3) + Q(0, 4)$$

     depicted in plot (b)
(a), (b), and (c): precision diagrams for the different stages of the construction of the level $L = 4$ nested CC sparse grid.

- again, we do not add the $Q(3, 2)$ and $Q(2, 3)$ rules because this a step towards the tensor product rule $Q(3, 3)$
- instead, we add the $Q(2, 2)$ rule to obtain the sum

$$Q(4, 0) + Q(3, 1) + Q(2, 2) + Q(1, 3) + Q(0, 4)$$

as depicted in plot (c)
– at this point we are done except that we notice that these rules overlap

– for example, the square in the first row and column in plot (c) is present in all five rules

– the numbers within the squares in plot (d) indicate how many times that square has appeared in the five rules the current state of the rule

Overlap count of the quadrature rules in $\mathcal{Q}(4, 0) + \mathcal{Q}(3, 1) + \mathcal{Q}(2, 2) + \mathcal{Q}(1, 3) + \mathcal{Q}(0, 4)$
– it is easy to see that by subtracting the rules $Q(3, 0)$, $Q(2, 1)$, $Q(1, 2)$, and $Q(0, 3)$ from the sum $Q(4, 0) + Q(3, 1) + Q(2, 2) + Q(1, 3) + Q(0, 4)$ we remove all the overlaps

– then, final sparse grid rule is given by

$$S_{CC}(4, 2) = + Q(4, 0) + Q(3, 1) + Q(2, 2) + Q(1, 3) + Q(0, 4)$$

$$- Q(3, 0) - Q(2, 1) - Q(1, 2) - Q(0, 3)$$

– the structure used to express the level 4 rule is useful in later discussions

– the pattern we want to notice is that to obtain the level $L = 4$ sparse grid we added all rules with index sum $|l| = \ell_1 + \ell_2 = 4 = L$

and

subtracted all rules with index sum $|l| = \ell_1 + \ell_2 = 3 = L - 1$
What is the precision of this rule, i.e., what is the highest degree of the total degree two-dimensional polynomial space that is integrated exactly by this rule?

- the answer is easy to deduce from plot (c):
  - it is given by largest right isosceles triangle anchored at the origin which contains squares all of which also belong to the set of squares included in the sparse grid rule
- that triangle is depicted in plot (c)
- thus, we see that the precision of the sparse grid rule is $p = 9$
  - which is less than the precision $p = 17$ of the parent $Q(4, 4)$ tensor product rule
To compare the efficiency gained, if any, by the sparse grid rule over tensor product rules, we have to do two things

- first, to be fair, we should compare to the tensor produce grid having the same precision as the sparse grid (and not its parent tensor product grid which has higher precision)

- thus, in our example, we want to compare to the tensor product rule having precision $p = 9$; that rule is depicted in plot (e)
• Second, we have to determine how many function evaluations, i.e., how many solves of the PDE one has to do, to evaluate the approximation
  – one has to do a function evaluation at each quadrature point of the nine quadrature rules appearing in the level \( L = 4 \) quadrature rule
  – from the order the rules, we see that
    \[
    \begin{align*}
    Q(4,0) & + Q(3,1) + Q(2,2) + Q(3,1) + Q(4,0) \\
    17 & = 17 \times 1 \quad 27 = 9 \times 3 \quad 25 = 5 \times 5 \quad 27 = 3 \times 9 \quad 17 = 1 \times 17 \\
    - Q(3,0) - Q(2,1) - Q(1,2) - Q(0,3) ,
    \end{align*}
    \]
    where the numbers below the braces is the number of quadrature points for the quadrature rule above the braces
  – however, because the CC quadrature rules are nested, we do not have to do 161 function evaluations.\(^27\)
  – the quadrature points for all the rules that are subtracted are already in the set of 113 quadrature points of the rules that are added

\(^{27}\)Thus, at least for the CC rules, we see the great advantage of using a nested family of one-dimensional rules to construct multi-dimensional quadrature rules.
moreover, the rules that are added have many points in common so that, because of nesting, the quadrature actually uses only 65 unique points.

- the point diagram for this CC rule is given by 65 light and dark grey squares in plot (f).

- on the other hand, the tensor product rule of the same precision $p = (9, 9)$ precision has 81 points, 16 more than the CC sparse grid.

- the point diagram for this rule is given by the medium and dark grey squares in plot (f).
• Further examples in two dimension are given by
  – the level $L = 3$ nested CC sparse grid quadrature rule
    \[
    S_{CC}(3, 2) = +Q(3, 0) + Q(2, 1) + Q(1, 2) + Q(0, 3)
    - Q(2, 0) - Q(1, 1) - Q(0, 2)
    \]
    which has 29 quadrature points and precision 7

  and by
  – the level $L = 5$ nested CC sparse grid quadrature rule
    \[
    S_{CC}(5, 2) = +Q(5, 0) + Q(4, 1) + Q(3, 2) + Q(2, 3) + Q(1, 4) + Q(0, 5)
    - Q(4, 0) - Q(3, 1) - Q(2, 2) - Q(1, 3) - Q(0, 4)
    \]
    which has 145 quadrature points and precision 11

  – the corresponding tensor product rules having the same precision have 49 and 121 points, respectively

  – precision and point diagrams are given in the following figure
Precision diagrams for the (a) level $L = 3$ and (d) level $L = 5$ nested CC rules in two dimensions; (b) and (e): precision diagrams for the tensor product rules with the same precision, respectively. (c) and (f): point diagrams for the level $L = 3$ and $L = 5$ nested CC rules, respectively.
• The level 5 CC sparse grid quadrature rule does worse
  it has more quadrature points and therefore requires more
  solutions of the PDE
  than the tensor product quadrature rule having the same precision
  – the situation gets worse for higher levels
  – indeed, for $N = 2$ the nested CC family sparse grid rules do worse than
    tensor product grids

• However, for larger values of $N$, the situation changes (dramatically) in favor
  of the CC rules
  – why is this?
To gain an inkling of why this is so, consider the construction of the level 4 nested CC sparse grid rules for $N = 12$

– as part of the construction, we will have to sum all the tensor product rules whose indices add up to 4
  
i.e., all tensor product rules $Q(\ell_1, \ell_2, \ldots, \ell_{12})$ for which
  
  $|\mathbf{l}| = \ell_1 + \ell_2 + \cdots + \ell_{12} = 4$

– this implies that, in the tensor product quadrature rules used to construct the sparse grid, at most 4 of the indices are nonzero

– this occurs, e.g., for the tensor product rule $Q(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)$

– the consequence of having all the zeros is that the quadrature points tend to pack near the boundaries of the hypercube $\Gamma$

– on the other hand, tensor product grids are distributed throughout the hypercube
so, for the same multidimensional precision, there is a trade-off between
the smaller number of points in the one-dimensional rule
that underlies the tensor product grid
and
the packing of the nested CC quadrature points along the
lower-dimensional boundary

for $N = 2$, the tensor product rule wins the competition, but for even
moderate values of $N$, the nested CC rule wins, with the winning margin
increasing rapidly as $N$ increasing
Slow-growth nested CC sparse grid quadrature rules

- Despite the fact for all but the smallest dimensions, the nested CC sparse grid quadrature rules are better than tensor product rules, the former are still wasteful
  - for example, consider the level $L = 4$ precision diagram given in the figure

  ![Diagram](image)

  - see that the $Q(4, 0)$ rule (all of the bottom two rows) does not add any precision over and above that of the $Q(3, 1)$ rule (the bottom 4 rows up to the 10th column)

  - Analogous observations can be made about the first two columns and the $Q(0, 4)$ rule
• How do we systematically remove the wasteful points?
  – first, we have the following result due to \(^\text{??????}^{28}\)
    if the one dimensional family of rules have precision \(2i + 1\),
    then corresponding sparse-grid rule will also have precision \(2i + 1\)
    where \(i\) is the index of the one-dimensional rules
  – but, as we have seen, the precision of the nested one-dimensional CC rules
    grows much faster that \(2i + 1\)
  – this means that we can reuse rules according to the following table which
    lists the order of the one-dimensional rules that satisfy two criteria:
    their precision is at least \(2i + 1\)
    and
    they are members of the nested family of CC rules

\(^{28}\)This result applies to any family of sparse-grid quadrature rules, regardless of the specific choice for the
parent one-dimensional family of rules.
The order (number of points) and precision (degree polynomial integrated exactly) of the first few members of the nested family of one-dimensional Clenshaw-Curtis rules that minimally meet the needed precision given in the last row.

- thus, for example, for $i = 3$ we use the order 9 rule having precision $9 > 2 \times 3 + 1$ because we cannot use the order 7 rule with precision exactly $7 = 2 \times 3 + 1$ because that rule is not a member of the nested family

- on the other hand, for $i = 4$ we do not use the precision 17 rule as indicated in the table of nested CC rules, but instead we still use the precision 9 rule because $9 = 2 \times 4 + 1$ is good enough according to the theoretical result
that rule is not good enough for $i = 5$, so that we have to move up to the next member of the nested CC family, which is the precision 17 rule

but that rule remains good enough for $i = 6, 7, \text{ and } 8$ so that the next change only need occur when we arrive at $i = 9$

These observations can be used to build CC sparse grids with considerably fewer points compared to the standard CC sparse grids
What about the quadrature weights?

- Consider applying the simplest nested CC sparse grid quadrature rule that is still interesting, i.e., the level $L = 1$ rule in $N = 2$ dimensions given by

$$Q(1) \otimes Q(0) + Q(0) \otimes Q(1) - Q(0) \otimes Q(0)$$

to approximate the integral

$$\int_{-1}^{1} \int_{-1}^{1} f(y_1, y_2) \, dy_1 dy_2$$

- here, we have $\Gamma = [-1, 1] \times [-1, 1]$ and $f(\vec{y}) = G(\vec{y}) \rho(\vec{y})$
• The nested CC one-dimensional rules we need are the 1 point rule (the midpoint rule) and the 3-point rule
  
  we then have the component tensor product rules
  
  \[ Q(1) \otimes Q(0) = \frac{1}{3} f(-1, 0) + \frac{4}{3} f(0, 0) + \frac{1}{3} f(1, 0) \]
  
  \[ Q(0) \otimes Q(1) = \frac{1}{3} f(0, -1) + \frac{4}{3} f(0, 0) + \frac{1}{3} f(0, 1) \]
  
  \[ Q(0) \otimes Q(0) = 2 f(0, 0) \]
  
• Substituting the component rules into the CC sparse-grid rule results in
  
  \[ \frac{1}{3} f(-1, 0) + \frac{1}{3} f(1, 0) + \frac{2}{3} f(0, 0) + \frac{1}{3} f(0, -1) + \frac{1}{3} f(0, 1) \]
  
  – this is simply a 5-point quadrature rule in two dimensions with quadrature points \((-1, 0), (1, 0), (0, 0), (0, -1), \text{ and } (0, 1)\)
  
  and corresponding quadrature weights \(1/3, 1/3, 2/3, 1/3, \text{ and } 1/3\)
  
  – note that we used \(f(0, 0)\) three times
• In higher dimensions or higher levels, this repeated use of the same function evaluation caused by nesting happens even more often
  – in this way, the weights of a CC sparse-grid quadrature rule is determined by combining the one-dimensional weights for duplicate function evaluation, into a single weight for the multidimensional rule
  – although writing down a general formula for the combined weights is not possible, this task is easily accomplished in a computer code

• Applying the rule to the function $f(y_1, y_2) = e^{-(y_1-0.3)^2-(y_2-0.4)^2}$ that has the exact integral $\approx 0.815937$, we find that the component rules result in the approximations

\[
Q(1) \otimes Q(0) \approx 0.88604 \\
Q(0) \otimes Q(1) \approx 0.88232 \\
Q(0) \otimes Q(0) \approx 0.95122
\]

and the CC sparse grid rule results in the approximation $0.88604 + 0.88232 - 0.95122 = 0.81714$

---

For this example, we have mapped everything from the interval $[-1, 1]$ to the interval $[0, 1]$. 
The definition of the $L$-level sparse grid quadrature rule in dimension $N$ is given by

$$S(L, N) = \sum_{\max\{0, L-N+1\} \leq |l| \leq L} (-1)^{L-|l|} \binom{N-1}{L-|l|} (Q(\ell_1) \otimes \cdots \otimes Q(\ell_N))$$

$l = (\ell_1, \ell_2, \cdots \ell_N)$ denotes the vector describing the individual product rules

$|l| = \sum_{n=1}^{N} \ell_n$

The product rules for which $|l| = L$ are multiplied by

$$-1^0 \binom{N-1}{0} = 1$$

the product rules for which $|l| = L-1$ are multiplied by

$$-1^1 \binom{N-1}{1} = -(N-1)$$

and so on
• Implementing Smolyak’s formula for given \( L \) and \( N \) requires
  
  – a loop on \(|l|\) from \( \max\{0, L - N + 1\} \) to \( L \)
  – computing the value \((-1)^{L-|l|}\)
  – computing the combinatorial coefficient \( \binom{M-1}{L-|l|} \)
  – generating every nonnegative integer \( N \)-vector \( l = (\ell_1, \ell_2, \cdots, \ell_N) \) whose entries sum to \(|l|\)
  – determining the distinct quadrature points among the quadrature points of all the component rules \( Q(\ell_1) \otimes \cdots \otimes Q(\ell_N) \)
  – determining the quadrature weights corresponding to each quadrature point

• Fortunately, several good implementations for the construction of sparse grids are available, e.g., look at
  
  https://people.sc.fsu.edu/~jburkardt/
The need for smoothness

- The product rules we have discussed, including the sparse-grid rules, putatively achieve greater accuracy as the polynomial precision increases.

- Actually achieving the increased accuracy requires the integrand be “sufficiently” smooth, i.e., to have a “sufficient” number of continuous derivatives.

- For an integrand that is not sufficiently smooth, e.g., for the absolute value function or any discontinuous function, the sparse-grid approach using global polynomial interpolatory quadrature rules result in generally very bad approximations.\textsuperscript{30}

\textsuperscript{30} The same comment holds for global orthogonal polynomial approximations, i.e., for polynomial chaos methods.
For example, in dimension $N = 6$, let $f(y)$ denote the characteristic function of the unit ball embedded in a unit cube

- the table shows the non-convergence of nested of CC sparse grid quadrature for this example

- for comparison purposes, that table also provides the convergence history of Monte Carlo approximate integration for the same function

<table>
<thead>
<tr>
<th>$Q$</th>
<th>SG estimate</th>
<th>SG error</th>
<th>MC estimate</th>
<th>MC error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.000</td>
<td>1.167</td>
<td>0.00000</td>
<td>5.16771</td>
</tr>
<tr>
<td>13</td>
<td>64.000</td>
<td>58.832</td>
<td>0.00000</td>
<td>5.16771</td>
</tr>
<tr>
<td>85</td>
<td>-42.667</td>
<td>-47.834</td>
<td>3.01176</td>
<td>2.15595</td>
</tr>
<tr>
<td>389</td>
<td>-118.519</td>
<td>-123.686</td>
<td>4.77121</td>
<td>0.39650</td>
</tr>
<tr>
<td>1457</td>
<td>148.250</td>
<td>143.082</td>
<td>5.16771</td>
<td>0.01555</td>
</tr>
<tr>
<td>4865</td>
<td>-24.682</td>
<td>-29.850</td>
<td>5.41994</td>
<td>0.25226</td>
</tr>
<tr>
<td>exact</td>
<td>5.16771</td>
<td>–</td>
<td>5.16771</td>
<td>–</td>
</tr>
</tbody>
</table>

Comparison of sparse grid and Monte Carlo approximations of the integral of a discontinuous function.
Complexity comparison of different sparse grid rules

- The table provides the number of distinct quadrature points for the first 11 levels of several sparse-grid quadrature rules in $N = 2, 6, \text{ and } 10$ dimensions.

- We also see that although sparse grids represent a very great savings over tensor product grids:
  
  e.g., for $N = 10$ and precision 21, the number of points for a tensor product quadrature rule based on one-dimensional Gauss rule is $25,937,424,601$.

  Sparse grids still suffer from the curse of dimensionality so that they are useful only for relatively moderate dimensions and relatively moderate precisions.

- Of course, the same could be said, perhaps more so, for stochastic Galerkin methods, i.e., polynomial chaos methods.

- As mentioned before, good software is readily available for determining the quadrature points and weights for all the sparse grids listed in the table, as well as some others.
The number of distinct quadrature points, e.g., the number of times the PDE has to be solved, for several sparse grids for the first eleven levels and for three values of $N$, the number or random parameters. CC=nested Clenshaw-Curtis; SGCC=slow growth Clenshaw-Curtis; GL=Gauss-Legendre; SGGL=slow growth Gauss-Legendre; SGGP=slow growth Gauss-Patterson.

<table>
<thead>
<tr>
<th>level</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
</tbody>
</table>

$N = 2$

<table>
<thead>
<tr>
<th>number of distinct quadrature points</th>
<th>CC</th>
<th>1</th>
<th>5</th>
<th>13</th>
<th>29</th>
<th>65</th>
<th>145</th>
<th>321</th>
<th>705</th>
<th>1537</th>
<th>3329</th>
<th>7169</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGCC</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>29</td>
<td>49</td>
<td>81</td>
<td>129</td>
<td>161</td>
<td>225</td>
<td>257</td>
<td>385</td>
<td></td>
</tr>
<tr>
<td>GL</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>29</td>
<td>53</td>
<td>89</td>
<td>137</td>
<td>201</td>
<td>281</td>
<td>381</td>
<td>501</td>
<td></td>
</tr>
<tr>
<td>SGGL</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>33</td>
<td>45</td>
<td>81</td>
<td>97</td>
<td>161</td>
<td>181</td>
<td>281</td>
<td></td>
</tr>
<tr>
<td>SGGP</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>33</td>
<td>33</td>
<td>65</td>
<td>97</td>
<td>97</td>
<td>161</td>
<td>161</td>
<td></td>
</tr>
</tbody>
</table>

$N = 6$

<table>
<thead>
<tr>
<th>number of distinct points</th>
<th>SGCC</th>
<th>1</th>
<th>13</th>
<th>85</th>
<th>389</th>
<th>1409</th>
<th>4289</th>
<th>11473</th>
<th>27697</th>
<th>61345</th>
<th>126401</th>
<th>244289</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGGL</td>
<td>1</td>
<td>13</td>
<td>73</td>
<td>257</td>
<td>737</td>
<td>1925</td>
<td>4509</td>
<td>9837</td>
<td>20445</td>
<td>40025</td>
<td>75917</td>
<td></td>
</tr>
<tr>
<td>SGGP</td>
<td>1</td>
<td>13</td>
<td>73</td>
<td>257</td>
<td>737</td>
<td>1889</td>
<td>4161</td>
<td>8481</td>
<td>16929</td>
<td>30689</td>
<td>53729</td>
<td></td>
</tr>
</tbody>
</table>

$N = 10$

<table>
<thead>
<tr>
<th>number of distinct points</th>
<th>SGCC</th>
<th>1</th>
<th>21</th>
<th>221</th>
<th>1581</th>
<th>8721</th>
<th>39665</th>
<th>155105</th>
<th>536705</th>
<th>1677665</th>
<th>4810625</th>
<th>12803073</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGGL</td>
<td>1</td>
<td>21</td>
<td>201</td>
<td>1201</td>
<td>5281</td>
<td>19165</td>
<td>61285</td>
<td>177525</td>
<td>474885</td>
<td>1192425</td>
<td>2835589</td>
<td></td>
</tr>
<tr>
<td>SGGP</td>
<td>1</td>
<td>21</td>
<td>201</td>
<td>1201</td>
<td>5281</td>
<td>19105</td>
<td>60225</td>
<td>169185</td>
<td>434145</td>
<td>1041185</td>
<td>2347809</td>
<td></td>
</tr>
</tbody>
</table>
VII - STOCHASTIC SURROGATES
Surrogates

● In our context, surrogate approximations are approximations to the solution of a PDE that are cheaper to obtain for any given $y \in \Gamma$ compared to solving the PDE for that value of $y$.

● Surrogate approximations are built by
  – choosing sample points $\{y_s\}_{s=1}^{N_s}$
  – determining the high-fidelity, high-cost spatial approximation, e.g., the FEM approximation, $u_h(x, y_s)$ of the solution of the PDE at each of the sample points $y_s$, $s = 1, \ldots, N_s$
  – use the samples $\{u_h(x, y_s)\}_{s=1}^{N_s}$ to build, with respect to the parameters $y$, a surrogate or response surface $u_{h,N_s}(x, y)$ for the spatial approximation $u_h(x, y)$ of the solution of the PDE
  – then, when one needs the solution of the PDE at a $y \notin \{y_s\}_{s=1}^{N_s}$, one evaluates the surrogate instead of solving the PDE for that value of $y$. 

• Interpolation, least-squares approximation, reduced-order models, orthogonal polynomial approximation are well known types of surrogates; there are others

• Surrogates are especially useful when one has to solve the PDE many times
  – optimization, inverse problems, inference, and parametric studies are examples
  – evaluating statistical integrals is another
    we consider this setting before we return to briefly discuss interpolatory surrogates

• Note that surrogates are built with respect to the parameters $y$

• We label the surrogates $u_{h,N_s}$ with the spatial grid size $h$ to highlight the fact that surrogates in parameter space are determined from spatial approximations of the solution of the PDE
Surrogates and statistical integrals

- If we want to evaluate a quantity of interest (QoI) determined by an integral over the parameter domain $\Gamma$
  - using a $Q$-point quadrature rule having quadrature points $\{y_q\}_{q=1}^Q \subset \Gamma$
  and quadrature weights $\{w_q\}_{q=1}^Q$

  it is natural

  - to choose the set of sample points $\{y_s\}_{s=1}^{N_s}$ that are used to solve the PDE $N_s$ times to be the same as the set of quadrature points $\{y_q\}_{q=1}^Q$
  that are used to approximate the quantity of interest

  - this is, in fact, what we did in previous lectures
• However, often one must choose $Q$ large in order to obtain an accurate approximation of the QoI, i.e., of the integral over the parameter domain $\Gamma$
  – this is certainly the case for Monte Carlo approximations of parameter integrals, even for moderate parameter dimension
  – this is also the case for all other quadrature rules including sparse-grid rules

• Because one obtains a solution of the PDE for each sample point we may not be able to afford choosing $N_s = Q$
  – in fact, we often want $N_s \ll Q$, i.e., relative very few solutions of the discretized PDE
  – however, to approximate the integral, we still need to have approximate solutions of the PDE for every one of the $Q$ quadrature points in the parameter domain
So, at any of the $Q \gg N_s$ quadrature points $y_q$, one evaluates the surrogate approximation instead of solving for the expensive FEM approximation the PDE, i.e.,

$$Q_{\text{ol}} \approx Q_{\text{ol}}_h = \sum_{q=1}^{Q} w_q G\left( u_h(x, y_q) \right) \rho(y_q)$$

is replaced by

$$Q_{\text{ol}} \approx Q_{\text{ol}}_{h, N_s} \approx \sum_{q=1}^{Q} w_q G\left( u_{h, N_s}(x, y_q) \right) \rho(y_q).$$

Thus, if the surrogate is an accurate enough approximation of the solution of the PDE and $Q \gg N_s$, then one can determine an approximation of the QoI much more cheaply using surrogates compared to not using them.
In the UQ setting, it is usually more efficient to directly build a surrogate \( F_{\text{surrogate}}(u_h(x, y); x, y) \) for the output of interest \( F(u(x, y); x, y) \) or even a surrogate \( G_{\text{surrogate}}(u_h(x, y); x, y) \) for the integrand \( G(u(x, y); x, y) \).

– Certainly, if the output of interest \( F(u) \) is a functional of the solution of the PDE that only depends on the parameters \( y \) (and not on the spatial variables \( x \)), it is markedly more efficient to build surrogates for \( F(y) \) or \( G(y) \) than for \( u(x, y) \) and then use the latter to evaluate approximations for \( F(y) \) or \( G(y) \).
Choosing the specific surrogate

- The recipes given so far are incomplete
  i.e., we use the phrase “from the samples . . . build a surrogate”
  but do not mention how this constructions is done
  - for this purpose, one can, in principle, use any method, e.g., interpolation,
    least-squares approximation, min-max approximation, reduced-order modeling, etc., from the rich literature about the approximation of functions
  - here, to provide a concrete illustration, we only consider the use of global polynomial interpolation for building surrogates for the approximate output of interest $F_h(x, y)$
    - building surrogates for the approximate solution of the PDE $u_h$ or for the approximate integrand $G_h$ would proceed in an entirely similar manner
• A set of points \( \{y_s\}_{s=1}^{N_s} \) is chosen in the parameter domain \( \Gamma \) which are used to obtain the \( N_s \) samples \( \{F_h(x, y_s)\}_{s=1}^{N_s} \) of the output of interest
  
  – of course, each of the samples of \( F_h(x, y_s) \) require the solution of the spatially discretized PDE

  – the sample points are also use to build the set of Lagrange fundamental polynomials \( \{\ell_s(y)\}_{s=1}^{N_s} \)

  i.e., polynomials having the property \( \ell_s(y_{s'}) = \delta_{ss'} \) for all \( s, s' = 1, \ldots, N_s \)

  – then, the Lagrange interpolation surrogate of the quantity of interest is given by

  \[
  F_{h,N_s}(x, y) = \sum_{s=1}^{N_s} F_h(x, y_s)\ell_s(y)
  \]

• Two issues still remain
• First, how does one choose the sample points \( \{y_s\}_{s=1}^{N_s} \)?

  – in particular, one would want sample points for which the Lebesgue constant corresponding to the points remains reasonably small with respect to increases in the parameter dimension \( N \) and the polynomial degree \( p \)

  – in one and two-dimensional hypercubes, several good choices are known
    - in higher dimensions, the situation is not anywhere as good

  – one could use tensor products of good one dimensional interpolation points
    - e.g., the Chebyshev points
    but, of course, as we well know by now, this is not a viable approach

  – fortunately, at least for moderate parameter dimensions, sparse grids can be used effectively for interpolation
• The second issue is how does one construct the Lagrange fundamental polynomials $\{\ell_s(y)\}_{s=1}^{N_s}$?

- in


an efficient algorithm is given for the construction of these polynomials for an arbitrary set of points

- the algorithm only uses the basic arithmetic operations, unlike other approaches that involve the expensive and sometimes unstable evaluation of Vandermode determinants

- for sparse grid, a systematic method is known for building interpolants that is very similar to the way sparse grid quadrature rules are built
Surrogates and discontinuous integrands

- There are UQ situations, however, in which building a surrogate for $G_h$ is not preferable to building one for $F_h$ or even $u^h$

  - the best example is the (approximate) event probability QoI, i.e.,

    $$Q_{oI} = \text{prob} \left[ F^h \geq F_0 \right] = \int_{\Gamma} \chi_{F^h \geq F_0} \rho(y) \, dy$$

    where $F^h = F(u_h(x; y))$ is an approximate output of interest that depends on the spatial approximation of the solution of the PDE and $F_0$ is a prescribed threshold value

  - in this case

    $$G_h(x, y) = G(F^h(x, y)) = \chi_{F^h \geq F_0}$$

    which is a discontinuous function of $y$ even when $F_h(x, y)$ is a smooth function of $y$
- in such cases, it is much better to build the surrogate for $F_h$ than for $G_h$
  because a much smaller $N_s$
    - i.e., fewer samples of the expensive FEM model
  is required to build an accurate approximation for $F_h$ because it is smooth, compared to the number of samples needed build an accurate approximation of $G_h$ which is not smooth
- if for some reason $F_h(x, y)$ is not a smooth function of $y$ but $u_h(x, y)$ is, then building a surrogate for $u^h$ would be preferable to building one for $F_h(x, y)$
Surrogates for $F_h(y)$ are very useful for approximating $\text{QoI}_h$ by a quadrature rule

- because of the discontinuous integrand, using quadrature rules, e.g., sparse grid rules, that require smoothness of the integrand are not effective, even compared to Monte Carlo (MC) rules
- for this reason, MC remains the method of choice for approximating such integrals
- thus, we have the computable approximation of the QoI given by

$$\text{QoI}^{MC}_h = \frac{1}{Q} \sum_{q=1}^{Q} \chi_{F_h(y_q) \geq F_0},$$

where $F_h(y_q) = F(u_h(x; y_q))$

- the slow convergence of MC methods means that this approximation requires $Q \gg 1$ solutions of the discretized PDE

\[^{31}\text{For simplicity, we again assume that the output of interest } F \text{ if only a function of } y.\]
if we instead use $N_s$ samples $y_s \in \Gamma$, $s = 1, \ldots, N_s$, to construct a surrogate $F_{h,N_s}(y)$ for a smooth (with respect to $y$) approximate output of interest $F_h(y)$, we can define the approximation of the QoI given by

$$Q\text{ol}_{MC}^{N_s} = \frac{1}{Q} \sum_{q=1}^{Q} \chi_{F_{h,N_s}(y_q) \geq F_0}.$$ 

in general, we would have $N_s \ll Q$ so that using the surrogate results in a huge savings over using FEM solutions of the PDE
VIII - MULTIFIDELITY APPROACHES TO UQ FOR PDES
SETTING
- Parameterized PDEs

  - inputs that depend on a finite number of random parameters \( \{y_n\}_{n=1}^N \)
    
    e.g., coefficients, data in boundary or initial conditions, etc.

  - parameters could appear

    - as physical parameters occurring in the problem description
      
      e.g., permeability, diffusion coefficient, stock price,
      
      reaction rate, applied voltage, . . .

    - in a truncation of a parameterized representation of a random field
      
      e.g., a Karhunen-Loève expansion

    - a combination of both types
parameters are collected in a random input vector

\[ \mathbf{y} = (y_1, \ldots, y_n)^T \in \Gamma \subset \mathbb{R}^N \]

- of course, the inputs could also depend on
  the spatial variable \( \mathbf{x} \in D \subset \mathbb{R}^d \)
  and temporal variable \( t \in [0, T] \)

\[ \Gamma = \text{parameter domain (ubiquitously a hyper-rectangle)} \]
\[ D = \text{spatial domain having boundary } \partial D \]
\[ [0, T] = \text{time interval} \]

- consider a problem of the form\(^1\)

\[ \mathcal{L}_y(u(\mathbf{x}; \mathbf{y})) = f_y \]

- the operator \( \mathcal{L}_y \) depends on the parameter vector \( \mathbf{y} \)
- the right-hand side \( f_y \) depends on the parameter vector \( \mathbf{y} \)

\[ \implies \text{the solution } u(\mathbf{x}; \mathbf{y}) \text{ depends on } \mathbf{y} \text{ (as well as on } \mathbf{x} \in D) \]

\(^1\)For simplicity, we drop mention of temporal dependences
Ubiquitous example

\[-\nabla (a(x; y) \cdot \nabla u(x; y)) = f(x; y) \quad \text{for } x \in D, y \in \Gamma\]

\[u(x; y) = 0 \quad \text{for } x \in \partial D, y \in \Gamma\]

- Probabilistic description of uncertainty

- each parameter \( y_n \in \Gamma_n \) is accompanied by a PDF \( \rho_n(y_n) \)

- assume the parameters are independent so that

\[\rho(y) = \prod_{n=1}^{N} \rho_n(y_n) \quad \text{and} \quad \Gamma = \prod_{n=1}^{N} \Gamma_n\]

Uncertainty quantification

given statistical information about the inputs of a system,
determine statistical information about the outputs of that system
• Outputs of interest are
  – not individual realizations of the solution $u(x; y)$
  – generally not the solutions themselves
  – often a functional $F(u)$ of the solution
    - spatial averages, spatial extrema, norms, …

• Quantities of interest (QoIs) are statistical information about outputs of interest
  - expected values, variances, standard deviations, covariances, higher moments, probabilities, PDFs, …
  – often involve statistical integrals, e.g.,

$$E(F) = \int_{\Gamma} F(y)\rho(y)dy \iff \text{expected value of } F$$
$$\nabla(F) = \int_{\Gamma} (F(y) - E[F])^2\rho(y)dy \iff \text{variance of } F$$
• We consider QoIs of this type

\[ QoI = \int_\Gamma G(y) \rho(y) dy \]

where \( G(y) \) incorporates the desired statistical information about the output of interest, e.g.

- for the expected value of \( F \) \( \Rightarrow \) \( G(y) = F(y) \)
- for the variance of \( F \) \( \Rightarrow \) \( G(y) = (F(y) - \mathbb{E}[F])^2 \)

• In general, the statistical integral over the parameter domain has to be approximated by a quadrature rule, e.g.
  - Monte Carlo
  - sparse grid
• Generally, only approximations of the solution \( u(x; y) \) of the PDE are obtainable

\[ \implies \text{discretization is needed with respect to both spatial-temporal and probabilistic variables} \]

- the output of interest \( F \) depends on the solution \( u \)

\[ \implies \text{spatial approximations of} \ u \text{engender spatial approximations of} \ F \]

- examples of spatial approximations are
  - finite element, finite difference, finite volume, spectral approximations
  - reduced-order models such as proper orthogonal decomposition (POD) or reduced basis methods
  - interpolants or least squares approximations
  - ...............
The goal is to construct approximate QoIs
- we have two approximation steps
  - spatial approximation of $F$
  - quadrature rule approximation of the statistical integral
- generally, the two types of approximations are uncoupled
  - one chooses a means to effect spatial approximation
    - finite elements, reduced-order model, . . .
  - one chooses a quadrature rule for approximating integrals
    - Monte Carlo, sparse grid, . . .
- for example, one could use the Monte Carlo rule

\[
\text{QoI} \approx \frac{1}{M} \sum_{m=1}^{M} G(y_m)
\]

where $G(y)$ is evaluated at $M$ random points in $\Gamma$ using a finite element approximation of the solution of the PDE.
• Our goal is to accelerate the convergence of this approach
Approximating QoIs using parameter sampling + spatial grid approximations

- Recall that we are addressing the task of approximating integrals over the parameter domain $\Gamma$ of the form

$$QoI = \int_{\Gamma} G(y) \rho(y) \, dy$$

that defines a QoI

- the integrand $G(y) = G\left(F(u(y))\right)$ depends on
  the solution $u(y)$ of the PDE
  and
  the chosen output of interest $F(u)$

- For any $y \in \Gamma$, we can solve the spatially discretized PDE using a grid size $h$ to obtain an approximation solution $u_h(y)$
• Having obtained the spatially approximate solution $u_h(y)$
  
  – we can obtain a spatial approximation $G_h(y) = G(F(u_h(y)))$ of the integrand $G(y)$

and then

– we can obtain a spatial approximation $\text{QoI}_h$ of the QoI given by

$$\text{QoI} \approx \text{QoI}_h = \int_{\Gamma} G_h(y)\rho(y) \, dy$$

• We then choose quadrature rule to approximate the integral over $\Gamma$

$$Q_M(\Phi) = \sum_{m=1}^{M} w_m \Phi(y_m)\rho(y_m)$$

using quadrature points $\{y_m\}_{m=1}^{M}$ in $\Gamma$ and quadrature weights $\{w_m\}_{m=1}^{M}$
We then define the computable approximation $Q_{M,h}$ of the QoI

$$Q_{M,h} = Q_M(G_h)$$

$$= \sum_{m=1}^{M} w_m G_h(y_m) \rho(y_m) \approx \int_{\Gamma} G_h(y) \rho(y) \, dy = Q_{h} \approx Q_{\text{ol}}$$

where

$$G_h(y_m) = G(F(u_h(y_m)))$$

for $m = 1, \ldots, M$

- $w_m = 1/M$ for $m = 1, \ldots, M$

- $\{y_m\}_{m=1}^{M}$ i.i.d. random samples in $\Gamma$  \(\Leftarrow\) Monte Carlo method

- $\{y_m\}_{m=1}^{M}$ deterministic sparse-grid points in $\Gamma$

$w_m$ for $m = 1, \ldots, M$ corresponding

sparse-grid quadrature weights  \(\Leftarrow\) a stochastic collocation method
Multifidelity methods for UQ

- We have
  - the truth model $\Leftarrow$ the expensive and accurate model we want to use
  - a parameter sampling scheme

  $\Rightarrow$ truth model + sampling scheme
  $\Rightarrow$ accurate but costs too much

- We also have some surrogate models $\Leftarrow$ cheaper and less accurate models

  $\Rightarrow$ any surrogate model by itself + sampling scheme
  $\Rightarrow$ costs less but is inaccurate

- Goal: find ways to

  combine \{truth model + surrogate models\} + sampling scheme

  so that

  $\Rightarrow$ costs less and is accurate
Several choices for models and sampling schemes

- Models of varying fidelity all depending on the parameters $y$
- Parameter sampling schemes

<table>
<thead>
<tr>
<th>Models of varying fidelity all depending on the parameters $y$</th>
<th>Parameter sampling schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolants</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>least-squares approximations</td>
<td>quasi-Monte Carlo</td>
</tr>
<tr>
<td>reduced-order models</td>
<td>Latin hypercube</td>
</tr>
<tr>
<td>simplified physics</td>
<td>CVT</td>
</tr>
<tr>
<td>coarser grid sizes</td>
<td>sparse grids</td>
</tr>
<tr>
<td>unconverged iterations</td>
<td>. . .</td>
</tr>
<tr>
<td>machine learning models</td>
<td></td>
</tr>
<tr>
<td>experimental data</td>
<td></td>
</tr>
<tr>
<td>. . . . . . . . . . . .</td>
<td></td>
</tr>
</tbody>
</table>

- we will look at three choices
- **Multilevel Monte Carlo**
  - truth model = fine grid discretization
  
  - surrogates and sampling scheme used $\Rightarrow$

<table>
<thead>
<tr>
<th>Deterministic models of varying fidelity all depending on the parameters $y$</th>
<th>Parameter sampling schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolants</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>least-squares approximations</td>
<td>quasi-Monte Carlo</td>
</tr>
<tr>
<td>reduced-order models</td>
<td>Latin hypercube</td>
</tr>
<tr>
<td>simplified physics</td>
<td>CVT</td>
</tr>
<tr>
<td>coarser grids</td>
<td>sparse grids</td>
</tr>
<tr>
<td>unconverged iterations</td>
<td>. . . . .</td>
</tr>
<tr>
<td>machine learning models</td>
<td>. . . . .</td>
</tr>
<tr>
<td>experimental data</td>
<td>. . . . .</td>
</tr>
</tbody>
</table>
• Multilevel stochastic collocation
  – truth model = fine grid discretization
  – surrogates and sampling scheme used ⇒

<table>
<thead>
<tr>
<th>Deterministic models of varying fidelity all depending on the parameters ( y )</th>
<th>Parameter sampling schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolants</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>least-squares approximations</td>
<td>quasi-Monte Carlo</td>
</tr>
<tr>
<td>reduced-order models</td>
<td>Latin hypercube</td>
</tr>
<tr>
<td>simplified physics</td>
<td>CVT</td>
</tr>
<tr>
<td>coarser grids</td>
<td>sparse grids</td>
</tr>
<tr>
<td>unconverged iterations</td>
<td>. . . . .</td>
</tr>
<tr>
<td>machine learning models</td>
<td>. . . . .</td>
</tr>
<tr>
<td>experimental data</td>
<td>. . . . .</td>
</tr>
</tbody>
</table>
• We will then consider a more general multifidelity method
  – several different types of surrogates are used
  – sampling via Monte Carlo
Multilevel Monte Carlo Methods

Multifidelity $\Leftarrow$ gauged by different spatial grid sizes;
explicit model hierarchy

Sampling $\Leftarrow$ Monte Carlo

+ many more
• Classical Monte Carlo (MC) methods determine the approximation to a QoI by the simple random sampling and averaging formula

\[
\text{QoI}_{mc, h_{\text{target}}} = \mathbb{Q}_{mc}^{h_{\text{target}}} (G_{h_{\text{target}}})
\]

\[
= \frac{1}{M_{mc}} \sum_{m=1}^{M_{mc}} G_{h_{\text{target}}} (y_m) \approx \int_{\Gamma} G_{h_{\text{target}}} (y) \rho(y) \, dy = \text{QoI}
\]

where

\[
\{y_m\}_{m=1}^{M_{mc}} \leftrightarrow M_{mc} \text{ i.i.d. random points in } \Gamma
\]

\[
G_{h_{\text{target}}} (y) = G\left( \left( F(u_{h_{\text{target}}}(y)) \right) \right) \leftrightarrow \text{integrand that depends on the (spatially discretized) solution of the PDE through a choice for an output of interest } F \text{ and the desired statistical information}
\]

– the spatial grid size \( h_{h_{\text{target}}} \) is chosen so that the spatial error is less than a prescribed tolerance \( \epsilon \)
the number of samples $M_{mc}$ is chosen so that the $O(1/\sqrt{M_{mc}})$ sampling error is commensurate with the $O(h_{\alpha_{target}})$ spatial discretization error

in this way the total error due to both sources is of $O(\epsilon)$

all

$M_{mc}$ solutions $\{u_{target}(y_m)\}_{m=1}^{M_{mc}}$

of the spatially discretized PDE corresponding to the

$M_{mc}$ sample points $\{y_m\}_{m=1}^{M_{mc}}$

are determined

using the same spatial grid of size $h_{target}$

- Multilevel Monte Carlo (MLMC) methods are designed to obtain an approximation of the QoI that has the same nominal accuracy $\epsilon$ but at less cost
• MLMC methods make use of a hierarchy of spatial grids
\[ h_l = \frac{h_{l-1}}{\eta} \quad \text{or} \quad h_l = \frac{h_0}{\eta^l} \quad \text{for } l = 1, \ldots, L \]

- \( h_0 \) is a given coarse grid size
- \( \eta > 1 \) is usually chosen to be an integer (most often 2)
- \( L \) is chosen as \( h_L = h_{\text{target}} \) so that there is hope that the approach meets the same accuracy threshold \( \epsilon \)
  - the actual MLMC algorithm then ensures that this is a certainty

• MLMC methods also make use of a hierarchy of MC quadrature rules
\[
Q_{M_l}^{mc}(\Phi) = \frac{1}{M_l} \sum_{m_l=1}^{M_l} \Phi(y_{m_l}) \approx \int_{\Gamma} \Phi(y) \rho(y) \, dy \quad \text{for } l = 0, 1, \ldots, L
\]
using the randomly sampled points \( \{y_{m_l}\}_{m_l=1}^{M_l} \) in \( \Gamma \)
• At each level \( l \) and for any \( y \in \Gamma \), we have a spatially approximate solution \( u_{h_l}(y) \) of the PDE with random inputs.

• We can then evaluate, for each level \( l \) and for any \( y \in \Gamma \), the spatial approximation \( G_{h_l}(y) = G\left( F\left( u_{h_l}(y) \right) \right) \) of the integrand \( G(y) \) in the QoI.

• Then, for each level \( l \), we have the spatial approximation of the QoI given by

\[
QoI_{h_l} = \int_{\Gamma} G_{h_l}(y) \rho(y) \, d(y) \quad \text{for } l = 0, \ldots, L
\]

• Obviously, the approximation \( QoI_{h_L} \) of the QoI on the finest spatial grid \( h_L = h_{tg} \) can be written in the form of the telescoping sum

\[
QoI_{h_L} = QoI_{h_0} + \sum_{l=1}^{L} (QoI_{h_l} - QoI_{h_{l-1}})
\]
• We express this more economically as

\[ Q_{oi h_L} = \sum_{l=0}^{L} \Delta h_l \]

where

\[ \Delta h_0 = Q_{oi h_0} \quad \text{and} \quad \Delta h_l = Q_{oi h_l} - Q_{oi h_{l-1}} \quad \text{for} \ l = 1, \ldots, L \]

• For any level \( l, \ l = 1, \ldots, L \), we determine an MC approximation of \( \Delta h_l \) using the level \( l \) MC quadrature rule \( Q_{mc}^{MC} \) using the points \( \{y_{ml}\}_{m_l=1}^{M_l} \)

i.e., we have

\[ \Delta_{mc}^{M_l,h_l} = Q_{mc}^{MC}(G_{h_l} - G_{h_{l-1}}) \]

\[ = \frac{1}{M_l} \sum_{m_l=1}^{M_l} (G_{h_l}(y_{ml}) - G_{h_{l-1}}(y_{ml})) \approx \Delta h_l \]
• The MLMC approximation of the QoI is then given as

$$QoI \approx QoI_{h_L} \approx QoI_{mlmc} = \sum_{l=0}^{L} \Delta_{mc}^{M_l,h_l}$$

- note that we do not apply the MC method to any $G_{h_l}(y)$ for $l > 0$, but rather to the differences $\Delta_{h_l}(y) = G_{h_l}(y) - G_{h_{l-1}}(y)$

• The total number of samples taken is $M = \sum_{l=0}^{L} M_l$

- note that because $h_l < h_{l-1}$
  the cost of obtaining samples of $\Delta_{h_l}(y)$ increases as the level $l$ increases

• How does one choose the number of samples $M_l$ for each level $l = 0, \ldots, L$?

- again, the aim is to have the error in the MLMC approximation to be less than a given tolerance $\varepsilon$

- then, $M_l$, $l = 0, \ldots, L$, are determined by minimizing the total sampling cost subject to the constraint that the total sampling error is of $O(\varepsilon)$
• The result of the optimization process is that
  the number of needed samples $M_l$ decreases as the level $l$ increases

• Thus, we see that
  – $M_l$ is large when $h_l$ is large
    - one has to do relatively lots of sampling when the realizations
      of the solution of the discretized PDE are relatively cheap
  – $M_l$ is small when $h_l$ is small
    - one has to do relatively little sampling when the realizations
      of the solution of the discretized PDE are relatively expensive
Thus, there is a tradeoff in using MLMC compared to MC to obtain the same error for both methods:

- the total number of samples $\sum_{l=0}^{L} M_l$ taken by MLMC may be larger than the total number of samples used by MC however
  - all the MC samples are taken on the finest spatial grid whereas
    - some of the MLMC samples are taken on coarser spatial grids

Who wins?
  - does the MLMC method save over the MC?
    - the answer is yes
Why does MLMC win?

- the **sampling error** at the level $l$ is proportional to $\sigma_l / \sqrt{M_l}$
  - $\sigma_l$ is a measure of the standard deviation of $\Delta_{h_l}(y)$

- the key is that the **variances** $\sigma^2_l$ of the differences $\Delta_{h_l}(y) = u_{h_l}(y) - u_{h_{l-1}}(y)$ decrease as $l$ increases

- thus

  to equilibrate errors across all levels

  one can use

  a smaller $M_l$ for larger $l$
Multiindex Monte Carlo

- A generalization of MLMC in which in addition to first differences, higher-order mixed differences are used

A.-L. Haji-Alia, F. Nobile, and R. Tempone

Multiindex Monte Carlo: when sparsity meets sampling, arXiv:1405.3757
MULTILEVEL STOCHASTIC COLLOCATION METHODS

Multifidelity ⇐ gauged by different spatial grid sizes; explicit model hierarchy

Sampling ⇐ sparse grids

A. Teckentrup, P. Jantsch, C. Webster, and M. Gunzburger
A multilevel stochastic collocation method for partial differential equations with random input data
SIAM/ASA Journal on Uncertainty Quantification 3, 2015, 1046-1074.
To define a multilevel stochastic collocation method (MLSC), we again introduce a hierarchy of spatial grids

\[ h_l = \frac{h_{l-1}}{\eta} \quad \text{or} \quad h_l = \frac{h_0}{\eta^l} \quad \text{for } l = 1, \ldots, L \]

- \( h_0 \) is a given coarse grid size
- \( \eta > 1 \) is usually chosen to be an integer (most often 2)
- \( L \) is chosen as \( h_L = h_{\text{target}} \) so that there is hope that the approach meets the same accuracy threshold \( \epsilon \)

- the actual MLMC algorithm then ensures that this is a certainty

MLSC methods also make use of a hierarchy of quadrature rules

\[ Q_{M_l}^{\text{sc}}(\Phi) = \sum_{m_l=1}^{M_l} w_{m_l} \Phi(y_{m_l}) \approx \int_\Gamma \Phi(y) \rho(y) \, dy \quad \text{for } l = 0, 1, \ldots, L \]

using quadrature points \([y_{m_l}]_{m_l=1}^{M_l}\) in \( \Gamma \) and quadrature weights \([w_{m_l}]_{m_l=1}^{M_l}\)
At each level $l$ and for any $y \in \Gamma$, we have a spatially approximate solution $u_{h_l}(y)$ of the PDE with random inputs.

We can then evaluate, for each level $l$ and for any $y \in \Gamma$, the spatial approximation $G_{h_l}(y) = G\left(F\left(u_{h_l}(y)\right)\right)$ of the integrand $G(y)$ in the QoI.

Then, for each level $l$, we can then determine the spatial approximation of the QoI given by

$$QoI_{h_l} = \int_{\Gamma} G_{h_l}(y) \rho(y) \, d(y) \quad \text{for } l = 0, \ldots, L$$

Obviously, the approximation $QoI_{h_L}$ of the QoI on the finest spatial grid $h_L$ can be written in the form of the telescoping sum

$$QoI_{h_L} = QoI_{h_0} + \sum_{l=1}^{L} (QoI_{h_l} - QoI_{h_{l-1}})$$
• We express this more economically as

\[ QoI_{h_L} = \sum_{l=0}^{L} \Delta h_l \]

where

\[ \Delta h_0 = QoI_{h_0} \quad \text{and} \quad \Delta h_l = QoI_{h_l} - QoI_{h_{l-1}} \quad \text{for } l = 1, \ldots, L \]

• For any level \( l, \ l = 1, \ldots, L \), we determine an SC approximation of \( \Delta h_l \) using the level \( l \) quadrature rule \( Q_{M_l}^{sc} \), i.e., we have

\[ \Delta_{M_l,h_l}^{sc} = Q_{M_l}^{sc}(G_{h_l} - G_{h_{l-1}}) \]

\[ = \sum_{m_l=1}^{M_l} w_{m_l}(G_{h_l}(y_{m_l}) - G_{h_{l-1}}(y_{m_l})) \rho(y_{m_l}) \approx \Delta h_l \]
Then, the multilevel stochastic collocation (MLSC) approximation of the QoI is given by

\[
\text{QoI} \approx \text{QoI}_{h_L} \approx \text{QoI}^{\text{mlsc}}_{h_L} = \sum_{l=0}^{L} \Delta_{h_l,M_{L-l}}^{sc} = \sum_{l=0}^{L} Q_{h_{l},M_{L-l}}^{sc} \left( G_{h_{l}}(y_{m_{l}}) - G_{h_{l-1}}(y_{m_{l}}) \right)
\]

Rather than simply applying a quadrature rule to \( G_{h_{l}}(y) \), this approximation applies a different level quadrature rule to each difference \( G_{h_{l}} - G_{h_{l-1}} \)

– note the reversal of the order of the index for the quadrature rule relative to the index for the spatial grid
Thus

the accuracy of the quadrature rule used decreases as the level \( l \) increases

so that

the number of samples taken decreases as the spatial grid size decreases

so that

the number of samples taken decreases as the cost of obtaining a sample increases

– more accurate quadrature rules requiring a relatively large number of quadrature points are used with the relatively cheaper coarser grid spatial approximations

– less accurate quadrature rules requiring a relatively small number of quadrature points are used with relatively more expensive finer grid spatial approximations
Error and cost estimates

- We make the following assumptions

  spatial error \[ \text{QoI} - \text{QoI}_{h_l} \Rightarrow O(h_l^\alpha) \]
  difference \[ \text{QoI}_{h_l} - \text{QoI}_{h_{l-1}} \Rightarrow O(h_l^\beta) \]
  quadrature error \[ \text{QoI}_{h_l} - \Delta_{h_l, M_l}^{sc} \Rightarrow O(M_l^{-\mu}) \]

- then, if we choose the number of samples

\[
M_{L-l} = O(h_L^{-\frac{\alpha}{\mu}} h_l^{\frac{\beta}{\mu}}) \quad \text{for } l = 0, \ldots, L
\]

we have that the error in the multilevel stochastic collocation approximation of the quantity of interest satisfies

\[
\text{QoI} - \text{QoI}_{h_L}^{mlsc} = O(h_L^\alpha)
\]
• If $\beta = \alpha$ (the usual case), we have that

\[
\text{number of parameter samples} \quad \Rightarrow \quad M_{L-l} = O\left(\left(\frac{h_l}{h_L}\right)^{\frac{\alpha}{\mu}}\right)
\]

taken on the spatial grid $h_l$

– for example, we have

\[
\begin{align*}
\text{number of parameter samples taken on} & \Rightarrow M_{L-L} = O\left(\left(\frac{h_L}{h_L}\right)^{\frac{\alpha}{\mu}}\right) = O(1) \\
\text{finest grid } h_L (l = L) & \\
\text{number of parameter samples taken on} & \Rightarrow M_{L-0} = O\left(\left(\frac{h_0}{h_L}\right)^{\frac{\alpha}{\mu}}\right) \\
\text{coarsest grid } h_0 (l = 0) &
\end{align*}
\]

– thus, we see that indeed one does

- relative few samples of expensive, relatively fine spatial grid discretizations of the PDE

- relative many samples of cheap, relatively coarse spatial grid discretizations of the PDE
• If we also assume that

  – the cost $C_l$ of a single solve of the discretized PDE with the grid $h_l$ 

  \[
  \Rightarrow \quad O(h_l^{-\gamma})
  \]

• The total cost for obtaining the MLSC approximation is then

  \[
  C_{mlsc} = \sum_{l=0}^{L} M_{L-l} C_l
  \]

• If we let $C_{sc}$ denote the total cost to render $QoI - QoI_{sc}^{h_L} = O(\varepsilon)$ using the single-level standard SC method using a spatial grid size $h_L$, we have the estimates

  \[
  \frac{C_{mlsc}}{C_{sc}} = \begin{cases}
  O(\varepsilon^{\frac{\gamma}{\alpha}}) & \text{if } \alpha > \mu \gamma \\
  O(\varepsilon^{\frac{\gamma}{\alpha}} |\ln \varepsilon|^{1+\frac{1}{\mu}}) & \text{if } \alpha = \mu \gamma \\
  O(\varepsilon^{\frac{1}{\mu}}) & \text{if } \alpha < \mu \gamma
  \end{cases}
  \]
• If $\beta > \gamma \mu$
  
  – the cost of obtaining one sample of $u_{h_l}$ grows slowly with respect to $l$
  
  – most of the computational effort of the multilevel approximation is at the coarsest level $l = 0$
  
  – the savings in cost compared to the standard SC method hence corresponds to the difference in cost between obtaining samples $u_{h_0}$ on the coarse grid $h_0$ and obtaining samples $u_{h_L}$ on the fine grid $h = h_L$ used by the standard SC method
  
  – this gives a saving of $O\left(\left(\frac{h}{h_0}\right)^\gamma\right) = O\left(\varepsilon^{\gamma/\alpha}\right)$

• If $\beta = \mu \gamma$
  
  – the computational effort is spread evenly across the levels
  
  – this gives a savings of $O\left(\varepsilon^{\gamma/\alpha} \ln \varepsilon \left|\ln \varepsilon\right|^{1+\frac{1}{\mu}}\right)$
• If $\beta < \gamma \mu$
  – the computational cost of computing one sample of $u_{h_l}$ grows quickly with respect to $l$
  – most of the computational effort of the multilevel approximation is on the finest level $l = L$
  – the cost savings compared to the standard SC method hence corresponds approximately to the difference between
    
    \[ M_0 \Rightarrow \text{for MLSC, the number of samples done of the finest-grid PDE approximation} \]
    
    and
    
    \[ M_{SC} \Rightarrow \text{for SC, the number of samples done of the finest-grid PDE approximation} \]
    
    – this gives a savings of \( O(M_0/M_{SC}) = O((h_L^\beta)^{1/\mu}) = O(\varepsilon^{\beta/\alpha \mu}) \)
For the types of hierarchies of quadrature rules typically used in stochastic collocation methods, there is often an abrupt increase in the number of quadrature points even if one goes to the next member of the hierarchy

– the theoretically predicted number of points will not, in general, have an associated quadrature rule

– as a result, theoretical predictions of the number of quadrature points needed cannot be well matched by a member of the quadrature rule hierarchy

– in such cases, in choosing a quadrature rule, one may have to choose a rule that uses substantially more quadrature points than what the theory predicts is necessary
– however,

- it is reasonable to assume that there exists an $\widetilde{M}$-point quadrature rule in the hierarchy such that

$$M_{\text{theory}} \leq \widetilde{M} \leq CM_{\text{theory}}^s$$

for some $s \geq 1$ and where $M_{\text{theory}}$ denotes the theoretically predicted number of points needed

- one can then again proceed to derive a bound on the $\varepsilon$-cost of the resulting multilevel approximation
• The abrupt change in the number of quadrature points and hence in the accuracy of the associated quadrature rule as one moves up the hierarchy, is the cause of sometimes wanting to use the same quadrature rule for more than one level

– suppose that in going from level \( l-1 \) to level \( l \) one has to use a new quadrature rule

– because of the abrupt change in the accuracy of the rules, the new quadrature rule used at level \( l \) may also be good enough to use at the level \( l+1 \)

– in this case, the same quadrature rule can be used at levels \( l \) and \( l+1 \)
• Elliptic PDE example
  – QoI = average of $u$ in neighborhood of $(1/2, 1/2)$
  – assumptions “confirmed” numerically
  – comparisons of computational costs: $d = 1$, $N = 20$

*Computational cost versus relative error $\varepsilon$*
Multiindex stochastic collocation

A.-L. HAJI-ALIA, F. NOBILE, L. TAMELLINI, AND R. TEMPO

Multiindex stochastic collocation for random PDEs, arXiv:1508.07467

A.-L. HAJI-ALIA, F. NOBILE, L. TAMELLINI, AND R. TEMPO

Multiindex stochastic collocation convergence rates for random PDEs with parametric regularity
arXiv:1511.05393
OPTIMAL MULTIMODEL, MULTIFIDELITY MONTE CARLO METHOD

Multifidelity ⇐ a plethora of surrogates; implicit model hierarchy

Sampling ⇐ Monte Carlo

B. Peherstorfer, K. Willcox, and M. Gunzburger
Optimal model management for multi fidelity Monte Carlo estimation

B. Peherstorfer, T. Cui, Y. Marzouk, and K. Willcox
Multifidelity importance sampling
\[
\begin{aligned}
\mathbf{y} \in \Gamma \subset \mathbb{R}^N & \iff \text{vector of random inputs} \\
F^{(hf)}(\mathbf{y}) : \Gamma \to \mathbb{R} & \iff \text{high-fidelity output functional of interest} \\
\text{QoI}^{(hf)} & \iff \text{statistical information about } F^{(hf)}(\mathbf{y}) \\
C_{hf} & \iff \text{cost to obtain } F^{(hf)}(\mathbf{y}) \text{ for any given } \mathbf{y} \in \Gamma
\end{aligned}
\]
\[
\begin{align*}
\mathbf{y} & \in \Gamma \subset \mathbb{R}^N \quad \iff \quad \text{vector of random inputs} \\
& \Downarrow \\
\text{high-fidelity PDE solves at sample points in } \Gamma \\
& \Downarrow \\
F^{(hf)}(\mathbf{y}) : \Gamma \to \mathbb{R} & \iff \quad \text{high-fidelity output functional of interest} \\
& \Downarrow \\
\text{use the high-fidelity PDE solves to approximate the QoI} \\
& \Downarrow \\
\text{QoI}^{(hf)} & \iff \quad \text{statistical information about the output of interest}
\end{align*}
\]

\[C_{hf} \geq 0 \quad \iff \quad \text{cost to obtain } F^{(hf)} \text{ for any given } \mathbf{y} \in \Gamma \approx \text{cost of the high-fidelity PDE solve}\]

\[-G^{(hf)}(\mathbf{y}) = G(F^{(hf)}(\mathbf{y})) \quad \iff \quad \text{integrand in } \text{QoI}^{(hf)}\]
• Then, the high-fidelity Monte Carlo estimator of the QoI is given by

\[ QoI_{M_{hf}}^{(hf)} = \frac{1}{M_{hf}} \sum_{m=1}^{M_{hf}} G^{(hf)}(y_m) \approx QoI \]

\[ \{y_m\}_{m=1}^{M_{hf}} \text{ are } M_{hf} \text{ i.i.d. samples in } \Gamma \]

\[ \text{requires } M_{hf} \text{ high-fidelity solves of the PDE} \]

\[ \text{cost of doing the } M_{hf} \text{ solves is } M_{hf}C_{hf} \]

• Mean-square error of the high-fidelity Monte Carlo estimator

\[ \text{error}(QoI_{M_{hf}}^{(hf)}) = \mathbb{E} \left[ \left( \mathbb{E}[G^{(hf)}] - G^{(hf)} \right)^2 \right] = \frac{1}{M_{hf}} \mathbb{V}[G^{(hf)}] \]

• If \( C_{hf} \gg 1 \), one looks for ways to reduce the cost of approximating the QoI
• In many situations, a surrogate model is available for which the corresponding approximation $F^{(sur)}(\mathbf{y})$ is less expensive to obtain
  
  if $C^{(sur)}$ denotes the cost of obtaining $F^{(sur)}(\mathbf{y})$, we have
  
  $C^{(sur)} \ll C^{(hf)}$
The construction of some surrogates, e.g., interpolants, reduced-order models require the solution of the expensive high-fidelity model

- the hope is that the high offline construction cost is amortized over many subsequent online solves using the surrogate

The Monte Carlo estimator of the QoI is now given by

\[ QoI_{M_{sur}}^{(sur)} = \frac{1}{M_{sur}} \sum_{m=1}^{M_{sur}} G^{(sur)}(y_m) \approx \mathbb{E}(F) = QoI \]

where

\[ G^{(sur)}(y) = G\left(F^{(sur)}(y)\right) \]

- by using the surrogate instead of the high-fidelity model, the cost of determining the approximation of the QoI reduces from \( M_{hf}C_{hf} \) to the much smaller \( M_{sur}C_{sur} \)
Unfortunately, less costly surrogates are also usually less accurate

- in which case $QoI_{\text{sur}}$ is not as good estimator of the QoI as is $QoI_{\text{hf}}$
  - even if $M_{\text{sur}} \gg M_{\text{hf}}$, i.e., the number of samples taken of the surrogate is greater than that taken of the high-fidelity model

- there is no guarantee of the efficacy of surrogates if reliable error bounds or estimators are not available
  - and even if bounds on the surrogate model error are available, it is still not completely obvious how the error on the surrogate model output propagates onto the error of the estimate

- in some cases, a surrogate can be rebuilt (adding to the construction cost) to improve its accuracy
In many situations, several surrogates $F^{(2)}(y), \ldots, F^{(K)}(y)$ for the output of interest are available

- for $k = 2, \ldots, K$

- the cost $C_k$ for obtaining $F^{(k)}(y)$ is often significantly less than is the cost $C_1 = C_{hf}$ for obtaining $F^{(1)}(y) = F^{(hf)}(y)$

- QoI$^{(k)}$ can viewed as a “lower” fidelity “approximation” of QoI$^{(1)} = QoI^{(hf)}$
Many types of surrogate models are possible

- interpolants, least-squares approximations
- projection-based reduced-order models
- simplified-physics models
  - linearized PDEs
  - averaged models (e.g., RANS)
- coarse-grid approximations
- stopping iterations early resulting in higher residuals
- machine-learning-based models
- experimental data

these are very different from each other with respect to cost, fidelity, and construction
The hope is that somehow the cheaper surrogate models can be used to speed up, without compromising accuracy, the approximation of the QoI corresponding to the high-fidelity model.
• Model management
  – combine all model outputs of interest $F^{(hf)} = F^{(1)}, F^{(2)}, \ldots, F^{(K)}$
  – use surrogates $F^{(2)}, \ldots, F^{(K)}$ for speedup
  – use $F^{(1)}$ for accuracy
  – balance the number of model evaluations (number of samples) among models
  – establish accuracy guarantees

• Because we want to be able to incorporate models of vastly different nature, we generally do not have available estimates of the relative accuracy of those models
  – this is unlike the situation for MLMC and MLSC methods for which we have explicit information about those differences because the different models are generated solely by changing the grid size
  – thus, we do not assume we have any a priori information about fidelities of any of any model relative to the other ones
UQ ?? \[ \text{QoI}^{(mfmc)} = \text{cheaper QoI approximation} \]

\[ F^{(1)}(y), \ldots, F^{(K)}(y) \]

- high-fidelity model
- surrogate model 2
- \( \vdots \)
- surrogate model \( K \)
Before going on to remove the ?? in the UQ box, we note that there are other applications in which surrogates have been used to improve the efficiency of a high-fidelity computation, including closed-loop settings:

- optimization: corrections
- inference: delayed acceptance MCMC
• Acceleration of Monte Carlo with surrogate models
  – multilevel Monte Carlo [Giles 2008], [Heinrich 2001] + many others
  – reduced basis and control variates [Boyaval 2012], [Vidal et al. 2015, 2015]
  – data-fit models and control variates [Tracey et al. 2013]
  – two models and control variates [Ng & Willcox 2012, 2014]
  – importance sampling [Li & Xiu], [Peherstorfer et al. 2015]
Multifidelity Monte Carlo method

- Setup
  - \( \mathbf{y} \in \Gamma \) \( \leftarrow \) vector of random parameters with joint PDF \( \rho(\mathbf{y}) \)
  - \( F^{(1)}(\mathbf{y}) \) \( \leftarrow \) output of high-fidelity model (“truth”)
  - \( F^{(2)}(\mathbf{y}), \ldots, F^{(k)}(\mathbf{y}) \) \( \leftarrow \) output of \( K - 1 \) surrogate models
  - \( C_k \) \( \leftarrow \) cost for obtaining \( F^{(k)}(\mathbf{y}) \) for given \( \mathbf{y} \in \Gamma \)
  - \( M_k \) \( \leftarrow \) number of model evaluations of \( F^{(k)} \)
    with \( 0 < M_1 \leq M_2 \leq \cdots \leq M_K \)
    (only needed to find a closed-form expression for the variance)
  - \( \text{QoI}^{(1)} = \text{QoI}^{(hf)} \) \( \leftarrow \) statistical information about the high-fidelity output \( F^{(1)}(\mathbf{y}) = F^{(hf)}(\mathbf{y}) \)

- Goal: estimate \( \text{QoI}^{(1)} \)
  specifically \( \leftarrow \) \( \text{QoI}^{(1)} = \mathbb{E}[G^{(1)}(\mathbf{y})] \)
• Problem definition: given a computational budget $p > 0$
  – derive estimator of $\text{QoI}^{(1)} = \mathbb{E}[G^{(1)}(y)]$ that uses surrogate models
  – estimator has cost $p$
  – estimator is unbiased with respect to $\text{QoI}^{(1)}$
  – estimator has lower MSE than the MC estimator also having cost $p$

• No assumptions about surrogate models
  – use as many surrogate models of any type as one has available
  – no a priori error bounds
  – no a posteriori error estimators
  – models do not have to form an explicit hierarchy through known error bounds, e.g., as in MLMC or MLSC where such bounds are known in terms of the grid size
    - the hierarchy is established implicitly through correlations between models
• Model management questions
  – how to combine models? ⇐ control variates
  – how to balance model evaluations among them? ⇐ optimization

• Model sampling
  – draw $M_K$ (i.i.d.) realizations $y_1, \ldots, y_{M_K} \in \Gamma$ of $y$
  – for $k = 1, \ldots, K$
    - evaluate $F^{(k)}$ at the realizations $y_1, \ldots, y_{M_k}$ to obtain
      $$F^{(k)}(y_1), \ldots, F^{(k)}(y_{M_k})$$
    - construct the $k$th MC estimate $Q_{\text{ol}}^{(k)}_{M_k} = \frac{1}{M_k} \sum_{m=1}^{M_k} G^{(k)}(y_m)$
    - reuse evaluations to estimate $Q_{\text{ol}}^{(k)}_{M_{k-1}} = \frac{1}{M_{k-1}} \sum_{m=1}^{M_{k-1}} G^{(k)}(y_m)$
• The multifidelity Monte Carlo (MFMC) estimate for $QoI^{(mfmc)}$ is then defined as

$$QoI^{(mfmc)} = QoI^{(1)}_{M_1} + \sum_{k=2}^{K} \alpha_k \left( QoI^{(k)}_{M_k} - QoI^{(k)}_{M_{k-1}} \right)$$

• Properties of the MFMC estimator

  – cost: $\hat{C}_{MK} = \sum_{k=1}^{K} M_k C_k$

  – the MFMC estimator is unbiased, even in absence of error bounds or estimators for the surrogate models

• A feature of our MFMC method is the optimal (with respect to the MSE) selection of

  the number of model evaluations $0 < M_1 \leq M_2 \leq \cdots \leq M_K$

and of

  the coefficients $\alpha_2, \ldots, \alpha_K$

that is applicable to any types of surrogate models
• The variance of MFMC estimator $Q_{\text{ol}}^{(mfmc)}$ is

$$\mathbb{V}[Q_{\text{ol}}^{(mfmc)}] = \frac{\sigma_1^2}{M_1} + \sum_{k=2}^{K} \left( \frac{1}{M_{k-1}} - \frac{1}{M_k} \right) (\alpha_k^2 \sigma_k^2 - 2\alpha_k s_k \sigma_1 \sigma_i)$$

where

$$\sigma_k^2 = \text{the variance of } F^{(k)}(\mathbf{y})$$

$$s_k = \text{correlation coefficient between } F^{(1)}(\mathbf{y}) \text{ and } F^{(k)}(\mathbf{y})$$

• Because the MFMC estimator is unbiased, the MSE of MFMC estimator is

$$\text{error}(Q_{\text{ol}}^{(mfmc)}) = \mathbb{V}[Q_{\text{ol}}^{(mfmc)}]$$

• Define the vectors

$$\tilde{\mathbf{M}} = (M_1, \ldots, M_K)^T \in \mathbb{R}^K$$

$$\tilde{\mathbf{C}} = (C_1, \ldots, C_K)^T \in \mathbb{R}^K$$

$$\tilde{\mathbf{\alpha}} = (\alpha_2, \ldots, \alpha_K)^T \in \mathbb{R}^{K-1}$$
• Given a computational budget $p$, the optimal $\vec{M}$ and $\vec{\alpha}$ are determined by solving the problem:

$$\text{find } \vec{M}^* \text{ and } \vec{\alpha}^* \text{ such that}$$

$$(\vec{M}^*, \vec{\alpha}^*) = \arg \min_{\vec{M} \in \mathbb{R}^K, \vec{\alpha} \in \mathbb{R}^{K-1}} \mathbb{V}[\text{QoI}^{(mfmc)}]$$

subject to

$$M_k - M_{k-1} \geq 0, \quad k = 2, \ldots, K$$

$$M_1 > 0$$

$$\vec{M} \cdot \vec{C} = p$$

– a unique (analytic) solution exists if

$$s_1^2 > \cdots > s_K^2$$

and

$$\frac{C_{k-1}}{C_k} > \frac{s_{k-1}^2 - s_k^2}{s_k^2 - s_{k+1}^2}, \quad k = 2, \ldots, K$$
• Variances, corellation coefficients, and costs, if not known, can be approximated by sampling the outputs \( \{ F^{(k)} \}_{k=1}^{K} \) of the various models.

• A continuous version of the optimization problem is actually solved, after which the optimal solution is replaced by the nearest integer. Because the number of samples is usually large, changing the optimal (non-integer) solution by less than \( \pm 1 \) is not a problem.

• The requirement \( s_{1}^{2} > \cdots > s_{K}^{2} \) is easily arranged by re-ordering the models.

• The requirement \( \frac{C_{k-1}}{C_{k}} > \frac{s_{k-1}^{2} - s_{k}^{2}}{s_{k}^{2} - s_{k+1}^{2}} \) is violated when a surrogate model is not well correlated with the high-fidelity model and/or the cost of the model is high.

  - if this is the case, the surrogate model should not be used.

  - the worst-case scenario is that only the high-fidelity model is left, in which case we learn that the surrogates chosen are not useful for accelerating the QoI estimation for the high-fidelity model.
• Solution

- optimal coefficients: for \( k = 2, \ldots, K \)
  \[
  \alpha_k^* = \frac{\sigma_1}{\sigma_k} s_k
  \]

- optimal number of samples: for \( k = 2, \ldots, K \)
  \[
  M_k^* = M_1^* r_k^*
  \]

with

\[
M_1^* = \frac{p}{\vec{C} \cdot \vec{r}^*}, \quad \vec{r}^* = (r_2^*, \ldots, r_K^*)^T \in \mathbb{R}^{K-1}
\]

and, for \( k = 2, \ldots, K \)

\[
r_k^* = \sqrt{\frac{C_1 (s_k^2 - s_{k-1}^2)}{C_k (1 - s_2^2)}}
\]

- number of model evaluations \( M_k^* \) depend linearly on budget \( p \)
• Reduction in variance (or MSE) compared to benchmark Monte Carlo estimator \( \hat{Q}_{M_k}^{(1)} \) with same cost \( p \)
  
  – the variance reduction ratio is

\[
\frac{(\hat{Q}_{M_k}^{(1)})}{e(\hat{Q}_{M_{mc}}^{(1)})} = \left( \sum_{k=1}^{K} \sqrt{\frac{C_k}{C_1}} \left( s_k^2 - s_{k+1}^2 \right) \right)^2
\]

– MFMC has lower MSE than the Monte Carlo estimator if

\[
\sum_{k=1}^{K} \sqrt{\frac{C_k}{C_1}} \left( s_k^2 - s_{k+1}^2 \right) < 1
\]

– condition on the collective whole of the models (sum), not on properties of each model separately

– interaction between the models is what drives the MFMC estimator, not model properties alone
Numerical example: locally damaged plate in bending

- Input $y \in \mathbb{R}^4$: nominal thickness, load, damage
  - uniformly distributed in $[0.05, 0.1] \times [1, 100] \times [0, 0.2] \times [0, 0.05]$

- Output of interest $F$: maximum deflection of plate

- QoI: the expected value of $F$

- Six models
  - high-fidelity model: FEM, 300 DoFs
  - reduced-order model: POD with 10 DoFs
  - reduced-order model: POD with 5 DoFs
  - reduced-order model: POD with 2 DoFs
  - data-fit model: linear interpolation with 256 points, using data from high-fidelity model
  - support vector machine with 256 points

- Variance, correlation, and costs estimates determined from 100 samples of each model
thickness: no damage

thickness: damage up to 20%

deflection: no damage

deflection: damage up to 20%
- Variance and MSE for several model combinations

- largest improvement from one → two and two → three models
- adding further reduced/SVM models reduces MSE only slightly
- theoretical and computational MSEs match well

- MFMC provides for optimal load balancing on HPCs because we know a priori how often each model will be evaluated
Distribution of number of samples (model evaluations) among models

- MFMC distributes samples among models depending on correlations and costs
- number of samples changes exponentially between models
- highest number of samples for data-fit and SVM models (cost ratio $C_1/C_6 \approx 10^6$)
Numerical nonlinear example: limit cycle oscillation (LCO)

- Non-adiabatic tubular reactor
- Arrhenius-type (exponential) nonlinear term
- Input: Damköhler number $\sim \mathcal{N}(0.167, 0.03)$
- Output of interest: LCO amplitude
- QoI: expected value of the output of interest
- Four models
  - high-fidelity model: finite difference with 198 DoFs
  - reduced-order model: POD with 10 DoFs
  - reduced-order model: POD + DEIM with 10 DoFs
  - data-fit model: interpolation with 10 points
- Speedup of four orders of magnitude with high-fidelity + reduced order (POD) + data

- Adding another reduced model (DEIM), only a slight improvement

- Agreement of variance (theoretical) and estimated MSE (numerical)
REFERENCES FOR MULTIFIDELITY


