Uncertainty Quantification

Rudraprasad Bhattacharyya

Abstract

This document is prepared based on the lectures and notes for the graduate level course- ‘Uncertainty Quantification’ (CE 6310) taught by Dr. Sankaran Mahadevan at Vanderbilt University. I have prepared this document based on the study materials I have gathered during ‘Spring 2015’ and ‘Spring 2019’.

Contents

1 Introduction

2 Variation in Space and time

2.1 Random field or random process

Jan 11, 2019

At any time t, for example $t = t_1$ if we take the values here, that would substitute a random variable. So we can imagine, at each time $t = t_1,\ t_2,\ \cdots$ we basically have lots of random variables. Those random variables constitute the random process. **Random process/random field** is collection of random variables.

There is a correlation between these random variables. Let's say the random variable at t_1 and random variable at t_2, they are coming from the same random process so they are correlated. Hence, based on that we can say something about the type of the random process we have. If all the statistics are constant throughout i.e. no change then that would be called a stationary random process. All the random variables have the same statistics,i.e. same mean, variance and distribution. Now the correlations between any two random variables is only a function of distance between them. So that is an ideal case and we have correlations only a function then that would be referred to as a weakly stationary random process. Strongly stationary would be those where all the statistics are same and they are perfectly correlated; whereas, when correlation is the function of the distance then that would be weakly stationary.

The non-stationary would be specifically connected to the actual location. Even though these two locations and these two locations have same distance but their correlation is completely different. That means this is the function of the positions, not distance. So based on that we would come up with different methods for simulating the random process.

Note: t could be X i.e. spatial coordinate. Both, t and X, follow the same mathematics. At any time t, we could have a distribution, if the distribution is Gaussian everywhere then it is called a Gaussian random process. That's something what we would use quite frequently. You may have Gaussian at one location and non-Gaussian at other location. Then it will become difficult to handle.

Now if we want do the simulation, doing Monte Carlo simulation, we want to represent the input which is really not a random variable, but a random process which means now we have to simulate a collection of random variables. For example, suppose we say this is a random process of duration 10 s excitation and we say, "We are going to represent it by 10 random variables". Then we are discretizing the coordinate into 10 locations. So that means we have 10 random variables to represent that random process.

In '*Probabilistic Methods in Engineering Design*' course when we solved the problem $Y = X_1X_2 - X_3$, X_1, X_2, and X_3 each one was a random variable, it had a one distribution and we generated samples from them. But now if we say one of this variables X_1 is also distributed over space or time then we have to simulate each quantity which explodes into multiple random variables. So we have to simulate all of them. And then simulate our random field or random process.

How to simulate random field/ random process?

Depending on the type of random process you require different methods for simulation. Simulation means we want to be able to generate the realizations like this across space or time. In the case of distribution it was just a single random variable. Simulation involves generating these single point values of the random variable from the distribution. Now if it is random field or random process it requires simulating the entire signal. So if we are discretizing into (let say) 10 locations (we can think of this as 10 locations), you are told that they are completely independent of each other. Suppose we model this process using 10 independent

random variables. Then at each location you generate some random sample of that. Add picture here.
Then you generate another set of random variables. 'Independent' means there is no correlation between one location to the other. So such a random process would be referred to as a 'White Noise'. Picture
For a white noise random process or random field is very jagged. In the context of random field and random process we use the terms 'Auto-correlation' and 'cross-correlation'. *Auto-correlation* is the correlations between the random variables within the same random process/field. On the other hand, *cross-correlation* is the correlation between two different quantities derived from two different random processes/fields.

In the example from '*Probabilistic Methods in Engineering Design*' course the equation $Y = X_1X_2 - X_3$, if we talk about correlation between X_1 and X_2, we are talking about cross-correlation. Within X_1 correlation between different random variables is called auto-correlation. If it is a 'White noise' process, that means the correlation is dropping very fast. Correlation of a random variable with itself is 1 and correlation right next discretization is zero. We use 2 quantities to characterize the correlation between random fields.

1. Auto-correlation function, $R(\tau)$
2. Power spectral density (PSD), $S(\omega)$

Power spectral density is the Fourier Transform of auto-correlation function.

$$S(\omega) \propto \int R(\tau)e^{-j\omega\tau}d\tau \tag{1}$$

where $j = \sqrt{-1}$, ω is the frequency and τ is the time or space domain.

Add a picture of white noise

The auto-correlation function of a white noise is a delta function. That means this signal has all frequencies. Mathematically, if it is a very jagged one that means it contains many sinusoidal waves with different frequencies. White noise is a broad band signal. referring the bandwidth from picture Figure 4.6 (narrow band.

Why is this useful?
This will tell us how many locations should I use to discretize the random process. If it is jagged, I need lots of points to represent the random process. For simulation of random processes/fields auto-correlation function is very useful. If there is a strong correlation persists over a longer distance, then I can use one random variable, perhaps 2. If it was a perfect sine wave, that means it has single frequency. It is perfectly correlated. The auto correlation function looks like Figure, power spectral density is a delta function due to presence of single frequency.

What is meant by 'randomness'? The randomness is with respect to the amplitude, not the frequency. The correlation characteristics of random fields or random processes will guide you to decide how many points you need to discretize. That's the first thing we do.

Now with this idea we might think about how to simulate random process/field. We are simulation random variables which are correlated.

Simulating correlated random variables:
Write equations Correlated normal —–>

For each variable in the set you have a random number u.

This is how we generate correlated normal variables. If it is a Gaussian random process or field, each one is Gaussian and the correlation function is the distance between 2 points. If we want to represent a random process/field by 10 random variables and we have a correlation matrix of size 10×10, we get the eigenvalues and eigenvectors. The only difference now is that the covariance matrix is coming from an auto-correlation function.

$$\mathrm{Cov}(X_1, X_2) = \sigma_{X_1}\sigma_{X_2}\rho_{X_1X_2} \tag{2}$$

where $\rho_{X_1X_2}$ is the correlation. In our current notation, the equation is

$$\mathrm{Cov}(X_1, X_2) = \sigma_{X_1}\sigma_{X_2}R(X_1, X_2) \tag{3}$$

X_1 and X_2 are at t_1 and t_2. There are many ways to represent the process/field. Quite often in Gaussian process literature you will find the auto-correlation function like Figure Correlation matrix diagram. Now $\tau = X_1 - X_2$, the distance between them. Commonly we use squared exponential decay.

$$R(\tau) \propto e^{-|\tau|^2} \tag{4}$$

As τ increases, the correlation decreases. Some people refer it as a Gaussian correlation function. Squared exponential correlation function is commonly used in modeling Gaussian process. For academic purposes, or if you don't know anything about the correlation function you may use it. Actually it takes the form of

$$R(\tau) = c_1 e^{-c_2|\tau|^2} \tag{5}$$

and from the data we calculate the value of c_1 and C_2. We'll learn this in the later part of the semester when we'll learn Gaussian Process modelling. In that case, c_1 is referred as the process noise and c_2 is the length scale parameter. c_1 is just like a variance term, and c_2 is telling you how fast the correlation function is decaying. This is the brute force method to do Gaussian random process simulation. However, this process is very expensive and explode very fast. Here comes an alternate way to simulate Gaussian random process. We take the covariance matrix and compute the eigenvalues and eigenvectors. We select the top 2-3 components and ignore everything else. This is referred as 'Principal Component Analysis'.

2.2 Principal Component Analysis

The same concept is referred to different names in different fields. A more general case in mathematics is called Singular Value Decomposition (SVD). For example in signal processing you may have data for 1000 corresponding to 10 signals. In that case the covariance matrix is rectangular. In case of PCA we are restricted to use square matrix. For a $n \times n$ matrix we have n eigenvalues. Out of those we might select 2 or 3. Consequently, I have a much smaller matrix to use in doing the simulation. A continuum version of PCA is called 'Karhunen-Loeve Expansion' (K-L Expansion). When we implement it we implement it discretely.

Any function $w(X, \theta)$ can be represented as a series:

$$w(X, \theta) = \bar{w}(X) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\, \xi_i(\theta) f_i(X) \tag{6}$$

X represents the coordinate for the random process/random field. The mean is not necessarily zero or constant. This could also be trend function/ mean function. λ is the eigenvalue, f_i is the eigenvector, ξ is the random variable, If you use 3 random variables then you have ξ_1, ξ_2 and ξ_3. So the number of random variables being used to represent the random process/field is 3. θ is just an index. Suppose in an actual problem 10 different quantities, and each one is a random process then $\theta = 10$. We can ignore θ for time being. The size of the correlation matrix will come from the discretization of the random variables. K-L expansion is primarily used for Gaussian random process. In this case ξ-s are Gaussian. These ξ-s are standard normal random variables. Let's consider 2 components, the expression will be

$$w(X) = \bar{w}(X) + \sqrt{\lambda_1}\xi_1(\theta)f_1(X) + \sqrt{\lambda_2}\xi_2(\theta)f_2(X) \tag{7}$$

We are randomly sampling ξ_1 and ξ_2 from independent standard normal distribution. Once the eigenvalue analysis is done λ_1, λ_2, f_1 and f_2 are fixed. $\bar{w}(X)$ is a deterministic mean function. w is your random process of interest. One set of ξ_1 and ξ_2 will give you a realization. The other set of ξ_1 and ξ_2 will give you another realization. So we are mapping to the eigen space and picking 2 or 3. This is how generate the realization of the random process/field. If it is non-stationary the $\bar{w}$ will be different at different location; and the correlation matrix will no longer be the function of distance rather will be function of actual positions.

SRV-Non-Gaussian process For non-Normal we can find out the equivalent Normal. Fill up from class notes.

This method doesn't work well for non-stationary random processes. K-L expansion works very well for Gaussian stationary process, and works reasonably for non-Gaussian stationary processes. If you want to use non-Gaussian non-stationary, this approach doesn't work well. For non-stationary processes we will use 'Polynomial chaos' and 'Auto Regressive Moving Average' (ARMA) methods.

Reading materials provided in the class:

1. Theory of Random Processes
2. K-L expansion PDF
3. 'Collocation-based stochastic finite element analysis for random field problems' by Shuping Huang, Sankaran Mahadevan, Ramesh Rebba cite
4. 'Simulation of Stochastic Processes' by Ramesh Rebba

Jan 15, 2019

One of the simplest ways to represent random process is using Fourier series. Any function can be decomposed into Fourier series, means the basis is a sinusoidal wave. We represent the weighted sum of them. In the following equation the phase angle ϕ is the random variable. Each of the sine functions could have a different phase angle, that means they all start at same zero.

$$g(t) = \sqrt{2}\sum_{k=0}^{N-1}\sqrt{2S_{gg}(\omega_k)\Delta\omega}\,\cos(\omega_k t + \phi_k) \tag{8}$$

First you get the auto-correlation function. Then you convert it into the power spectral density. In the power spectral density you observe the frequency ranges. Suppose your PSD looks something like Figure Add figures from class notes You may take small segments however you want. For each frequency there is a sine function or cosine function with some random phase angle. You discretize the frequency spectrum into a number of frequencies. All the ϕ_k-s are random. Suppose I get a PSD and I discretize into 20 frequencies. That means I have 20 random variables. So I generate random sample of 20 ϕ_k-s to get one realization of the random process. Similarly, if I get another 20 random samples of ϕ_k we can get another realization of the random process. This is the simplest way to do random process simulation. This is quite often used in the earthquake simulation, where you decompose a signal into Fourier series. That gives the PSD and then we discretize that to get the realization of the random process g. There is a similar expression below:

$$g(t) = \sum_{k=0}^{N-1} \sqrt{2S_{gg}(\omega_k)\Delta\omega}\ [U_k\ \cos(\omega_k t) + V_k\ \sin(\omega_k t)] \tag{9}$$

Instead of ϕ here U_k and V_k are the random variables, which are obtained from standard normals. So there are couple of ways in doing random process simulations. This is very effective for stationary Gaussian process simulation. K-L expansion also we use for stationary Gaussian random processes. In K-L expansion we represent the random process through the eigen decomposition. You take the auto-correlation matrix and find the Eigenvalues and Eigenfunctions.

$$g(t) = \bar{g} + \sum_{i=1}^{n} \sqrt{\lambda_i}\ f_i(t)\xi_i \tag{10}$$

So here, ξ-s are the standard normal random variables. This is a much more efficient way of doing it than Fourier series. Because here with 2-3 terms you can capture much better the random process. The reason is that when you are doing the eigenvalue decomposition you are identifying the principal directions. That means the dominant directions are identified. In the methods PCA, active sub-spaces we are trying to identifying the principal directions of that function.

This are all methods that focus the central tendencies, pretty much first and second moments. They are not as good when you are looking at upcrossings. These methods perform well near the mean, not the tail end of the distribution.

Fredholm integral equation:

$$\int_D C(t_1, t_2) f_i(t_2) dt_2 = \lambda_i f_i(t_1) \tag{11}$$

Separate document about detail of K-L expansion.

The basic concept is that we are representing any random function into a basis and some coefficients.

get the basis list from Dr. Maha's slide

Today's focus are (1)ARMA method and (2) Polynomial chaos.

2.3 Auto Regressive Moving Average (ARMA)

This is particularly useful for time dependent random process simulation. Gaussian random process we quite often use to represent spatial random fields. Let's say spatial distribution of material properties or media properties, they don't vary that much. They are fairly stationary in the sense that the mean values remain the same. Though in fluid flow that may not be the case. There you have variation across both space and time. ARMA methods are frequently used in forecasting (weather forecasting, financial markets)-any time series forecasting.

Let's say we have a time series. The coordinate is time instead of space. Basically we can represent the $y(t)$ as some function of the values of the previous time steps. We can include any number of time-steps we want.

$$y_t = b_0 + b_1 y_{t-1} + b_2 y_{t-2} + \cdots \tag{12}$$

This is called AR model. Auto regressive because the variable is regressed on its own previous values. This is similar to regression analysis. Suppose we've used previous 2 time steps, then there is off-course a regression error.

$$y_t = b_0 + b_1 y_{t-1} + b_2 y_{t-2} + \epsilon_t \tag{13}$$

That error is also accumulating. To avoid that we can use the following:

$$\epsilon_t = c_0 + c_1 \epsilon_{t-1} + c_2 \epsilon_{t-2} + \cdots + \epsilon \tag{14}$$

This is referred to as a Moving Average (MA) model. The term 'moving average' simply refers to the fact that you are accounting for the errors in the previous time steps. It's like an averaging operation, when you take averages you don't lose any error information. Now you have a fairly good prediction. As a regression we'll solve b-s and c-s. Once you have the model developed, you can use the model to simulate values of random process. The coefficients are all fixed. Take the previous values. There will be an initial error, ϵ, which can be taken as Gaussian White Noise with standard normal variable. In regression, the general assumption is that the residuals are zero mean Gaussian. That's the underlying theory of basic regression analysis. This will work fine for stationary random processes. If it is non-stationary then we have do one more step. The method needed is ARIMA. The "I" stands for 'integrated'. Here we use 'differencing', we don't use raw values. We look at the difference between the previous and current value.

$$\Delta y_t = b_0 + b_1 \Delta y_{t-1} + b_2 \Delta y_{t-2} + \cdots \tag{15}$$

Wikipedia has a very good page on ARMA and ARIMA. Pages are not good for Polynomial chaos or K-L expansion. Wikipedia also described the standardized functions used in different software like MATLAB, R, Python etc. Here are the links for wikipedia page:

1. ARMA: https://en.wikipedia.org/wiki/Autoregressive-moving-average_model

2. ARIMA: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

ARIMA is the most general one. The order is represented as AR(2), ARMA(2,2)
ARIMA(p,d,q) Instead of modeling the function we are modeling the rate of change of the function. 'Seasonal differencing'.

Paper: Fatigue life prediction of mechanical system: MSSP paper

One question comes that "How many time steps back should I go? What order I should choose?" Quite often this is a trial-and-error process. Typically we use 2 or 3 different competing models. Based on the limited data we don't know which model is better. Then as we collect new data we use Bayesian updating of the weights of the different competing models. There are some special cases in ARIMA Wikipedia page.

For choosing the order there are some standard hypothesis testing based approaches, model selection approaches. These will be discussed later.

2.4 Polynomial Chaos Expansion

Polynomial Chaos representation of a random process or random field is another parametric realization. You have some basis function and some coefficients.

$$f = \sum_{i=0}^{N_0} U_i \Gamma_i = F_0 + F_1 g + F_2(g^2 - 1) + F_3(g^3 - 3g) + F_4(g^4 - 6g^2 + 3) + \cdots \tag{16}$$

These are Hermite polynomial. They are referred to as 'chaos polynomial' or 'Weiner polynomial'. The term 'chaos' comes because these were used to solve partial differential equations that would have some chaos behavior. Weiner used Hermite polynomial. Later it was generalized to many other type of polynomials. A random process has some statistics, i.e. mean, variance. We have to try different coefficients and see which coefficients give us the statistics we are trying to simulate. We are matching the statistics by choosing the coefficients.

There are different kinds of polynomials Dr. Maha's slide. For example, if your random process is Gaussian then typically we use Hermite polynomial. If your random process has Uniform random variables then we use Legendre polynomial. This is referred to as ASKEY scheme. Generalization of Hermite polynomial into different processes. Underlying random variable: If we take any instance and if this was Gaussian process, then all the realizations would be consisted of Gaussian random variables. If they all substitute a Uniform random variable then you use Legendre polynomial to represent the random process. Then you estimate the coefficients by matching the statistics.

This method is nothing different, conceptually, from a nonlinear regression. What choices we have? (1) least squares, (2) maximum likelihood or (3) Bayesian regression. Basically we are trying to fit a known function through some coefficients. Because of the orthonormality of the basis, the quadrature points for these polynomials are theoretically and numerically well established.

Our objective is to simulate random process. That's the bottom line. That means we want to simulate multiple realizations of random processes. That means we have to simulate some random variables, and use random variables to generate realizations of the random process. The simplest method is the simulate random variable at each location. Then that becomes very cumbersome as we need lots of random variables to simulate the random process. So we want some efficient ways of doing this.

$$Y = \Theta^{\intercal} H(\xi) + \epsilon_{\text{su}} \tag{17}$$

We are trying to use a low dimensional representation of a random process for simulation purposes. Truly a random process is a collection of infinite number of random variables. I'm trying to represent this with a small number of random variables to do my simulation. Our objective is to simulate the input, pass it to our model to get the output. Some of your inputs are varying over space or time. That's what we have to deal with.

Paper: 'Collocation-based stochastic finite element analysis for random field problems' by Shuping Huang, Sankaran Mahadevan, Ramesh Rebba

There are 2 things. (1) Order of the polynomial, (2) number of variables. Beam example from the paper was discussed. In the paper Eqn. 13 is 2nd order and Eqn. 14 is 3rd order. Correction in Eqn.13 $1,\ \xi_1,\ \xi_2,\ \xi_1^2 - 1,\ \xi_1\xi_2,\ \xi_2^2 - 1$

$$y = b_0 + b_1\xi_1 + b_2\xi_2 + b_{11}(\xi_1^2 - 1) + b_{22}(\xi_2^2 - 1) + b_{12}\ \xi_1\xi_2 \tag{18}$$

This is a 2nd order Hermite polynomial based PCE. We have 2 variables ξ_1 and ξ_2.

For a non-stationary Gaussian random field then K-L expansion will not work. So we've to fit a polynomial chaos using Hermite polynomials. If it is a non-stationary, non-Gaussian process then I have to see if it is Uniform. If it is uniform then we'll use Legendre polynomials.

2.5 Summary of Random Field or Random Process

Random field representation

- Gaussian vs. non-Gaussian
- stationary vs. non-stationary

Jan 22, 2019

Homework problem discussion. Add the figure here. we have 3 input variables, L is constant. The model is given by:

$$y = \frac{PL^3}{3EI} = \frac{CX_1}{X_2X_3} \tag{19}$$

Now if we want to do a Polynomial chaos to approximate this function and we use Hermite polynomial. We use 2nd order, 3 variables. We should try using 3rd order also to compare the results. In a PCE model we has 2 descriptors:

1. Dimension, i.e. number of inputs
2. Order, i.e. order of polynomial

This is kind of glorified regression model. In any regression model we have basis and a coefficient. In the paper the problem is not a point load, it's a distributed load. We are assuming the property of the beam is a Gaussian Random Field, which has exponential covariance function. Simulating that input using K-L expansion was the 1st homework problem. We simulate many realizations of this input random field. Then we generate the training points to finding out the outputs. In FE analysis, suppose we divide the beam into 10 segments. We can obtain some realizations of EI. In any of these segments we can take one value from one of the realizations to put for in FE analysis. There are several methods to do that. People use 'midpoint method', 'Spatial averaging method' (where we average over the window) or 'Peak value method'. In a straightforward way we can say that each of these beam elements will have different EI based on the realizations. Then we run the FE analysis to compute the displacement of the beam. But in the homework problem we were asked to get the displacement using PCE instead of FE analysis. For example, we have expressed the random field into 2 random variables in terms of ξ_1 and ξ_2 as:

$$EI(x,\theta) = \overline{EI}(x) + \sum_{i=1}^{2} \sqrt{\lambda_i}\ \xi_i(\theta)\ f_i(x) \tag{20}$$

Now the output tip displacement y gets expressed in terms of second order Hermite polynomial using ξ_1 and ξ_2 as:

$$y = a_0 + a_1\xi_1 + a_2\xi_2 + a_3(\xi_1^2 - 1) + a_4\ \xi_1\xi_2 + a_5(\xi_2^2 - 1) \tag{21}$$

We have ξ_1, ξ_2 and y. If we have enough data we can do regression to get the coefficient a_1 to a_5. So we get the Polynomial chaos surrogate model in terms of random variables that express the input random field. In case of the homework, to make it convenient we may use 3 random variables ξ_1, ξ_2, ξ_3 for 3 input random variables. Because our training points are arranged in that way. But it is not necessary.

2.6 Time dependent reliability

In last class we studied ARMA and ARIMA model. These models are used for time series models, where we try to forecast models in time. These are also regression models. In case of using fluctuating demands with time , e.g. loading in a bridge, traffic loading on bridge or fatigue loading on an aircraft component, the input is not static, but time dependent. Add figure Based on previous classes we know how to simulate the processes. There are several approaches:

1. **Equivalent Static Approach:** We are not going to consider all variations over time. We don't have to use time explicitly, so these are based on Extreme Value Distribution.

   - Type I: If it is a Gaussian Process then Type I distribution can be used to model the distribution of the extreme value. In the field of hydrology Gumbel distribution is used to represent Type I largest.
   - Type II: If it is a lognormal process. This is Frechet distribution.
   - Type III: If it is a bounded process, e.g. a minimum value at which something will fail. A special case of Type III is Weibull distribution.

   Type I and II are used for modeling the loads or demand on the system. That's why we look at the largest value. In Type III we use the capacity of the system, so we look at the smallest value. Just looking at the distribution we replace the time dependent with a single random variable. For example in the beam problem, the displacement of the beam is obtained considering the force P as an extreme value distribution. If P is the time dependent load I don't explicitly consider the time variation. But I just replace with one extreme value distribution. Then the problem becomes a static problem. This is commonly done in engineering to avoid the complicated calculations. This is a very simple method. The drawback of this method is that it can answer only one type of question, e.g. "Over the lifetime of the system what the probability that the maximum value will exceed some threshold?" or "Over the lifetime what's the probability that the minimum value will be less than some threshold?" Weibull tested fibers and steel bars under tensile load and was seeing at what value they fail. It was one type of testing. Another type of testing was that he was applying fatigue load on the system to see when do they fail. He also set up a distribution for 'time to failure'. Study of this 'time to failure' became very popular in the field of Industrial Engineering. But in a deeper sense this raised the question that if it system does not follow a nice process is there a way to predict the 'time to failure'. This question lead to the outcrossing based approach.

2. **Outcrossing approach:** Add diagram In this method we say that, if we have a threshold how often does it cross the threshold. If we look at the random process realizations then we can count. Another way of thinking this is "When is the first time it will pass?" That is referred to as 'first passage problem'. Even we can ask the question "What is the average interval between crossings?" The outcrossing event probability is a 'Poisson distribution'.

$$P(N = n|\nu, t) = \frac{(\nu t)^n e^{-\nu t}}{n!} \tag{22}$$

   where, ν = Average outcrossing rate
   That means the outcrossing are 'memoryless'. These are all independent arrivals. There is no correlation between first arrival and second arrival. That's why it is also called as 'Markov Process'. Probability of n outcrossings in t duration. In reality we simulate the random field and get the average of outcrossing rate. Nobody is going to give us an outcrossing rate.

   In a real system what we do is that we assume the loading is following a Gaussian Process. So we'll use K-L expansion first to generate the realizations of the Gaussian Process. Then for each realization we count the number of outcrossings over a certain time period. Then we get an average outcrossing rate. That's how we get our ν.

If $T = t$ is the time of occurrence of first failure/ first outcrossing, then we can calculate the probability as follows:

$$P(T = t) = e^{-\nu t} \tag{23}$$

until time t the occurrence $N = 0$. Typically we like to write the probability distribution as a CDF. Hence,

$$P(T \leq t) = 1 - e^{-\nu t} \tag{24}$$

If we see correlation then we need to do more in addition to Poisson distribution. Outcrossing rate may not be constant all through. Sometimes we may have more frequent outcrossings in some region and less frequent outcrossings in some region. In that case we divide that up into regions. The outcrossing rate is also a function of time. Then we can integrate it over the duration of interest.

$$p_f = 1 - R(0)\exp\left\{ - \int_0^T \nu^+(t)\ dt \right\} \tag{25}$$

where $R(0)$ is the initial reliability. It may not be always be 100%.

There are some systems which is gradually degrading, like a crack growth. Those cases are straight forward.

There are theoretical derivations in the literature on how to calculate the outcrossing rate. Threshold need not be constant.

How to make sure the simulated random fields are correct?

Suppose the outcrossing rate is known. There are published data on outcrossing rate, reliability rate, MTBF (Mean Time Between Failures). If we want to match the extreme tendencies, instead of matching the first or second moments, it's better to match the outcrossing rate.

How to tune it?

What are the parameters we are using to generate the realizations. For a Gaussian random process we started with a correlation function. l is the length scale parameter, σ is the process noise and the mean value. So we usually tune these 3 parameters to match the realization. We can do it using 'least squares' or 'Maximum Likelihood' or Bayesian way. In case of K-L expansion we can tune the number of eigenvalues we are going to use.

Jan 24, 2019

2.7 Stochastic Finite Element Method

Add picture The model is a FE model. The inputs could be random variables or random processes or random field. They could have variation across time or space and variation across samples, which is what random variable is. Stochastic finite element refers to the situation where our model is a finite element model. The concern there is considering spatial variability. Stochastic finite element can be both for properties and loads. The loading could be some kind of random field. So the loading is also different on each element. The challenge is how do I represent this spatially varying random field either in properties or in loads in our FE analysis. Here we have to characterize the randomness using random filed or random process. The next step is to simulate the random field/process. That simulation is incorporated in the FE analysis. There are 2 ways of doing this:

1. Intrusive approach: Intrusive approach goes into the finite element code and calculate quantities, e.g. how the stiffness matrix changes. Suppose we have some function $y = g(x)$. We can approximate it using first order Taylor series as:
$$y = g(\mathbf{x}) + \sum \frac{\partial g}{\partial x_i} x_i \tag{26}$$
In the intrusive method we need to find the analytical gradient.

2. Non-intrusive approach: Non-intrusive approach treats the code as a black-box. In case of non-intrusive method we obtain the gradient using 'adjoint method'. In finite element we use $KU = F$ where F is the demand/force on the system, U is the response/displacement on the system and K is the properties of the system/ stiffness. We solve the response as $U = K^{-1}F$. The most expensive part is the inverting K. In the adjoint method we take the advantage of the inversion of K. Some change in force yields to:
$$K(U + \Delta U) = (F + \Delta F) \implies U + \Delta U = K^{-1}(F + \Delta F) \implies \Delta U = K^{-1}(F + \Delta F) - K^{-1}F \tag{27}$$
In this method we use adjoint gradient. In dynamic response of aircrafts, for aerodynamic loads inverting this K is far more complicated than for civil structures. The other non-intrusive method is surrogate model. We just replace the complicated model with a surrogate. The surrogate is so cheap, we'll just running brute force Monte Carlo simulation for millions of times we can get the information.

As the computers were getting faster and faster surrogate modelling is the most beneficial and practical to do. There are so many good theories related to stochastic finite element. One of the challenges is representing the spatial randomness of the loads and properties. We can represent this by certain number of elements spatially discretizing it. When we have very complicated geometries it becomes harder to take care the randomness. That's why we go to a 'Spectral representation', where we take the eigenvalue and eigenvectors. Then we use top 2-3 eigenvalues and corresponding eigenvectors to represent random process or random field; rather than spatially discretizing it. There are some other problems. If we discretize it too fine then 2 adjacent points will have very high correlation. The correlation matrix will become ill-conditioned. On the other hand, if we discretize it too coarsely then we are being very approximate. So we run into challenge between accuracy and numerical feasibility. Stochastic finite element is now driven by surrogate modeling. One of the first thing came with SFEM is the polynomial chaos. 'Spectral Stochastic Finite Element Method' where the word 'spectral' comes from spectrum of different frequency sinusoidal function.
A very good review paper: "Stochastic Finite Element Methods and Reliability:A State-of-the-Art Report" by Prof. Bruno Sudret.

3 Surrogate modeling

K-L expansion is one way to generate input and polynomial chaos to do surrogate model. add picture Pushing some distribution into the model to get some distribution is uncertainty propagation. This model could be quite expensive. So we would like to replace this model with a less expensive model, i.e. inexpensive replacement. In early days this was called as "response surface approach". The national labs use the name "Emulator". Now there is another big buzz word called "machine learning". Surrogate model is supervised learning. Surrogate model is not 'Uncertainty Quantification'. 'Uncertainty Quantification' address something about epistemic uncertainty. Either the model is uncertain, or the data is uncertain or both. Two types of surrogate modeling below:

1. **ALGEBRAIC SURROGATES:**
   The generalized mathematical form is $y = g(x)$.
   - Regression: There are 3 methods (1) least squares, (2) linear/polynomial (3) maximum likelihood
   - Polynomial chaos (different kinds of basis leads different coefficients)
   - Radial Basis Function (RBF)
   - Support Vector Machines
   - Gaussian Process/ Kriging models
   - Neural network: This has multiple layers.
2. **DISTRIBUTION SURROGATES:**
   $f_{\mathrm{xy}}(x, y)$ joint PDF/ distribution of input and output
   - Copula based models
   - Mixture models: Most commonly Gaussian mixtures
   - Bayesian networks
   - Hidden Markov Models

A conceptual point: Somebody makes the model and collects the data and gives to the analyst to fit the data. That's not the best approach. The best way is to have an interaction between the modeler and experiment person. This is because of the fact that we'll generate training data to train our surrogate model. Given that our resource are limited we want to generate training data in a more efficient way. That depends on what kind of model we are planning to use. For each of these training models the optimum training data will be different. That's why we need a back and forth feedback. Quite often what we do is "Adaptive" selection of the training points. Now a days that is referred to as 'Active Learning'.

Jan 24, 2019

3.1 Algebraic surrogates

3.1.1 Gaussian Process modeling

It is heavily used in Machine Learning. Add picture from class note. First we talk about basic regression, least square we did in '*Probabilistic Methods in Engineering Design*' course . If we have input-output and some data, we fit a least squares or regression model like in Figure. In that case the assumption is that the residuals are normally distributed with zero mean and some standard deviation. Mathematically that is represented as:

$$y = b_0 + b_1 x + \epsilon \qquad \epsilon \sim N(0, \sigma^2) \tag{28}$$

The coefficients of regression are obtained as:

$$\mathbf{b} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{Y} \tag{29}$$

Conceptually we are minimizing the length of the error vector using least square method (sum of square of error). This error is a single normal variable and these are different realizations of that normal variable. One single normal variable represent the residual of that fit. Add the comparative result diagram here.

When we have situation like Add figure. The residuals have a spatial variability. In Gaussian Process regression we represent the residual has a mean, a variance and some covariance. In GP the error is represented as:

$$\epsilon \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{30}$$

where, $\boldsymbol{\Sigma}$ is the covariance matrix. GP allows for different realizations, where at each point we have Gaussian variable. We need to know the mean, the variance and the correlation function. If we have a trend function which is already learned, then we can use the mean as zero. That means the trend is taken into account. The covariance will be represented as:

$$\boldsymbol{\Sigma}_x = \sigma^2 \rho(x) = \sigma^2 \exp\left[- \left(\frac{\Delta x}{l} \right)^2 \right] \tag{31}$$

where, σ is called process noise. There are many other possible selections for the correlations function. Δx is the distance between 2 points and l is the length scale parameter or correlation length. Add the picture of squared exponential correlation function. We don't know the length scales. We have to learn the length scale parameter also. Similar to regression, here we have to learn the trend function and then the parameters of GP. In case of multidimensional problem, e.g. X_1 and X_2, where the variation of correlation is different (e.g. one slow decay and the other decays faster) we have to use 2 different length scale parameters l_1 and l_2. Here we have to learn not only the regression coefficients but also the length scale and variance parameters for the GP residual. Instead of doing all in one shot, the practical process of doing it is do a simple regression. That will give us the 'Trend function'. Then take the residuals and learn the parameters of Gaussian Process. In some literature we also find as follows:

$$\sim \mathrm{GP}(\mu(x), \Sigma_x) \qquad \text{or} \qquad \sim GP(b_0 + b_1 X,\ \boldsymbol{\Sigma}_X) \tag{32}$$

In simple regression we use same random variable everywhere. So the variance doesn't change; and it is perfectly correlated. In GP we have different variances in different places. Assuming the trend function is taken care of we focus on the residual.

Prediction:

What we predict with a simple regression model is a mean line or trend line. There is a residual still there. Mathematically we should write it as:

$$E(Y) = b_0 + b_1 X \qquad \text{or} \qquad Y = b_0 + b_1 X + \epsilon \tag{33}$$

Similarly, in GP model we have the mean function. the function has a variance which is varying at different locations. For data-driven model we may say it as 'model uncertainty'. In '*Probabilistic Methods in Engineering Design*' course we discussed multivariate distribution. When we have multi-normal distribution (i.e. all the variables are jointly normal) the mean and the covariance matrix define that distribution. There is a great property for multi-normal distribution: **If a set of variable is jointly normal, then any subset of variables is also jointly normal**. The marginals are normal.

Meaning of GP:

It means every cut we take is a Gaussian variable. The question comes is "how many variables are there?" The answer is "infinite". But we consider a finite number when simulating. A Gaussian Process is a multivariate normal distribution in infinite dimension; which means any subset is also normal. Now the math comes into picture. Equations from slide will be here.

My mean prediction is given by the m. This expression is mathematically as Equation (29). The variance is given by S. These are prediction formula. So the question 'how to learn is done'. We can use methods like Markov Chain Monte Carlo (MCMC) or MLE.
Maximum Likelihood Estimation:
The expression is written as:

$$\log p\ (\ y_T|\ x_T;\ \theta) = -\frac{1}{2}y^{\intercal}(K_{TT}+\sigma_n^2\mathbf{I})^{-1}y - \frac{1}{2}\log|K_{TT}+\sigma_n^2\mathbf{I}| + \frac{D}{2}\log(2\pi) \tag{34}$$

K is the covariance matrix, which is calculated using assumed correlation function. The unknown l is in the K matrix. So we'll maximize the likelihood function to estimate l.
Maximum Likelihood Method to Estimate the Parameters of Regression Model:
Likelihood is probability of observing data given the model prediction. The equation is presented as:

$$L(\underline{b}) = P(y_{\text{obs}}|\underline{b}) = P(y_{\text{obs}}|b_0+b_1x) = P(y_{\text{obs}}|y_{\text{pred}}) \propto \text{ PDF of } \epsilon \tag{35}$$

In this case the model prediction $y_{\text{pred}} = b_0 + b_1x$. The idea of proportionality comes from the distribution of error, which is coming from normal distribution with zero mean and some variance. It can also be expressed as: $\epsilon = (y_{\text{obs}} - y_{\text{pred}}$. As this is a continuous variable, probability of observing an actual value is zero. Then we take a small strip from the distribution as a probability value. The value comes from width $\times$ PDF value. As we can select any arbitrary width, this is proportional to the PDF of ϵ. Take our model with some assumed value of coefficients, for a given input we get the predicted value. If we have the data, we have observed value. We can predict based on some assumed value of the parameter. We have to find the parameters and maximize the likelihood function.

If we consider a two dimensional Gaussian Process the correlation function will look like:

$$\exp\left[-\frac{r_1^2}{l_1^2} - -\frac{r_2^2}{l_2^2}\right] \tag{36}$$

In one dimension it may vary slowly and in other dimension it may vary fast. If it is isotropic then we can consider the l to be same. Add diagram of sausage plot related to GP interpolation and regression.

There are 2 ways to do this.

1. **GP interpolation:** If our data is point value, e.g. if it comes from a computer code, it's a precise value. In case of simple regression our bound was parallel to the mean. Now the variance changes from point to point. These are referred to as 'sausage plots'. The variance at the training point is zero as there is no uncertainty. In between 2 training points our prediction is uncertain. Further away from a point it grows and then it shrinks.

2. **GP regression:** On the other hand, we may have noisy data. The data is some interval value. In that case we call it Gaussian Process regression. They will shrink closer to the data point, but they will not go to zero. This is the great thing about GP model, that we know the uncertainty of the prediction.

Adaptive Kriging: Story of Krige and oil well modeling. Spatial statistics. Weighted combination of known data.

$$y^* = \sum w_i y_i(x_i) \tag{37}$$

Programming aspect:

- MATLAB: Use the function `fitrgp`.

- Python: Use machine learning library `scikit-learn`. Nice tutorial by Prof. Chris Fonnesbeck (Assistant Professor of Biostatistics, Vanderbilt University Medical Center) is available at : `https://blog.dominodatalab.com/fitting-gaussian-process-models-python/`.

Jan 29, 2019

Discussion on homework. If the function `fmincon` is not able to converge then we can use `fminsearch` or `fminunc`. There is another way called `DIRECT` algorithm, which means 'DIvding RECTangle'. It fits a quadratic approximation to the function. This is called 'Sequential Quadratic Programming' (SQP). It is robust and converge to the 'global optimum'. Compared to the Newton based methods (which goes straight by gradient, so fast) DIRECT is slow though. All these `fmin`$\cdots$ methods are based on Newton method or gradient projection. That's why they could stuck in local optimum. We have to figure out what function has replaced the function of 'DIRECT' algorithm?
Picture from Hombol's paper-"Bias Minimization in Gaussian Process Surrogate Modeling for Uncertainty Quantification" Cite the paper
In this case we have a function which is smooth in one region and noisy in another region.One sigma bounds are in grey color. The mean prediction green, true function black. But there is a lot more uncertainty in the prediction. So we have 3 notable regions (1) large variance, small error, (2) small variance, small error and (3) small variance, large error. Moral of the story: When we talk about a surrogate model we have to think about the errors in the surrogate model. The uncertainty of the surrogate model prediction has 2 aspects to it:

1. Bias
2. Variance: For GP it's directly available from the mathematical expression. In regression we also have the variance.

These two together contribute to the uncertainty of the surrogate model. The mean line is the 50% prediction. The residual has a zero mean and some variance. At the training point there is no bias and variance. To identify the bias we have to do 'cross-validation'. Typically we use 80% data to train the model and rest 20% to cross-validate the model. That cross-validation is done in multiple shuffles. The 'bias 'itself is a random variable.

Suppose we have surrogate model prediction y_s and surrogate model error ϵ_s then the model prediction y_m will be

$$y_s + \epsilon_s = y_m \qquad \text{where} \ \ \epsilon_s \sim N(\mu_{\epsilon_s}, \sigma_{\epsilon_s}) \tag{38}$$

in which, μ_{ϵ_s} is the bias and σ_{ϵ_s} is the variance. For the original model:

$$y_m + \epsilon_m = y_{\text{true}} \tag{39}$$

We can develop surrogates fro physics based models, e.g. a differential equation to describe a system based on some physical assumptions. In many of our cases the physics based model is very expensive so we replace it with a surrogate. In case of machine learning we only have data. Then we derive the surrogate directly from data and $\epsilon_m = 0$.
Paper: Nannapaneni, S. and Mahadevan, S., "Reliability Analysis under Epistemic Uncertainty", 2016. cite

Based on different training point we'll get different variance. For multiple splits we'll get multiple μ and σ. Then we average out them. That's why they are called 'hyper-parameters' ,i.e. parameters of parameter. If we have a few data points, e.g. 9-10 we use LOOCV (Leave One Out Cross Validation).

3.1.2 Radial Basis Function

Different regression models have different basis. Similarly the radial basis function is a regression model where the basis functions are of the form $\beta_i(X - X_i)^2$. This has similar effect as Kriging. β_i is called the 'estimate parameter'. A slide

$$\sigma_{\text{RBF}} = \Phi_{\text{PT}}(\Phi_{\text{TT}} + \sigma_n^2 \mathbf{I})^{-1} y_T \tag{40}$$

If the covariance function is isotropic, i.e. same in all direction, then we call it radial basis function. In multidimensional case we consider a circle. It is used to measure the decay of one point.
Kernel density: In case of Kernel density we use the same idea. It is used to fit a distribution numerically from available samples. In '*Probabilistic Methods in Engineering Design*' course we have discussed about finding distribution from a set of samples. But there we try out known distribution forms, i.e. parametric distributions. If the data is quite complicated we do a numerical fit. One of the ways to do numerical fitting is using 'Gaussian kernels'. In this case our basis function is Gaussian functions, centered at some point. At each of the points we have PDF values which contribute to the weight. No regression involved in this process. In MATLAB we use `ksdensity` to fit a distribution to the sample. It usually use Gaussian kernel.

3.1.3 Neural network

Let's assume we fit a regression model $y = b_0 + b_1 x + \epsilon$. Then the $\epsilon = c_0 + c_1 x + \epsilon\prime$ If we keep doing this we create multiple layers. Finally, we have to estimate all the coefficients. This is how layered regression model works. Neural network is a regression model with multiple layers. Example diagram. If we have multiple basis functions and multiple layers, then number of parameters to estimate becomes very large. That's why we need lots of training data to do neural networks. If we have large number of data to train the neural network, this method will beat all other algebraic surrogate models. In neural network we use sigmoidal functions. Each of this function is capturing one kind of trend. If the trend is very complicated it can be broken into linear, quadratic trends etc. depend on where what trend is dominant. If we keep increasing the number of layers we call it 'Deep Neural Network' (DNN). In ARMA model we tried to fit the previous and current time responses. We also have errors of that fit. Then we have differencing, which is a 3rd layer. We can also increase the number of functions in the regressions and the layers then that would be called as a 'Recurrent Neural Network' (RNN). RNN is a kind of complicated ARMA model.

3.2 Distribution surrogates

In this case we are trying create a joint distribution for the input and output. In case of algebraic surrogate model we have prediction of y given x.

$$y = g(x) \tag{41}$$

This works in the physical space, i.e. physical variables X and y. This is also called 'supervised learning'. In case of distribution surrogate we try to infer the joint distribution from the data. As we try to learn the joint distribution it is called 'unsupervised learning'.

$$f_{XY}(x, y) \implies \text{distribution} \tag{42}$$

If we have multiple variables learning their joint distribution and representing them is complicated. There are only few cases where we can have nice, closed form analytical solution. A best example is multivariate Gaussian. A multivariate Gaussian is completely specified with a mean μ and covariance Σ of the variables.

Another biggest challenge is to get the covariance matrices. In many cases we don't have data to build covariance matrix. If the experts say "I don't know" how should we construct the covariance matrix. So those places are left blank in the matrix. We can't put any arbitrary number. For example, if we put 0 or 1 then also we'll face the singularity problem. So we have to put numbers such a way that the matrix is positive definite. This is a famous problem in statistics called 'Completion Problem'. It is very very difficult to specify the joint distribution of multiple variables.

There are following methods to do that.

1. Copula
2. Mixture models
3. Bayesian Network

These are different level of approximation of the joint distribution.

3.2.1 Copula

In '*Probabilistic Methods in Engineering Design*' course we also used Copula without saying the name. Resource on Wikipedia related to Copula is good.
https://en.wikipedia.org/wiki/Copula_(probability_theory)

The joint CDF is:

$$F_{\underset{\sim}{X}}(x) = C\Big[F_{X_1}(x_1), F_{X_2}(x_2), \cdots F_{X_n}(x_n)\Big] = C\Big[U_1, U_2, \cdots U_n\Big] \tag{43}$$

C is the Copula function, which relates the marginal CDF to the joint CDF. Our aim is to estimate the Copula function. CDF goes from 0 to 1. These U-s are random numbers. add picture In '*Probabilistic Methods in Engineering Design*' course we did that. $u_i = F_{X_i}(x_i)$.

In the Gaussian Copula,

$$\Phi(x, y) = \Phi_{\mathrm{R}}(\Phi^{-1}(u_1), \Phi^{-1}(u_2)) \tag{44}$$

Gaussian Copula will work for some variables, not for all variables.
Note: If individual PDFs are Gaussian, their joint PDF is not necessarily Gaussian. We have individual/marginal CDFs that are Gaussian.

3.2.2 Mixture Models

Mixture model is a weighted sum of the individual components.

$$f_{\underset{\sim}{X}}(x) = \sum_{i=1}^{n} w_i \; f_{X^i}(\underset{\sim}{x}) \tag{45}$$

We'll look at the superscript i. This is also multivariate distribution. For example if we have 2 variable this formula expands to:

$$f_{X_1X_2}(x_1, x_2) = w_1 f_{X_1^1X_2^1}(x_1, x_2) + w_2 f_{X_1^2X_2^2}(x_1, x_2) \tag{46}$$

where, $f_{X_1^1X_2^1}$ and $f_{X_1^2X_2^2}$ could have two different PDF type. The superscript 2 refers to PDF type. It's weighted sum of 2 joint PDFs. Most commonly used is 'Gaussian Mixture Model' (GMM). In this case we take 2 normal distributions with different means and standard deviations. Kernel density is a Gaussian Mixture Model.

Gaussian Copula and Gaussian mixture model are very powerful in Bayesian updating and sensitivity analysis.

Jan 31, 2019

Probability-Space surrogates:

Multivariate Gaussian is a Gaussian of many variables. μ is a mean vector and Σ is the covariance matrix, $\underset{\sim}{\mathbf{x}}$ is the vector of variables. It is the generalization of the univariate Gaussian distribution. p is the number of variables.

$$f_X(x) = N_X(x \mid \underset{\sim}{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\underset{\sim}{\mathbf{x}} - \underset{\sim}{\mu})^{\mathsf{T}}\Sigma^{-1}(\underset{\sim}{\mathbf{x}} - \underset{\sim}{\mu})\right] \tag{47}$$

The covariance matrix has all the correlation information in it. Here, for individual variables we have CDFs, either specified or inferred from the data. As an example let's consider, for an experiment, x_1 and x_2 are 2 inputs and y_1 and y_2 are corresponding outputs. We are trying to infer the joint distribution of x_1, X_2 and y_1, y_2. add the X Y diagram here. The joint CDF of $\underset{\sim}{X}$ is

$$\Pr(X_1 \le x_1, X_2 \le x_2, \cdots) = \Pr(U_1 \le u_1, U_2 \le u_2, \cdots) \tag{48}$$

Since the CDF is a monotonically increasing function $\Pr(X_1 \le x_1) = \Pr(U_1 \le u_1)$. The commas indicate intersection operation. We can express the joint CDF either in the X-space or in the U-space. When we express it in the U-space we call it Copula. Keep in mind that we are talking about only in CDF space which is a uniform from 0 to 1. The underlying distribution can be anything. Copula is the joint CDF in the U-space. Gaussian Copula means the PDF of U-s is a Gaussian of U_1, U_2, i.e. $\Phi(U_1, U_2, \cdots, U_n)$. The joint distribution of U variables is a multivariate Gaussian. We have a bunch of data and we are trying to explain that through some analytical model. The strictest assumption is that all of them are multivariate Gaussian.

At the CDF level we have the probability equivalence. That's why we do inverse CDF. When we randomly sample a number between 0 and 1, and then we equate the probability. In '*Probabilistic Methods in Engineering Design*' course during generating correlated non-normals, first we generated uncorrelated non-normals. Then we went to correlated normals. Then equating the CDF we obtained the non-normals. In case of multivariate Gaussian it assumes that they are jointly Gaussian and individually Gaussian. Copula relaxes that assumption. It keeps the jointly Gaussian assumption, but says individuals can be any distribution. So we can write $\Phi_{\mathrm{G}}(X_1^N, X_2^N, \cdots)$.

In FORM, we took $X_i \longrightarrow X_i^N$. Then we were finding the β distance in that space. Hidden in this assumption was they are jointly normal. In '*Probabilistic Methods in Engineering Design*' course we were taking a Copula approximation of the underlying variables. Equivalent normals are jointly normal.

$$C_{\mathrm{G}}(u_{x_1}, u_{x_2}, u_{y_1}, u_{y_2}) = \frac{1}{\sqrt{\det(R)}} \exp\left[-\frac{1}{2}\begin{pmatrix}\Phi^{-1}(u_{x_1}) \\ \cdot \\ \cdot\end{pmatrix}^{\mathsf{T}} (R^{-1} - I) \begin{pmatrix}\Phi^{-1}(u_{x_1}) \\ \cdot \\ \cdot\end{pmatrix}\right] \tag{49}$$

where, R is the correlation matrix. There are other Copula approximation but they don't have nice closed form analytical expressions. Typically the way we verify is to take the samples and fit a Gaussian Copula to that. Then we have some other samples kept aside. We take the correlation matrix that the Gaussian Copula has specified ; and the correlation of the samples we are testing. Usually correlations are common way of verifying whether it is a good assumption or not.

Colored noise: It is one way of expressing the correlation.

We only use the joint CDF. PDF is more for graphical representation. Mathematically we talk about PDF. But practically to generate samples we use CDF. We use surrogate model to do some forward propagation. In 'Importance Sampling' we say that we generate samples from PDF. But actually we generate from CDF. However, from representation point of view all CDFs look alike. So it is difficult to distinguish between distributions. We use PDFs for the weights, to calculate likelihood. For generating sample CDF will help.

Gaussian Mixture:

It is weighted sum of Gaussian. These are multivariate normals. If we have 10 variables we have 10 dimensional Gaussian. So we use weighted sum of 10 dimensional Gaussians. Each one is a multivariate Gaussian, not individual.

$$f_X(x) = \sum_{i=1}^{N} w_i \; N_{X,i}(x \mid \mu_i, \Sigma_i) \tag{50}$$

An example is $0.4 \times \; N(8,3) + 0.6 \times \; N(14,2)$. Each of those Gaussian have different mean and covariance. We use a 2 step algorithm called 'Expectation Maximization' (EM) algorithm.

- E step: we estimate weights for given values of parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$
- M step: We estimate parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ for given values of weight

How to choose N? We have model selection criteria; e.g. MLE. If we use maximum likelihood we may overfit the data. AIC (Akike Information Criterion), BIC (Bayesian Information Criterion)
BIC $= k \times \ln(M) - 2 \times \ln(L)$ k is the number of parameters, M is the number of training points and L is the likelihood.

3.2.3 Bayesian Network

Another powerful approach. Add network diagram. In probability space surrogate our goal is to get the joint distribution. There is no approximation involved. If we have 2 variables x_1 and x_2, where x_1 is the conditioning variable and x_2 is the inferred variable and there is a direction for us to decompose it. It is called Directed Acyclic Graph (DAG). It's one way. We have decomposed the joint distribution for node a to g as:

$$\Pi(U) = \Pi(a) \times \Pi(b|a) \times \Pi(c|a) \times \Pi(d|c) \times \Pi(e|b,d) \times \Pi(f) \times \Pi(g|e,f) \tag{51}$$

We know what's the cause and what's the effect. That's why Bayesian Network is referred to as a 'Causal Graph'. On the other hand, if there is no cause and effect relationship and the relation could go both ways then we are talking about statistical correlation, not any physical functionality. That is referred as a 'Markov Network'. Any Copula or a multivariate Gaussian network of variables jointly connected are specified by the correlations between them. That could be think of as a Markov method. g has 2 parent nodes, so g has a conditional density function. If it is a continuous variable conditional probability we have PDFs and CDFs. If it is a discrete variable then there are multiple settings for each of those we have Conditional Probability Tables (CPT).

We can do it in different ways. People who deal with Physics based models the flow structure may be know to them based on the knowledge of the system. Once we know the structure we can learn the conditional probabilities from the data. It could be completely driven by data. In Python we have `bnfinder`, to identify the parameters, given the data. Even we can use hybrid methods. With the Gaussian based surrogates the representation is approximate but the inverse problem with Bayesian is exact. Here in inference problem (which is inverse problem) is approximate as we use sampling such as MCMC to infer. Given the limitation of our sampling techniques and computational resources it is approximate. So we have either an approximate representation and exact solution or exact representation and an approximate solution.

Depending on the problems we have to select the surrogate models. If we are interested in joint distribution then we go in this way. If we want to predict the value of the output we do an algebraic surrogate model.

4 Sensitivity Analysis

Sensitivity analysis is performed to know which inputs affect the outputs and how much they affect. Sensitivity analysis is also used to reduce the number of variables. If some variables are not important then we can discard them. If they are uncertain we don't discard them but discard their uncertainty. We fix their mean value or nominal value. There are 2 ways of thinking about sensitivity:

1. Local sensitivity: It's the derivative, like $\frac{\Delta y}{\Delta x}$. That sensitivity is local in the sense that the derivative changes with the value of the x. Add figure.

2. Global sensitivity

In '*Probabilistic Methods in Engineering Design*' course we studied 'Probability Sensitivity Factors'. In FORM, during calculating minimum distance β the gradient vector ∇g had 2 components. These 2 components are the *probability sensitivity factor*.

$$\alpha_i = \left(\frac{\dfrac{\partial g}{\partial x_i}}{\sqrt{\sum \left(\dfrac{\partial g}{\partial x_i}\right)^2}} \right)_{\underset{\sim}{\mathbf{x}}=\underset{\sim}{\mathbf{x}}^*} \tag{52}$$

The derivative we calculate at the minimum distance point. During minimum distance search we did compute the derivative at each point. That derivative is obviously local. add FORM graph. On the other hand, when we are talking about uncertainty analysis (where the inputs and outputs are uncertain) we are more interested in global sensitivity. In case of global sensitivity we measure how much the uncertainty contribute from one point to other. If we have 2 variables X_1 and X_2, each one has some distribution, we do variance decomposition. Add picture The question comes now "how much this X_1 and X_2 contribute to the scatter in Y?" Now we are considering all the contribution of X_1 and X_2 and their dispersion in Y. That's why we call it Global sensitivity analysis.

This global sensitivity is also referred as 'variance based sensitivity'. Variance is one of the measures of looking sensitivity. When the distribution is skewed this type of variance based measures are not good.
Note: If one variable dominates then the central limit theorem cannot be used.

Sobol indices: These are not very different from 'ANOVA' we learned in '*Probabilistic Methods in Engineering Design*' course . They are based on 'Variance decomposition theorem'. Wikipedia is a very good source for that.
https://en.wikipedia.org/wiki/Law_of_total_variance

The main formula is:

$$\mathrm{Var}(Y) = \mathrm{E}[\mathrm{Var}(Y|X)] + \mathrm{Var}(\mathrm{E}(Y|X)) \tag{53}$$

The first term is the variance for fixing one X and obtaining the output. The second term come from the variance of condition to all the mean values of Y. In '*Probabilistic Methods in Engineering Design*' course we used *standard error*. This is what we use to build the confidence bound.

Individual effects (S_I) and total effects (S_T)

$$S_I = \frac{V_{X_i}(E_{X_{\sim i}}(Y|X_i))}{V(Y)} \qquad S_T = \frac{E_{X_{\sim i}}(V_{X_i}(Y|X_{\sim i}))}{V(Y)} \tag{54}$$

S_I is usually less than 1 whereas S_T is typically greater than 1. The way we do it by using double loop sampling method.