Uncertainty Quantification

Perturbation and Monte Carlo Methods

Henrique Kroetz

January 7, 2026

Uncertainty Propagation

We consider a deterministic model $\mathcal{M}$ mapping uncertain inputs $\mathbf{X}$ to an output Y:

$$Y = \mathcal{M}(\mathbf{X})$$

- What is the expected model response?
- What is the variance of the response?
- Which distribution does the response follow?

Approach and Moments

The input vector $\mathbf{X} \in \mathbb{R}^n$ is characterized by:

- Mean vector: $E[\mathbf{X}] = \boldsymbol{\mu}_X$
- Covariance matrix: $\text{Cov}(\mathbf{X}, \mathbf{X}) = \boldsymbol{\Sigma}_X$

Goal: Approximate the moments of Y (mean μ_Y and variance σ_Y^2) without necessarily knowing the analytical form of $\mathcal{M}$.

Part I: The Perturbation Method

(Local Approximation / Moment Methods)

Mathematical Foundation: 1D Taylor Series

For a scalar function $f(x)$, the value at x can be estimated from a point a:

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \dots$$

- **First-order:** Linear approximation (tangent).
- **Second-order:** Quadratic approximation (curvature).

Expansion in Multi-Dimensional Space

For a model $\mathcal{M}(\mathbf{X})$, we expand around $\mathbf{a}$:

$$\mathcal{M}(\mathbf{x}) \approx \mathcal{M}(\mathbf{a}) + \nabla\mathcal{M}^T(\mathbf{x}-\mathbf{a}) + \frac{1}{2}(\mathbf{x}-\mathbf{a})^T\mathbf{H}(\mathbf{x}-\mathbf{a})$$

The Perturbation Method in UQ

In UQ, we expand around the **mean of the inputs** ($\boldsymbol{\mu}_X$). Let $\Delta\mathbf{X} = \mathbf{X} - \boldsymbol{\mu}_X$:

$$Y \approx \mathcal{M}(\boldsymbol{\mu}_X) + \mathbf{g}^T\Delta\mathbf{X} + \frac{1}{2}\Delta\mathbf{X}^T\mathbf{H}\Delta\mathbf{X}$$

Apply $E[\cdot]$ and $\mathrm{Var}(\cdot)$ to find the moments of Y.

First-Order Second-Moment (FOSM)

Truncating after the linear term: **Mean:**

$$E[Y] \approx E[\mathcal{M}(\boldsymbol{\mu}_X) + \mathbf{g}^T \Delta \mathbf{X}] = \mathcal{M}(\boldsymbol{\mu}_X)$$

Variance:

$$\sigma_Y^2 \approx E[(\mathbf{g}^T \Delta \mathbf{X})^2] = \mathbf{g}^T \boldsymbol{\Sigma}_X \mathbf{g}$$

Second-Order Mean Correction

Including the Hessian term to capture curvature:

$$E[Y] \approx \mathcal{M}(\boldsymbol{\mu}_X) + \frac{1}{2}\mathrm{Tr}(\mathbf{H}\boldsymbol{\Sigma}_X)$$

For independent inputs:

$$E[Y] \approx \mathcal{M}(\boldsymbol{\mu}_X) + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial^2 \mathcal{M}}{\partial X_i^2}\sigma_{X,i}^2$$

Numerical Gradients: Finite Differences

If the analytical derivatives are unavailable, we approximate the gradient $\mathbf{g}$ and Hessian $\mathbf{H}$ numerically. **Forward Difference (Gradient):**

$$g_i = \frac{\partial \mathcal{M}}{\partial X_i} \approx \frac{\mathcal{M}(\boldsymbol{\mu}_X + h_i \mathbf{e}_i) - \mathcal{M}(\boldsymbol{\mu}_X)}{h_i}$$

Central Difference (Hessian Diagonal):

$$H_{ii} = \frac{\partial^2 \mathcal{M}}{\partial X_i^2} \approx \frac{\mathcal{M}(\boldsymbol{\mu}_X + h_i \mathbf{e}_i) - 2\mathcal{M}(\boldsymbol{\mu}_X) + \mathcal{M}(\boldsymbol{\mu}_X - h_i \mathbf{e}_i)}{h_i^2}$$

Where h_i is a small step size (e.g., $10^{-5}\mu_i$).

Numerical Stability: The Step Size Dilemma

When models contain variables with vastly different scales (10^9 vs 10^{-5}):

- **Absolute Step:** $h = 10^{-5}$ is too large for $I = 10^{-5}$ (100% change) but too small for $E = 10^{11}$ (below machine precision).
- **Relative Step:** $h_i = \mu_i \times 10^{-4}$ ensures we perturb each variable by exactly the same proportion.

Stability Tip: Use **Central Differences** for the gradient as well. It cancels out the first-order error terms and is less likely to blow up near non-linearities.

Final Project Logic Summary

```
for i = 1:n
    h_i = mu_X(i) * 1e-4; % Proportional step

    % Central sampling
    Y_plus  = model(mu_X + step);
    Y_minus = model(mu_X - step);

    % Numerical derivatives
    g(i) = (Y_plus - Y_minus) / (2 * h_i);
    H(i) = (Y_plus - 2*Y0 + Y_minus) / (h_i^2);
end
```

Check: If Y still returns `Inf`, ensure the model is not dividing by zero during the perturbation!

Numerical Example: Mean Shift

Consider $Y = X^2$, where $X \sim \mathcal{N}(\mu_X = 10, \sigma_X = 2)$. **First-order approximation:**

$$E[Y] \approx 10^2 = \mathbf{100}$$

Second-order correction ($\mathcal{M}'' = 2$):

$$E[Y] \approx 100 + \frac{1}{2}(2)(2^2) = \mathbf{104}$$

Curvature shifts the mean!

Part II: The Monte Carlo Method

(Global Sampling / Statistical Approach)

The UQ Problem Statement

In Uncertainty Quantification, we are often faced with the **Uncertainty Propagation** problem:

- We have a model $\mathcal{M}$ (analytical, FEA, CFD).
- We have input variables $\mathbf{X}$ described by a joint PDF $f_{\mathbf{X}}(\mathbf{x})$.
- We want to characterize the output $Y = \mathcal{M}(\mathbf{X})$.

While local methods (Taylor series) approximate $\mathcal{M}$, **Monte Carlo** is a global method that samples the input space directly.

Numerical Approach for Solving Integrals

For a given statistic of the output Y written as an expectation:

$$\mathbb{E}[h(Y)] = \int_{\mathcal{D}_{\mathbf{X}}} h(\mathcal{M}(\mathbf{x})) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

Monte Carlo replaces this high-dimensional integral with a discrete sum.
Examples:

- If $h(Y) = Y$, we calculate the **Mean** (μ_Y).
- If $h(Y) = (Y - \mu_Y)^2$, we calculate the **Variance** (σ_Y^2).
- If $h(Y) = \mathbb{I}_{Y \leq 0}$, we calculate the **Probability of Failure** (P_f).

The Monte Carlo Workflow

The implementation of Monte Carlo simulation follows a systematic procedure:

1. **Input Sampling:** Generate a set of N independent and identically distributed (i.i.d.) realizations from the joint input PDF:

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\} \sim f_{\mathbf{X}}(\mathbf{x})$$

2. **Model Propagation:** Evaluate the computational model for each sample point to obtain the corresponding output realizations:

$$y^{(j)} = \mathcal{M}(\mathbf{x}^{(j)}) \quad \text{for } j = 1, \dots, N$$

Note: The model $\mathcal{M}$ is treated as a **non-intrusive "black-box."** This approach requires no internal knowledge of the solver, such as gradients or adjoint formulations.

Statistical Estimation

Once we have the set of outputs $\{y^{(1)}, \ldots, y^{(N)}\}$, we calculate estimators: **Sample Mean:**

$$\hat{\mu}_Y = \frac{1}{N} \sum_{j=1}^{N} y^{(j)}$$

Unbiased Sample Variance:

$$\hat{\sigma}_Y^2 = \frac{1}{N-1} \sum_{j=1}^{N} (y^{(j)} - \hat{\mu}_Y)^2$$

The "hat" notation (ˆ) signifies that these are **estimates** based on a finite sample size N.

Mathematical Foundation

Two theorems from probability theory justify this approach:

1. **Weak Law of Large Numbers (WLLN):** Guarantees that the estimator is **consistent**: $\hat{\mu}_Y \xrightarrow{P} \mathbb{E}[Y]$ as $N \to \infty$.
2. **Central Limit Theorem (CLT):** Provided $\text{Var}(Y) < \infty$, the distribution of the estimation error $(\hat{\mu}_Y - \mu_Y)$ converges to a Normal distribution $\mathcal{N}(0, \sigma_Y^2/N)$.

This allows us to calculate **Confidence Intervals** for our results, quantifying how much we can trust our N-sample estimate.

The Cost of Precision

The precision of our estimate is measured by the **Standard Error**:

$$\epsilon_{MC} = \frac{\sigma_Y}{\sqrt{N}}$$

Crucial Consequences:

- **Convergence Rate:** $\mathcal{O}(1/\sqrt{N})$.
- To reduce the error by a factor of 10, you must increase N by a factor of 100.
- **The Blessing:** This rate does not depend on the number of inputs n. This makes MC the only viable method for very high-dimensional problems.

Inferring the Full Output Distribution

The first two moments (μ, σ) often hide critical information (skewness, bimodal behavior). To "see" the whole distribution, we use the collection of outputs $\{y^{(1)}, \ldots, y^{(N)}\}$ to build an Empirical PDF.

Reconstruction using a Histogram

- Simplest way to visualize the output PDF behavior.
- **Pros:** Easy to compute and intuitive.
- **Cons:** Very sensitive to the choice of bin width and starting positions.

Non-parametric Density Estimation: KDE

Kernel Density Estimation (KDE) is a method to reconstruct the continuous PDF $f_Y(y)$ from a finite sample set $\{y^{(j)}\}_{j=1}^N$ without assuming a specific distribution shape.

The Kernel Estimator:

$$\hat{f}_Y(y) = \frac{1}{Nh} \sum_{j=1}^{N} K\left(\frac{y - y^{(j)}}{h}\right)$$

- **Kernel Function** $K(\cdot)$**:** A symmetric, non-negative function that integrates to one (e.g., Gaussian, Epanechnikov).
- **Bandwidth** $h > 0$**:** A smoothing parameter that controls the width of the kernels.

The Bandwidth Trade-off:

- **Small** h**:** Low bias, but high variance (the PDF is "noisy" and overfits the samples).
- **Large** h**:** Low variance, but high bias (the PDF is "oversmoothed" and masks features).

Summary: When to use Monte Carlo?

- **Use it when:** The model is a black-box, the number of inputs is huge, or the model is highly non-linear/discontinuous.
- **Avoid it when:** The model is very expensive (e.g., 1 run = 24 hours), and you only need a rough estimate of the mean.

Monte Carlo is the "Gold Standard": we use it to verify the accuracy of all other UQ methods.

Example 1: Quadratic Benchmark

A fundamental case to verify the second-order mean shift and Monte Carlo convergence.

Problem Setup:

- Model: $Y = X^2$
- Input: $X \sim \mathcal{N}(\mu = 10, \sigma = 2)$

Theoretical Moments:

- $E[Y] = \mu_X^2 + \sigma_X^2 = 104$
- $\mathrm{Var}(Y) = 4\mu_X^2\sigma_X^2 + 2\sigma_X^4 = 1632$

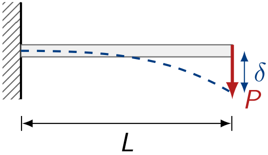
Self-Correction Check:
If your perturbation mean is 100, your Hessian implementation is missing.

Example 2: Engineering Project

A multi-variable model involving parameters with vastly different scales.

Model: $\delta = \frac{PL^3}{3EI}$

Variable	Mean (μ)	Std (σ)
Load (P)	5000 N	500 N
Length (L)	2.0 m	0.05 m
Modulus (E)	210 GPa	10 GPa
Inertia (I)	10^{-5} m^4	5×10^{-7} m^4



Practical Tips for Implementation

To ensure successful uncertainty propagation in your reports:

- **Vectorization:** Write your model function to accept matrices of size $(N \times n)$. This makes Monte Carlo significantly faster in MATLAB.
- **Numerical Stability:** Use relative step sizes for Finite Differences:

$$h_i = \mu_i \times 10^{-4}$$

- **Verification:** Always run Example 1 first. If your code cannot solve X^2, it will not solve a complex engineering model.
- **MC Precision:** Monitor the convergence of your results. If the mean fluctuates significantly, increase your sample size N.