Multifidelity Uncertainty Quantification

Benjamin Peherstorfer
Courant Institute of Mathematical Sciences, New York University

June 2022
Intro: Uncertainties due to data
Intro: Uncertainties due to unknown parameters

[Figures: Petra, Ghattas, Isaac, Martin, Stadler, et al.]
Intro: No hope to exhaustively model physics
Intro: Manufacturing variations

**Intro: Model**

**Model of system of interest**
- Model describes response of system to inputs, parameters, configurations
- Response typically is a quantity of interest
- Evaluating a model means numerically simulating the model
- Many models given in form of partial differential equations

![Diagram](image)

**Mathematical formulation**

\[ f : \mathcal{D} \rightarrow \mathcal{Y} \]

- Input domain \( \mathcal{D} \) and output domain \( \mathcal{Y} \)
- Maps \( z \in \mathcal{D} \) input onto \( y \in \mathcal{Y} \) output (quantity of interest)
Intro: Model - Navier-Stokes equations

\[
\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \Delta u + g
\]

Examples of inputs
- Density \( \rho \)
- Dynamic viscosity \( \mu \)

Examples of outputs (quantities of interest)
- Velocity at monitoring point
- Average pressure

[Figure: MFIX, NETL, DOE]
Intro: Model - Diffusion-convection-reaction flow

\[ \frac{\partial u}{\partial t} = \Delta u - \nu \nabla u + g(u, \mu) \]

**Examples of inputs**
- Activation energy and pre-exponential factor (Arrhenius-type reaction)
- Temperature at boundary
- Ratio of fuel and oxidizer

**Examples of outputs**
- Average temperature in chamber
Intro: Uncertain inputs

Inputs are uncertain

- Measurement errors in boundary conditions
- Manufacturing variations
- Model parameters determined by engineering judgment
- ...
Intro: Uncertain inputs

Inputs are uncertain

• Measurement errors in boundary conditions
• Manufacturing variations
• Model parameters determined by engineering judgment
• ...

Mathematically formulate uncertain inputs as random variables

\[ Z : \Omega \rightarrow \mathcal{D} \]
Intro: Uncertain inputs

Inputs are uncertain

- Measurement errors in boundary conditions
- Manufacturing variations
- Model parameters determined by engineering judgment
- ...

Mathematically formulate uncertain inputs as random variables

\[ Z : \Omega \rightarrow \mathcal{D} \]

Quantify effect of uncertainties in inputs on model outputs
Intro: General sampling-based approach to UQ

- Take many realizations of input random variable $Z$
  \[ z_1, \ldots, z_n \in \mathcal{D} \]
- Evaluate model $f$ at all $z_1, \ldots, z_n$ realizations
  \[ y_1 = f(z_1), \ldots, y_n = f(z_n) \]
- Estimate statistics (mean, std. deviation, etc) from outputs $y_1, \ldots, y_n$
Intro: General sampling-based approach to UQ

- Take many realizations of input random variable $Z$
  \[ z_1, \ldots, z_n \in \mathcal{D} \]
- Evaluate model $f$ at all $z_1, \ldots, z_n$ realizations
  \[ y_1 = f(z_1), \ldots, y_n = f(z_n) \]
- Estimate statistics (mean, std. deviation, etc) from outputs $y_1, \ldots, y_n$
Intro: General sampling-based approach to UQ

• Take many realizations of input random variable $Z$

$$z_1, \ldots, z_n \in D$$

• Evaluate model $f$ at all $z_1, \ldots, z_n$ realizations

$$y_1 = f(z_1), \ldots, y_n = f(z_n)$$

• Estimate statistics (mean, std. deviation, etc) from outputs $y_1, \ldots, y_n$
Monte Carlo

- Models treated as black box
- Dimension independent
- Easily parallelizable
Monte Carlo

- Models treated as black box
Monte Carlo

- Models treated as black box
- Dimension independent
Monte Carlo

- Models treated as black box
- Dimension independent
- Easily parallelizable
Intro: Challenges of sampling-based UQ

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension
Intro: Challenges of sampling-based UQ

Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive \( \Rightarrow \) multifidelity
- Uncertain parameters are of high dimension
Intro: Challenges of sampling-based UQ

- Many iterations of a computational model $f : D \rightarrow Y$

Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension
Intro: Challenges of sampling-based UQ

Challenges

- Formulation and modeling of uncertainties
- Models based on PDEs: nonlinear, multi-scale, multi-physics
- Single model solve expensive; repeated solves prohibitive $\Rightarrow$ multifidelity
- Uncertain parameters are of high dimension
Intro: Opportunity of low-fidelity models

Given is typically a high-fidelity model

- Large-scale numerical simulation
- Achieves required accuracy
- Computationally expensive

Additionally, often have available or can train low-fidelity models

- Approximate same quantity of interest as high-fidelity model
- Often orders of magnitudes cheaper than high-fidelity model
- Less accurate and typically no accuracy guarantees
Intro: Three types of low-fidelity models

simplified models
- Simplifying physics
- Coarser discretizations
- Linearized models
- Early stopping of iterative solvers

data-fit models
- Fitting model to data of input-output map given by high-fidelity model
- Response surfaces
- Gaussian processes
- Neural networks

reduced models
- Extract important dynamics of full states from data
- Approximate high-dimensional states in subspaces
- Restrict solving governing equations to subspaces
Intro: Low-fidelity models

Replace high- with low-fidelity model
- Costs of outer loop application reduced
- Often orders of magnitude speedups

Low-fidelity model introduces error
- Control with error bounds/estimators*
- Rebuild if accuracy too low
- No guarantees without bounds/estimators

Issues
- Propagation of output error on estimate
- Applications without error control
- Costs of rebuilding a low-fidelity model
Multifidelity: Combine multiple models

Combine high-fidelity and low-fidelity models
- Leverage low-fidelity models for speedup
- Recourse to high-fidelity for accuracy

Multifidelity speeds up computations
- Balance #solves among models
- Adapt, fuse, filter with low-fidelity models

Multifidelity guarantees high-fidelity accuracy
- Occasional recourse to high-fidelity model
- High-fidelity model is kept in the loop
- Independent of error control of low fidelity

Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization*

Benjamin Peherstorfer†
Karen Willcox‡
Max Gunzburger§

Abstract. In many situations across computational science and engineering, multiple computational models are available that describe a system of interest. These different models have varying evaluation costs and varying fidelities. Typically, a computationally expensive high-fidelity model describes the system with the accuracy required by the current application at hand, while lower-fidelity models are less accurate but computationally cheaper than the high-fidelity model. Outer-loop applications, such as optimization, inference, and uncertainty quantification, require multiple model evaluations at many different inputs, which often leads to computational demands that exceed available resources if only the high-fidelity model is used. This work surveys multifidelity methods that accelerate the solution of outer-loop applications by combining high-fidelity and low-fidelity model evaluations, where the low-fidelity evaluations arise from an explicit low-fidelity model (e.g., a simplified physics approximation, a reduced model, a data-fit surrogate) that approximates the same output quantity as the high-fidelity model. The overall premise of these multifidelity methods is that low-fidelity models are leveraged for speedup while the high-fidelity model is kept in the loop to establish accuracy and/or convergence guarantees. We categorize multifidelity methods according to three classes of strategies: adaptation, fusion, and filtering. The paper reviews multifidelity methods in the outer-loop contexts of uncertainty propagation, inference, and optimization.

Key words. multifidelity, surrogate models, model reduction, multifidelity uncertainty quantification, multifidelity uncertainty propagation, multifidelity statistical inference, multifidelity optimization

AMS subject classifications. 65-02, 62-02, 49-02

DOI. 10.1137/16M1082469
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

\[ f : \mathcal{D} \rightarrow \mathcal{Y} \]

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
MFMC: Monte Carlo estimation

High-fidelity ("truth") model, costs $w_1 > 0$

$$f^{(1)} : D \rightarrow Y$$

Random variable $Z$, estimate

$$s = \mathbb{E}[f^{(1)}(Z)]$$

Monte Carlo estimate of $s$ with real. $z_1, \ldots, z_n$

$$\bar{y}^{(1)}_n = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i)$$

Computational costs
- Many evaluations of high-fidelity model
- Typically $10^3 - 10^6$ evaluations
- Intractable if $f^{(1)}$ expensive
MFMC: Control variates

Given is a random variable $A$ and we want to estimate its mean

$$s_A = \mathbb{E}[A]$$
MFMC: Control variates

Given is a random variable $A$ and we want to estimate its mean

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$
MFMC: Control variates

Given is a random variable $A$ and we want to estimate its mean

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$

Regular Monte Carlo estimator of $s_A$

$$\bar{a}_n = \frac{1}{n} \sum_{i=1}^{n} a_i$$
MFMC: Control variates

Given is a random variable $A$ and we want to estimate its mean

$$s_A = \mathbb{E}[A]$$

Independent and identically distributed (i.i.d.) samples

$$a_1, \ldots, a_n$$

Regular Monte Carlo estimator of $s_A$

$$\bar{a}_n = \frac{1}{n} \sum_{i=1}^{n} a_i$$

Unbiased estimator $\mathbb{E}[\bar{a}_n] = s_A$ with mean-squared error (MSE)

$$\text{Var}[\bar{a}_n] = \frac{1}{n^2} \text{Var} \left[ \sum_{i=1}^{n} a_i \right] = \frac{\text{Var}[A]}{n}$$
MFMC: Control variates (cont’d)

Additional random variable $B$ with known mean $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$
MFMC: Control variates (cont’d)

Additional random variable $B$ with *known* mean $s_B = \mathbb{E}[B]$ and samples $b_1, \ldots, b_n$

Regular Monte Carlo estimator of $s_B$

$$b_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$
MFMC: Control variates (cont’d)

Additional random variable $B$ with known mean $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

Regular Monte Carlo estimator of $s_B$

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$

Control variate estimator of $s_A$ that uses samples from $A$ and $B$

$$\hat{s}_A = \bar{a}_n + (s_B - \bar{b}_n)$$

[\text{Nelson, 87}]
MFMC: Control variates (cont’d)

Additional random variable $B$ with known mean $s_B = \mathbb{E}[B]$ and samples

$$b_1, \ldots, b_n$$

Regular Monte Carlo estimator of $s_B$

$$\bar{b}_n = \frac{1}{n} \sum_{i=1}^{n} b_i$$

Control variate estimator of $s_A$ that uses samples from $A$ and $B$

$$\hat{s}_A = \bar{a}_n + (s_B - \bar{b}_n)$$

Introduce coefficient $\alpha \in \mathbb{R}$ to balance $A$ and $B$

$$\hat{s}_A = \bar{a}_n + \alpha (s_B - \bar{b}_n)$$

Combines $n$ samples of $A$ and $n$ samples of $B$

[Nelson, 87]
MFMC: Control variates (cont’d)

Control variate estimator

\[ \hat{s}_A = \bar{a}_n + \alpha (s_B - \bar{b}_n) \]
MFMC: Control variates (cont’d)

Control variate estimator

\[ \hat{s}_A = \bar{a}_n + \alpha (s_B - \bar{b}_n) \]

Unbiased estimator of \( s_A \) because

\[
\mathbb{E}[\hat{s}_A] = \mathbb{E}[\bar{a}_n] + \alpha \mathbb{E}[s_B - \bar{b}_n] = s_A = s_A + 0 = s_A
\]
MFMC: Control variates (cont’d)

Control variate estimator

\[ \hat{s}_A = \bar{a}_n + \alpha (s_B - \bar{b}_n) \]

Unbiased estimator of \( s_A \) because

\[ \mathbb{E}[\hat{s}_A] = \mathbb{E}[\bar{a}_n] + \alpha \mathbb{E}[s_B - \bar{b}_n] = s_A \]

Variance of control variate estimator for optimal* \( \alpha \in \mathbb{R} \)

\[ \operatorname{Var}[\hat{s}_A] = (1 - \rho^2) \frac{\operatorname{Var}[A]}{n} = (1 - \rho^2) \operatorname{Var}[\bar{a}_n] \]

- Correlation coefficient \(-1 \leq \rho \leq 1\) of \( A \) and \( B \)
- If \( \rho = 0 \), same variance as regular Monte Carlo
- If \( |\rho| > 0 \), lower variance
- The higher correlated, the lower variance of \( \hat{s}_A \)
MFMC: Multifidelity Monte Carlo Estimation

Estimate expected value

\[ s = \mathbb{E}[f^{(1)}(Z)] \]

Low-fidelity models

\[ f^{(2)}, f^{(3)}, \ldots, f^{(k)} : \mathcal{D} \rightarrow \mathcal{Y} \]

Correlation coefficients

\[ \rho_2 = \text{Corr}[f^{(1)}, f^{(2)}], \rho_3 = \text{Corr}[f^{(1)}, f^{(3)}], \ldots, \rho_k = \text{Corr}[f^{(1)}, f^{(k)}] \]

Costs

\[ w_1, w_2, \ldots, w_k > 0 \]

No need to know expected values of low-fidelity models!
MFMC: Multifidelity Monte Carlo

Reminder: Monte Carlo estimator

\[
\bar{y}_n^{(1)} = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i)
\]

Multifidelity Monte Carlo (MFMC) estimator

\[
\hat{s} = \bar{y}_{m_1}^{(1)} + \sum_{i=2}^{k} \alpha_i \left( \bar{y}_{m_i}^{(i)} - \bar{y}_{m_{i-1}}^{(i)} \right)
\]

- Monte Carlo estimator

\[
\bar{y}_{m_i}^{(i)} = \frac{1}{m_i} \sum_{i=1}^{m_i} f^{(i)}(z_i)
\]

- Number of model evaluations \( m = [m_1, \ldots, m_k]^T \)
- Control variate coefficients \( \alpha = [\alpha_2, \ldots, \alpha_k]^T \)
- Optimal selection of \( m \) and \( \alpha \rightarrow \) our code
MFMC: Multifidelity Monte Carlo

Reminder: Monte Carlo estimator

\[ \bar{y}^{(1)}_n = \frac{1}{n} \sum_{i=1}^{n} f^{(1)}(z_i) \]

Multifidelity Monte Carlo (MFMC) estimator

\[ \hat{s} = \underbrace{\bar{y}^{(1)}_{m_1}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \left( \underbrace{\bar{y}^{(i)}_{m_i} - \bar{y}^{(i)}_{m_i-1}}_{\text{from low-fid. models}} \right) \]

- Monte Carlo estimator

\[ \bar{y}^{(i)}_{m_i} = \frac{1}{m_i} \sum_{i=1}^{m_i} f^{(i)}(z_i) \]

- Number of model evaluations \( m = [m_1, \ldots, m_k]^T \)
- Control variate coefficients \( \alpha = [\alpha_2, \ldots, \alpha_k]^T \)
- Optimal selection of \( m \) and \( \alpha \rightarrow \text{our code} \)
MFMC: Recipe 1

Download

https://github.com/pehersto/mfmc

Given

- Models $f^{(1)}, f^{(2)}, \ldots, f^{(k)}$
- Computational budget $b$

Pilot run

- Draw $m_0$ ($\approx 50$) realizations of $Z$
- Evaluate each model $f^{(1)}, f^{(2)}, \ldots, f^{(k)}$ at the $m_0$ realizations

$$Y = \begin{bmatrix}
  f^{(1)}(z_1) & f^{(2)}(z_1) & \ldots & f^{(k)}(z_1) \\
  \vdots & \vdots & & \vdots \\
  f^{(1)}(z_{m_0}) & f^{(2)}(z_{m_0}) & \ldots & f^{(k)}(z_{m_0})
\end{bmatrix}$$

- Estimate computational costs of model evaluations $w = [w_1, \ldots, w_k]^T$
Determine number of model evaluations

\[ [m, a] = \text{optimMlevelCorr}(Y, w, b) \]

- Number of model evaluations \( m = [m_1, m_2, \ldots, m_k]^T \)
- Coefficients \( a = [\alpha_2, \alpha_3, \ldots, \alpha_k]^T \)

Draw realizations

\( z_1, z_2, \ldots, z_{m_k} \)

Evaluate models

\[ f(i)(z_1), \ldots, f(i)(z_{m_i}), \quad i = 1, \ldots, k \]

Estimate

\[ \hat{s} = \begin{cases} \bar{y}_{m_1}^{(1)} & \text{from HFM} \\ \sum_{i=2}^{k} \alpha_i \left( \bar{y}_{m_i}^{(i)} - \bar{y}_{m_{i-1}}^{(i)} \right) & \text{from low.-fid. models} \end{cases} \]
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM, LFM1, LFM2, LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget
```
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM,LFM1,LFM2,LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i=1:length(modelList)
    Y(:, i) = modelList{i}(mu);
end
```

% evaluate low-fidelity models
for i = 2:length(modelList)
    y = modelList{i}(z(1:m(i), :));
    sHat = sHat + alpha(i)*(mean(y)-mean(y(1:m(i-1))));
end
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM, LFM1, LFM2, LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i = 1:length(modelList)
    Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optimMlevelCorr(Y, w, budget); % MFMC
```
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM, LFM1, LFM2, LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i = 1:length(modelList)
    Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optimMlevelCorr(Y, w, budget); % MFMC

z = drawSamples(m(end)); % draw realizations
```
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM, LFM1, LFM2, LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i = 1:length(modelList)
    Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optimMlevelCorr(Y, w, budget); % MFMC
z = drawSamples(m(end)); % draw realizations

y = modelList{1}(z(1:m(1), :)); % evaluate HFM
sHat = alpha(1)*mean(y);
```
MFMC: Matlab code for Recipe 1

```matlab
modelList = {HFM, LFM1, LFM2, LFM3}; % models
w = [100, 50, 20, 10]'; % costs
budget = 1000*w(1); % total budget

mu = drawSamples(50); % pilot samples
for i = 1:length(modelList)
    Y(:, i) = modelList{i}(mu);
end

[m, alpha] = optiMlevelCorr(Y, w, budget); % MFMC

z = drawSamples(m(end)); % draw realizations

y = modelList{1}(z(1:m(1), :)); % evaluate HFM
sHat = alpha(1)*mean(y);

% evaluate low-fidelity models
for i = 2:length(modelList)
    y = modelList{i}(z(1:m(i), :));
    sHat = sHat + alpha(i)*(mean(y) - mean(y(1:m(i-1))));
end
```
MFMC: Recipe 2 (MFMC as post-processing)

Given
- Model evaluations
  \[ f^{(i)}(z_1), \ldots, f^{(i)}(z_{m_i}), \quad i = 1, \ldots, k \]
- Model evaluation costs \( w_1, \ldots, w_k \)

Pilot samples
- Use the first \( m_0 \ll m_1 \) samples to form
  \[
  Y = \begin{bmatrix}
  f^{(1)}(z_1) & f^{(2)}(z_1) & \ldots & f^{(k)}(z_1) \\
  \vdots & \vdots & & \vdots \\
  f^{(1)}(z_{m_0}) & f^{(2)}(z_{m_0}) & \ldots & f^{(k)}(z_{m_0})
  \end{bmatrix}
  \]
- Derive coefficients
  \[
  [ \sim, a ] = \text{optiMlevelCorr}( Y, w, b )
  \]

Estimate
\[
  s = \underbrace{\tilde{y}^{(1)}_{m_1}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \left( \underbrace{\tilde{y}^{(i)}_{m_i}}_{\text{from low.-fid. models}} - \underbrace{\tilde{y}^{(i)}_{m_{i-1}}}_{\text{from low.-fid. models}} \right)
\]
MFMC: Recipe 2 (MFMC as post-processing)

Given

- **Model evaluations**

\[ f^{(i)}(z_1), \ldots, f^{(i)}(z_{m_i}), \quad i = 1, \ldots, k \]

- **Model evaluation costs** \( w_1, \ldots, w_k \)

**Pilot samples**

- **Use the first** \( m_0 \ll m_1 \) **samples to form**

\[
Y = \begin{bmatrix}
  f^{(1)}(z_1) & f^{(2)}(z_1) & \ldots & f^{(k)}(z_1) \\
  \vdots & \vdots & & \vdots \\
  f^{(1)}(z_{m_0}) & f^{(2)}(z_{m_0}) & \ldots & f^{(k)}(z_{m_0})
\end{bmatrix}
\]

- **Derive coefficients**

\[
[ \sim, a ] = \text{optiMlevelCorr}( Y, w, b )
\]

**Estimate**

\[
s = \underbrace{\bar{y}_{m_1}^{(1)}}_{\text{from HFM}} + \sum_{i=2}^{k} \alpha_i \left( \bar{y}_{m_i}^{(i)} - \bar{y}_{m_{i-1}}^{(i)} \right)_{\text{from low.-fid. models}}
\]
MFMC: AeroStruct: Problem setup

Coupled aero-structural wing analysis
- Uncertain are angle of attack, air density, Mach number
- Estimate expected fuel burn

High-fidelity model $f^{(1)}$
- OpenAeroStruct code
- Vortex-lattice method
- 6 DoF 3-dim spatial beam model
- Used with default configuration

Low-fidelity models
- Spline interpolants on equidistant grid
- Low-fidelity model $f^{(2)}$ from 343 points
- Low-fidelity model $f^{(3)}$ from 125 points


https://github.com/johnjasa/OpenAeroStruct/
MFMC: AeroStruct: Distribution of work

Model properties

<table>
<thead>
<tr>
<th>model</th>
<th>evaluation costs [s]</th>
<th>offline costs [s]</th>
<th>correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-fid. $f^{(1)}$</td>
<td>$1.61 \times 10^{-1}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>low-fid. $f^{(2)}$</td>
<td>$1.23 \times 10^{-7}$</td>
<td>55.382</td>
<td>$9.9552 \times 10^{-1}$</td>
</tr>
<tr>
<td>low-fid. $f^{(3)}$</td>
<td>$1.21 \times 10^{-7}$</td>
<td>20.183</td>
<td>$9.9192 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

Number of model evaluations

<table>
<thead>
<tr>
<th>online costs [s]</th>
<th>Monte Carlo</th>
<th>MFMC with $f^{(1)}, f^{(2)}$</th>
<th>MFMC with $f^{(1)}, f^{(3)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#evals $f^{(1)}$</td>
<td>#evals $f^{(1)}$</td>
<td>#evals $f^{(2)}$</td>
</tr>
<tr>
<td>$7.99 \times 10^6$</td>
<td>50</td>
<td>$4.90 \times 10^1$</td>
<td>$4.48 \times 10^5$</td>
</tr>
<tr>
<td>1.61 \times 10^1</td>
<td>100</td>
<td>$9.90 \times 10^1$</td>
<td>$8.95 \times 10^5$</td>
</tr>
<tr>
<td>8.07 \times 10^1</td>
<td>500</td>
<td>$4.96 \times 10^2$</td>
<td>$4.48 \times 10^6$</td>
</tr>
<tr>
<td>1.61 \times 10^2</td>
<td>1000</td>
<td>$9.93 \times 10^2$</td>
<td>$8.95 \times 10^6$</td>
</tr>
<tr>
<td>8.07 \times 10^2</td>
<td>5000</td>
<td>$4.97 \times 10^3$</td>
<td>$4.48 \times 10^7$</td>
</tr>
</tbody>
</table>

MFMC trades high-fidelity evaluations for low-fidelity evaluations

- The high-fidelity model evaluations guarantee unbiased
- The low-fidelity model evaluations help to reduce the variance
- The balance is optimal with respect to the mean-squared error
MFMC: AeroStruct: Distribution of work

Model properties

<table>
<thead>
<tr>
<th>model</th>
<th>evaluation costs [s]</th>
<th>offline costs [s]</th>
<th>correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-fid. ( f^{(1)} )</td>
<td>( 1.61 \times 10^{-1} )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>low-fid. ( f^{(2)} )</td>
<td>( 1.23 \times 10^{-7} )</td>
<td>55.382</td>
<td>( 9.9552 \times 10^{-1} )</td>
</tr>
<tr>
<td>low-fid. ( f^{(3)} )</td>
<td>( 1.21 \times 10^{-7} )</td>
<td>20.183</td>
<td>( 9.9192 \times 10^{-1} )</td>
</tr>
</tbody>
</table>

Number of model evaluations

<table>
<thead>
<tr>
<th>online costs [s]</th>
<th>Monte Carlo ( #\text{evals} f^{(1)} )</th>
<th>MFMC with ( f^{(1)}, f^{(2)} ) ( #\text{evals} f^{(1)} ) ( #\text{evals} f^{(2)} )</th>
<th>MFMC with ( f^{(1)}, f^{(3)} ) ( #\text{evals} f^{(1)} ) ( #\text{evals} f^{(3)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 7.99 \times 10^0 )</td>
<td>50</td>
<td>( 4.90 \times 10^1 ) ( 4.48 \times 10^5 )</td>
<td>( 4.90 \times 10^1 ) ( 5.97 \times 10^5 )</td>
</tr>
<tr>
<td>( 1.61 \times 10^1 )</td>
<td>100</td>
<td>( 9.90 \times 10^1 ) ( 8.95 \times 10^5 )</td>
<td>( 9.90 \times 10^1 ) ( 1.19 \times 10^6 )</td>
</tr>
<tr>
<td>( 8.07 \times 10^1 )</td>
<td>500</td>
<td>( 4.96 \times 10^2 ) ( 4.48 \times 10^6 )</td>
<td>( 4.95 \times 10^2 ) ( 5.97 \times 10^6 )</td>
</tr>
<tr>
<td>( 1.61 \times 10^2 )</td>
<td>1000</td>
<td>( 9.93 \times 10^2 ) ( 8.95 \times 10^6 )</td>
<td>( 9.90 \times 10^2 ) ( 1.19 \times 10^7 )</td>
</tr>
<tr>
<td>( 8.07 \times 10^2 )</td>
<td>5000</td>
<td>( 4.97 \times 10^3 ) ( 4.48 \times 10^7 )</td>
<td>( 4.95 \times 10^3 ) ( 5.97 \times 10^7 )</td>
</tr>
</tbody>
</table>

MFMC trades high-fidelity evaluations for low-fidelity evaluations

- The high-fidelity model evaluations guarantee unbiased
- The low-fidelity model evaluations help to reduce the variance
- The balance is optimal with respect to the mean-squared error
Low-fidelity model alone leads to biased estimators

MFMC achieves speedup of about two order of magnitude
Constructing low-fidelity models incurs offline costs

In this example, offline costs low compared to savings
• Model $f^{(2)}$ and $f^{(3)}$ are similar with respect to costs/correlations
• Adding model $f^{(2)}$ has little effect
MFMC: Plate

Locally damaged plate in bending
- Inputs: nominal thickness, load, damage
- Output: maximum deflection of plate
- Only distribution of inputs known
- Estimate expected deflection

Six models
- High-fidelity model: FEM, 300 DoFs
- Reduced model: POD, 10 DoFs
- Reduced model: POD, 5 DoFs
- Reduced model: POD, 2 DoFs
- Data-fit model: linear interp., 256 pts
- Support vector machine: 256 pts

Var, corr, and costs est. from 100 samples
Largest improvement from “single → two” and “two → three”
Adding yet another reduced/SVM model reduces variance only slightly
MFMC: Plate: #evals of models

- MFMC distributes #evals among models depending on corr/costs
- Number of evaluation changes exponentially between models
- Highest #evals in data-fit models (cost ratio $w_1/w_6 \approx 10^6$)
Multifidelity prediction in wildfire spread simulation: Modeling, uncertainty quantification and sensitivity analysis

Mario Miguel Valero*,†, Lluís Jofre‡, Ricardo Torres§

* Department of Meteorology and Climate Science, San José State University, San José, CA, 95192, USA
† Center for Turbulence Research, Stanford University, Stanford, CA, 94305, USA
‡ Department of Fluid Mechanics, Technical University of Catalonia – BarcelonaTech, Barcelona, 08028, Spain
A multifidelity method for a nonlocal diffusion model

Parisa Khodabakhshi a,*, Karen E. Willcox a, Max Gunzburger a,b

a Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, TX 78712, USA
b Department of Scientific Computing, Florida State University, Tallahassee, FL 32306, USA

* Corresponding author.
Accelerating the estimation of collisionless energetic particle confinement statistics in stellarators using multifidelity Monte Carlo

Frederick Law*, Antoine Cerfon® and Benjamin Peherstorfer

Courant Institute of Mathematics, New York University, United States of America

E-mail: law@cims.nyu.edu

Received 18 August 2021, revised 16 December 2021
Accepted for publication 31 December 2021
Published 4 May 2022

Abstract
Control Variate Multifidelity Estimators for the Variance and Sensitivity Analysis of Mesostructure–Structure Systems

Variance and sensitivity analysis are challenging tasks when the evaluation of system performances incurs a high-computational cost. To resolve this issue, this paper investigates several multifidelity statistical estimators for the responses of complex systems, especially the mesostructure–structure system manufactured by additive manufacturing. First, this paper reviews an established control variate multifidelity estimator, which leverages the output of an inexpensive, low-fidelity model and the correlation between the high-fidelity model and the low-fidelity model to predict the statistics of the system responses. Second, we investigate several variants of the original estimator and propose a new formulation of the control variate estimator. All these estimators and the associated sensitivity analysis approaches are compared on two engineering examples of mesostructure–structure system analysis. A multifidelity metamodel-based sensitivity analysis approach is also included in the comparative study. The proposed estimator demonstrates its strength in predicting variance when only a limited number of expensive high-fidelity evaluations are available.
Control Variate Multifidelity

Applications of Multifidelity Reduced Order Modeling to Single and Multiphysics Problems

Pengchao Song, X.Q. Wang and Marc P. Minolet

AIAA 2020-2131
Session: Special Session: Managing Multiple Information Sources of Multi-Physics Systems

Published Online: 5 Jan 2020 • https://doi.org/10.2514/6.2020-2131

Abstract:
The focus of the present investigation is on assessing the applicability and performance of the recently introduced Multifidelity Monte Carlo (MFMC) for the computationally efficient prediction of the statistics of the random response of uncertain structures especially those undergoing large deformations and modeled within nonlinear reduced order models. Three such nonlinear applications are considered the first of which is a purely structural problem, a panel subjected to a large loads inducing nonlinear geometric effects. Reduced order models with different fidelities are then generated by reducing the size of the basis from a given set of basis functions.
MFMC: Multi-fidelity Monte Carlo in the wild

Control Variate Multifidelity

Applications of Multifidelity Reduced Order Modeling to Single and Multiphysics Problems

Data-driven low-fidelity models for multi-fidelity Monte Carlo sampling in plasma micro-turbulence analysis

Julia Konrad\textsuperscript{a,1}, Ionuț-Gabriel Farcaș\textsuperscript{b,*,1}, Benjamin Peherstorfer\textsuperscript{c}, Alessandro Di Siena\textsuperscript{b}, Frank Jenko\textsuperscript{a,b,d}, Tobias Neckel\textsuperscript{a}, Hans-Joachim Bungartz\textsuperscript{a}

\textsuperscript{a} Department of Informatics, Technical University of Munich, Germany
\textsuperscript{b} Den Institute for Computational Engineering and Sciences, The University of Texas at Austin, United States of America
\textsuperscript{c} Courant Institute of Mathematical Sciences, New York University, United States of America
MFMC: Multi-fidelity Monte Carlo in the wild

Control Variate Multifidelity
Applications of Multifidelity Reduced Order Modeling to Single and Multiphysics Problems

Water Resources Research
Efficient Monte Carlo With Graph-Based Subsurface Flow and Transport Models

Abstract
Simulating flow and transport in fractured porous media frequently involves solving numerical discretizations of partial differential equations with a large number of degrees of freedom using discrete fracture network (DFN) models. Uncertainty in the properties of the fracture network that controls flow and transport requires a large number of DFN simulations to statistically describe quantities of interest. However, the computational cost of solving more than a few realizations of a large DFN can be intractable. As a means of circumventing this problem, we utilize both a high-fidelity DFN model and a graph-based model of flow and transport in combination with a multifidelity Monte Carlo (MC) method to reduce the number of high-fidelity simulations that are needed to obtain an accurate estimate of the quantity of interest. We demonstrate the approach by estimating quantiles of the breakthrough time for a conservative tracer in an aquifer.
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

\[
\text{input } z \xrightarrow{\text{computational model }} y \xrightarrow{\text{output } y} E
\]

2. Multifidelity sensitivity analysis

\[
\text{input } z \xrightarrow{\text{computational model }} y \xrightarrow{\text{output } y} \text{ Pie Chart}
\]

3. Multifidelity failure probability estimation

\[
\text{input } z \xrightarrow{\text{computational model }} y \xrightarrow{\text{output } y} \text{ Arrow}
\]

4. Other multifidelity uncertainty quantification tasks
Uncertainty quantification tasks

1. Multifidelity uncertainty uncertainty propagation

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
MFGSA: Sensitivity analysis

Sensitivity analysis

- Determine which inputs influence output most
- Can sample $Y$ as a black box for inputs $Z$ and need to determine what components of $Z = [Z_1, \ldots, Z_d]^T$ influence $Y$ most

$Y$ is sensitive to $Z$

$Y$ is not sensitive to $Z$
MFGSA: Sensitivity analysis in engineering

Risk communication for decision-making
- Determine if one can rely on model output or if “noise”
- Communicate to upstream decision-making which inputs are critical

Feedback to improve model
- Determine which inputs need to be sampled carefully
- Prioritize effort on reducing uncertainty
- Modify model with respect to sensitive inputs

Model reduction and dimensionality reduction
- Focus on important inputs and ignore ineffective inputs
- Derive surrogate models that depend on important inputs only
MFGSA: Variance-based global sensitivity analysis

- Input \( Z = [Z_1, \ldots, Z_d]^T \in \mathcal{D} \) is a random vector

- Output of model \( Y = f^{(1)}(Z_1, \ldots, Z_d) \) is sensitive to inputs

- Measure sensitivity with variance

- Main effect sensitivity

\[
S_i = \frac{\text{Var}[\mathbb{E}[Y|Z_i]]}{\text{Var}[Y]}
\]

- Main sensitivity indices are normalized

\[
\sum_{i=1}^{d} S_i = 1, \quad S_i \in [0, 1]
\]
MFGSA: Multifidelity estimation

Estimation of sensitivity indices

- Estimate variance instead of expected value
  \[ S_i = \frac{\text{Var}[\mathbb{E}[Y|Z_i]]}{\text{Var}[Y]} \]

- Requires estimating variance for all \( d \) inputs \( Z = [Z_1, \ldots, Z_d] \)

Multifidelity estimation

- Given are low-fidelity models \( f^{(2)}, \ldots, f^{(k)} \)
- Similarly to MFMC, exploit correlations
  \[ \rho_2 = \text{Corr}[f^{(1)}, f^{(2)}], \rho_3 = \text{Corr}[f^{(1)}, f^{(3)}], \ldots, \rho_k = \text{Corr}[f^{(1)}, f^{(k)}] \]

- Estimator has similar structure as estimator for expected values
MFGSA: Premixed flame

Inputs to model are

- Parameters of Arrhenius reaction
- Temperatures at boundary
- Ratio of fuel and oxidizer
- Activation Energy

Output is maximum temperature in chamber

Models

- Model based on finite differences serves as high-fidelity model
- Model with lower fidelity derived with proper orthogonal decomposition

Code available on GitHub

https://github.com/elizqian/mfgsa
MFGSA: Premixed flame: Results

- Monte Carlo too inaccurate for ranking inputs
- Multi-fidelity Monte Carlo allows ranking of inputs
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

\[ f : \mathcal{D} \rightarrow \mathcal{Y} \]

2. Multifidelity sensitivity analysis

\[ f : \mathcal{D} \rightarrow \mathcal{Y} \]

3. Multifidelity failure probability estimation

\[ f : \mathcal{D} \rightarrow \mathcal{Y} \]

4. Other multifidelity uncertainty quantification tasks
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
MFIS: Failure probabilities

System described by high-fidelity model $f^{(1)} : \mathcal{D} \rightarrow \mathcal{Y}$

- Input $z \in \mathcal{Z}$
- Output $y \in \mathcal{Y}$
- Costs of one high-fidelity model evaluation $w_1 > 0$

Define indicator function

$$l^{(1)}(z) = \begin{cases} 1, & f^{(1)}(z) < 0 \\ 0, & \text{else.} \end{cases}$$

Indicator function $l^{(1)}(z) = 1$ signals failure for input $z$

Given random variable $Z$, estimate failure probability

$$P_f = \mathbb{E}_p[l^{(1)}(Z)]$$
MFIS: Rare event simulation

- Monte Carlo estimator of $P_f$ using $m \in \mathbb{N}$ realizations

$$P_{f}^{MC} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)} (z_i)$$

- If $P_f$ small, then only few realizations with $f^{(1)}(z) < 0$

- Require (very) large $m$ to obtain Monte Carlo estimator with acceptable accuracy $\rightarrow$ expensive
MFIS: Rare event simulation is challenging

Costs of rare event simulation grow inverse proportional to $P_f$

- Monte Carlo estimation of $P_f$ with $m$ realizations

$$P_{f}^{MC} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)}(z_i)$$

- Relative mean-squared error (MSE) of $P_{f}^{MC}$

$$e(P_{f}^{MC}) = \mathbb{E}_p \left[ \left( \frac{P_{f}^{MC} - P_f}{P_f} \right)^2 \right] = \frac{\text{Var}_p [I^{(1)}(Z)]}{P_f^2 m} = \frac{1 - P_f}{P_f m}$$

- For constant $m$, the rel. MSE increases inverse proportional to $P_f$
- A small failure probability $P_f$ needs to be compensated with a large number of samples $m$
- Example: For $P_f = 10^{-5}$ need $m \approx 10^7$ to achieve $e(P_{f}^{MC}) \leq 10^{-2}$

Challenge

costs per sample + number of samples
MFIS: Rare events in aerospace engineering

Rare event simulation
- Failure probability estimation
- Reliability engineering

Risk assessment
- Communicate to upstream decision-making
- Mitigate catastrophic events

Risk-averse optimization
- Deliver baseline performance outside nominal operating conditions
- Take into account dynamics at limit states
MFIS: Importance sampling

- Importance sampling (IS) creates biasing density \( q \) to put more weight on failure events.

- Let \( \hat{Z} \) be the corresponding random variable.

- Introduce the weight function
  \[
  r(z') = \frac{p(\hat{Z})}{q(\hat{Z})}
  \]

- Reformulate failure probability
  \[
  P_f = E_p[I^{(1)}(Z)] = E_q[I^{(1)}(\hat{Z})r(\hat{Z})]
  \]
MFIS: Multifidelity importance sampling

step 1
construction of biasing distribution

step 2
estimation of failure probability
MFIS: Multifidelity importance sampling

step 1
construction of biasing distribution

low-fidelity model

low-fidelity, cheap

biased

step 2
estimation of failure probability

low-fidelity model
MFIS: Multifidelity importance sampling

step 1
construction of biasing distribution

low-fidelity model

low-fidelity, cheap

biased

step 2
estimation of failure probability

low-fidelity model

high-fidelity, expensive

unbiased

high-fidelity model

computational costs
MFIS: Multifidelity importance sampling

Step 1: Construction of biasing distribution
- Low-fidelity model
  - Low-fidelity, cheap
  - Biased

Step 2: Estimation of failure probability
- Low-fidelity model
  - Multifidelity unbiased

- High-fidelity model
  - High-fidelity, expensive
  - Unbiased
MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $z_1, \ldots, z_n$ of $Z$
- Fit mixture model $q$ (biasing) to realizations \( \rightarrow \) scikit-learn, Matlab

\[ \{z_i \mid l^{(2)}(z_i) = 1, i = 1, \ldots, n\} \]

- Derive random variable $\hat{Z}$ with density $q$

Step 2: Estimate $P_f$ with high-fidelity model $f^{(1)}$

\[ P_{f}^{\text{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} l^{(1)}(\hat{z}_i) \frac{p(\hat{z}_i)}{q(\hat{z}_i)} \]

Multifidelity estimator $P_{f}^{\text{MFIS}}$ is unbiased

\[ P_{f^{(1)}} = \mathbb{E}_q[P_{f}^{\text{MFIS}}] \]
**MFIS: Recipe 3**

**Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$**
- Evaluate $f^{(2)}$ at (many) realizations $z_1, \ldots, z_n$ of $Z$
- Fit mixture model $q$ (biasing) to realizations → scikit-learn, Matlab

\[ \{z_i \mid l^{(2)}(z_i) = 1, i = 1, \ldots, n\} \]

- Derive random variable $\hat{Z}$ with density $q$

**Step 2: Estimate $P_f$ with high-fidelity model $f^{(1)}$**

\[
P_f^{\text{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} \left( l^{(1)}(\hat{Z}_i) \frac{p(\hat{Z}_i)}{q(\hat{Z}_i)} \right)_{\text{uses high-fidelity}}
\]

**Multifidelity estimator $P_f^{\text{MFIS}}$ is unbiased**

\[
P_f^{(1)} = \mathbb{E}_q[P_f^{\text{MFIS}}]
\]
MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model \( f^{(2)} \)

- Evaluate \( f^{(2)} \) at (many) realizations \( z_1, \ldots, z_n \) of \( Z \)
- Fit mixture model \( q \) (biasing) to realizations \( \rightarrow \) scikit-learn, Matlab

\[
\{ z_i \mid l^{(2)}(z_i) = 1, \ i = 1, \ldots, n \}
\]

- Derive random variable \( \hat{Z} \) with density \( q \)

Step 2: Estimate \( P_f \) with high-fidelity model \( f^{(1)} \)

\[
P_f^{MFIS} = \frac{1}{m} \sum_{i=1}^{m} \frac{l^{(1)}(\hat{Z}_i)}{q(\hat{Z}_i)} \quad \text{uses high-fidelity}
\]

\[
\quad \frac{p(\hat{Z}_i)}{q(\hat{Z}_i)} \quad \text{uses low-fidelity}
\]

Multifidelity estimator \( P_f^{MFIS} \) is unbiased

\[
P_f^{(1)} = \mathbb{E}_q[P_f^{MFIS}]
\]
MFIS: Recipe 3

Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$

- Evaluate $f^{(2)}$ at (many) realizations $z_1, \ldots, z_n$ of $Z$
- Fit mixture model $q$ (biasing) to realizations → scikit-learn, Matlab

\[
\{z_i \mid I^{(2)}(z_i) = 1, i = 1, \ldots, n\}
\]

- Derive random variable $\hat{Z}$ with density $q$

Step 2: Estimate $P_f$ with high-fidelity model $f^{(1)}$

\[
P_f^{\text{MFIS}} = \frac{1}{m} \sum_{i=1}^{m} I^{(1)}(\hat{Z}_i) \frac{p(\hat{Z}_i)}{q(\hat{Z}_i)}
\]

Multifidelity estimator $P_f^{\text{MFIS}}$ is unbiased

\[
P_{f(1)} = E_q[P_f^{\text{MFIS}}]
\]
**MFIS: Recipe 3**

**Step 1: Construct biasing distribution using low-fidelity model $f^{(2)}$**
- Evaluate $f^{(2)}$ at (many) realizations $z_1, \ldots, z_n$ of $Z$
- Fit mixture model $q$ (biasing) to realizations $\rightarrow$ scikit-learn, Matlab

$$\{z_i \mid l^{(2)}(z_i) = 1, i = 1, \ldots, n\}$$
- Derive random variable $\hat{Z}$ with density $q$

**Step 2: Estimate $P_f$ with high-fidelity model $f^{(1)}$**

$$P_f^{MFIS} = \frac{1}{m} \sum_{i=1}^{m} l^{(1)}(\hat{Z}_i) \cdot \frac{p(\hat{Z}_i)}{q(\hat{Z}_i)}$$

Multifidelity estimator $P_f^{MFIS}$ is unbiased

$$P_{f^{(1)}} = \mathbb{E}_q[P_f^{MFIS}]$$
MFIS: Optimization for risk-averse designs

- Optimization
- Statistics
- Design variable
- High-fidelity model
- Uncertainty quantification
- Output $y$
- Realization $z$

Diagram:
- Optimization
- Statistics
- Design variable
- High-fidelity model
- Uncertainty quantification
- Output $y$
- Realization $z$
MFIS: Risk-averse design of wing

Consider baseline wing definition in OpenAeroStruct
- Design variables are thickness and position of control points
- Uncertain flight conditions (angle of attack, air density, Mach number)
- Output is fuel burn

Minimize fuel burn at limit states
\[ \min_{x \in \mathcal{X}} \mathbb{E}[f^{(1)}(x, Z) \mid f^{(1)}(x, Z) \leq \beta] \]

Derive a data-fit surrogate at current design \( x \)
- Take a 3 \times 3 \times 3 equidistant grid in stochastic domain
- Evaluate high-fidelity model at those 27 points at current design \( x \)
- Derive linear interpolant of output

Apply multifidelity pre-conditioned cross-entropy method
MFIS: Risk-averse design of wing

Consider baseline wing definition in OpenAeroStruct

- Design variables are thickness and position of control points
- Uncertain flight conditions (angle of attack, air density, Mach number)
- Output is fuel burn

Minimize fuel burn at limit states

\[
\min_{x \in X} \mathbb{E}[f^{(1)}(x, Z) \mid f^{(1)}(x, Z) \leq \beta]
\]

Derive a data-fit surrogate at current design \( x \)

- Take a \( 3 \times 3 \times 3 \) equidistant grid in stochastic domain
- Evaluate high-fidelity model at those 27 points at current design \( x \)
- Derive linear interpolant of output

Apply multifidelity pre-conditioned cross-entropy method
MFIS: Risk-averse design of wing (cont’d)

![Graph showing comparison between high-fidelity model alone and multifidelity methods. The x-axis represents runtime in seconds, and the y-axis represents the objective function. The graph illustrates the trade-off between model accuracy and computational cost.]
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
Uncertainty quantification tasks

1. Multifidelity uncertainty propagation

\[ f : D \rightarrow Y \]

2. Multifidelity sensitivity analysis

3. Multifidelity failure probability estimation

4. Other multifidelity uncertainty quantification tasks
Outlook: Inverse problems

Bayesian inference of parameters $z$ from data $y$

- Parameters represented as random variable $z$ with prior $p(z)$
- Define likelihood $p(y|z)$ of data $y$ using model $f$
- Update distribution of $z$ ("infer") with Bayes’ rule

$$p(z|y) \propto p(y|z) p(z)$$
Outlook: Inverse problems (cont’d)

\[ p(z|y) \propto p(y|z)p(z) \]

Posterior provides complete description of uncertainties in \( z \)
- Input to future simulations for predictions with quantified uncertainties
- Explore posterior to reduce uncertainties in future predictions

Sampling posterior \( p(z|y) \)
- Evaluate posterior expectation for function \( g \)

\[ \mathbb{E}[g] = \int g(z)p(z|y)dz \]
- Samples required as inputs in upstream simulations
- Explore posterior to decide where to take new data points
- Estimate quantiles

Making sampling tractable \( \Rightarrow \) multifidelity
Outlook: Inverse problems (cont’d)

\[ p(z|y) \propto p(y|z) p(z) \]

Posterior provides complete description of uncertainties in \( z \)
- Input to future simulations for *predictions with quantified uncertainties*
- Explore posterior to *reduce uncertainties in future predictions*

Sampling posterior \( p(z|y) \)
- Evaluate posterior expectation for function \( g \)

\[ \mathbb{E}[g] = \int g(z) p(z|y) dz \]
- Samples required as inputs in upstream simulations
- Explore posterior to decide where to take new data points
- Estimate quantiles

Making sampling tractable \( \Rightarrow \) multifidelity
Outlook: Learning surrogates for multifidelity costs

Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

⇒ Need for model reduction that targets multifidelity
Outlook: Learning surrogates for multifidelity costs

Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

⇒ Need for model reduction that targets multifidelity
Outlook: Learning surrogates for multifidelity costs

Traditional model reduction is separate from multifidelity computations

• Measures error w.r.t. HFM output while outer-loop result is goal
• Ignores that surrogates are used together with other information sources
• While approximating HFM can be hard, supporting HFM might be easy

⇒ Need for model reduction that targets multifidelity
Outlook: Learning surrogates for multifidelity

Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

⇒ Need for model reduction that targets multifidelity
Traditional model reduction is separate from multifidelity computations

- Measures error w.r.t. HFM output while outer-loop result is goal
- Ignores that surrogates are used together with other information sources
- While approximating HFM can be hard, supporting HFM might be easy

⇒ Need for learning surrogates that target multifidelity
Outlook: Learning surrogates for multifidelity

Adapt surrogate models - but not too much

- Adapting surrogate models towards multifidelity is beneficial
- Crude, cheap surrogates can have better costs/benefit ratio
- Proved for MFMC that optimal amount to spend on learning surrogates is bounded

Summary: Multi-fidelity uncertainty quantification
Summary: Multi-fidelity uncertainty quantification

Wide applicability; integrates well with machine-learning surrogates

- Applicable to general low-fidelity models such as response surfaces, coarse-grid approximations, linearized models, neural-network models
Summary: Multi-fidelity uncertainty quantification

Wide applicability; integrates well with machine-learning surrogates

- Applicable to general low-fidelity models such as response surfaces, coarse-grid approximations, linearized models, neural-network models

Accuracy guarantees; even if errors of low-fidelity models unknown

- High-fidelity model stays in the loop; same accuracy guarantees as using high-fidelity model only
- Useful in real-world applications, where typically error control for low-fidelity models such as neural-network models is unavailable
Summary: Multi-fidelity uncertainty quantification

Wide applicability; integrates well with machine-learning surrogates

- Applicable to general low-fidelity models such as response surfaces, coarse-grid approximations, linearized models, neural-network models

Accuracy guarantees; even if errors of low-fidelity models unknown

- High-fidelity model stays in the loop; same accuracy guarantees as using high-fidelity model only
- Useful in real-world applications, where typically error control for low-fidelity models such as neural-network models is unavailable

Nonintrusive technique; no re-implementation of codes necessary

- Applicable in a black-box fashion; no in-depth insight in code/implementation/theory necessary
Summary: Multi-fidelity uncertainty quantification

Wide applicability; integrates well with machine-learning surrogates

- Applicable to general low-fidelity models such as response surfaces, coarse-grid approximations, linearized models, neural-network models

Accuracy guarantees; even if errors of low-fidelity models unknown

- High-fidelity model stays in the loop; same accuracy guarantees as using high-fidelity model only
- Useful in real-world applications, where typically error control for low-fidelity models such as neural-network models is unavailable

Nonintrusive technique; no re-implementation of codes necessary

- Applicable in a black-box fashion; no in-depth insight in code/implementation/theory necessary

Embarrassingly parallel; just as regular Monte Carlo

- Evaluations of low- and high-fidelity models can often be decoupled
- Applicable as post-processing step (re-use databases of past simulations)
Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization

Benjamin Peherstorfer†
Karen Willcox‡
Max Gunzburger§

Abstract. In many situations across computational science and engineering, multiple computational models are available that describe a system of interest. These different models have varying evaluation costs and varying fidelities. Typically, a computationally expensive high-fidelity model describes the system with the accuracy required by the current application at hand, while lower-fidelity models are less accurate but computationally cheaper than the high-fidelity model. Outer-loop applications, such as optimization, inference, and uncertainty quantification, require multiple model evaluations at many different inputs, which often leads to computational demands that exceed available resources if only the high-fidelity model is used. This work surveys multifidelity methods that accelerate the solution of outer-loop applications by combining high-fidelity and low-fidelity model evaluations, where the low-fidelity evaluations arise from an explicit low-fidelity model (e.g., a simplified physics approximation, a reduced model, a data-fit surrogate) that approximates the same output quantity as the high-fidelity model. The overall premise of these multifidelity methods is that low-fidelity models are leveraged for speedup while the high-fidelity model is kept in the loop to establish accuracy and/or convergence guarantees. We categorize multifidelity methods according to three classes of strategies: adaptation, fusion, and filtering. The paper reviews multifidelity methods in the outer-loop contexts of uncertainty propagation, inference, and optimization.

Key words. multifidelity, surrogate models, model reduction, multifidelity uncertainty quantification, multifidelity uncertainty propagation, multifidelity statistical inference, multifidelity optimization

AMS subject classifications. 65-02, 62-02, 49-02

DOI. 10.1137/16M1082469
Further reading on methods covered in this talk


Books on uncertainty quantification
Software

Software for uncertainty quantification

Software with explicit multifidelity support

Summary: Multi-fidelity uncertainty quantification

Wide applicability; integrates well with machine-learning surrogates
- Applicable to general low-fidelity models such as response surfaces, coarse-grid approximations, linearized models, neural-network models

Accuracy guarantees; even if errors of low-fidelity models unknown
- High-fidelity model stays in the loop; same accuracy guarantees as using high-fidelity model only
- Useful in real-world applications, where typically error control for low-fidelity models such as neural-network models is unavailable

Nonintrusive technique; no re-implementation of codes necessary
- Applicable in a black-box fashion; no in-depth insight in code/implementation/theory necessary

Embarrassingly parallel; just as regular Monte Carlo
- Evaluations of low- and high-fidelity models can often be decoupled
- Applicable as post-processing step (re-use databases of past simulations)