# Multi-fidelity Uncertainty Quantification of a Groundwater Model

Michael J. Asher

*Australian National University, Canberra (Australia)*

John D. Jakeman

*Sandia National Laboratories, Albuquerque, NM (United States)*

Luk Peeters

*CSIRO Land and Water, Adelaide (Australia)*

**Abstract**

In order to support decision making under uncertainty, models must provide detailed predictions in a timely manner. High-fidelity, complex models allow a gamut of data and processes to be incorporated at fine temporal and spatial scales, yet their slow runtimes inhibit their use in optimization, uncertainty analysis, integrated modeling, and decision support. While lower fidelity, faster models may lack detail and accuracy, they often capture important features of the underlying system. This fact is exploited by a novel method, multi-fidelity stochastic collocation, to create an approximation of a high-fidelity model based on low-fidelity simulations. The approach is non-intrusive (does not require either model to be edited), can be applied to a diverse range of high and low-fidelity outputs and allows the efficient use of computational resources (low and high-fidelity runs can be reused and parallelized). Significant reductions in runtimes are possible in many settings, as the vast majority of necessary high-fidelity runs can be approximated at the cost of a low fidelity output. Importantly for practitioner's confidence, compared to statistically based surrogates,

[☆] Fully documented templates are available in the elsarticle package on CTAN.

*Email addresses:* michael.asher@anu.edu.au (Michael J. Asher), jdjakem@sandia.gov (John D. Jakeman), luk.peeters@csiro.au (Luk Peeters)

is the ability to compare the multi-fidelity approximation with a low-fidelity output. This paper elaborates on the method and applies various configurations to a transient, spatially distributed groundwater flow model. A notable improvement in performance is demonstrated when collocation points are chosen from normalized candidates.

*Keywords:* Uncertainty Quantification, Multi-fidelity, Multilevel, Stochastic Collocation, Monte Carlo, Groundwater Flow Model

*2010 MSC:* 49Q12

---

## 1. Introduction

Surrogate methods are a valuable tool for reducing complexity and increasing the efficiency of numerical models. Consider the generic model

$$\mathbf{u} = u(\mathbf{z}), \tag{1}$$

which takes a number of inputs $\mathbf{z} = (z_1, z_2, \cdots z_D)$ and produces a number of outputs $\mathbf{u} = (u_1, u_2, \cdots u_Q)$. Models may be run in many settings, for prediction, uncertainty quantification, sensitivity analysis, or optimization. Here the inputs are the quantities to be varied in a particular setting. Similarly, the outputs are only the quantities of interest for a particular application. All other model coefficients, parameters, data, inputs and outputs which are not to be varied for the specific application at hand are not included in $\mathbf{u}$ or $\mathbf{z}$. A complex implementation of $u$, such as a partial differential equation solver with fine resolution, allows a variety of data and processes to be included at a many temporal and spatial scales and detailed predictions to be made. However, long runtimes limit the use of such high-fidelity models in applications which require real-time results, such as decision support, or many model runs, such as optimization, uncertainty analysis or integrated modeling.

Surrogate models aim to approximate the input-output relationship of $u$ at a fraction of the computational cost. Three broad categories of surrogates exist. Data-driven methods calibrate an empirical approximation on at set of inputs,

$\{\mathbf{z}_n\}_{n\in(1,2,\cdots,N)}$ and corresponding outputs $\{\mathbf{u}_n\}_{n\in(1,2,\cdots,N)}$. Common examples are Polynomial Chaos Expansions (Xiu and Karniadakis, 2002) and Radial Basis Functions (Powell, 1987). In projection based methods the governing equations of the complex model $u$ are projected onto a basis of orthogonal vectors. Examples include Proper Orthogonal Decomposition (Sirovich, 1987) and Reduces Basis Methods (Fox and Miura, 1971). Multi-fidelity or hierarchical surrogates are constructed by simplifying the underlying physics of a model or reducing numerical resolution to create a low-fidelity surrogate (e.g. Doherty and Christensen, 2011). Many multi-fidelity approaches, for example Multi-scale Finite Element methods (Efendiev and Hou, 2007), Multilevel Monte Carlo Giles (2008) or Sparse Grid Combination Techniques Harding and Hegland (2014) prescribe iterative procedures for combining models at various resolutions.

Which surrogate methods are most promising depends, of course, on the context. (Asher et al., 2015) review techniques from all three categories. This paper focuses on methods appropriate to increasing the computational efficiency of distributed groundwater flow models for applications such as sensitivity analysis, uncertainty analysis, and calibration. Many surrogate methods are unsuitable for this application as they are unable to handle large numbers of inputs and outputs (specifically spatial fields and time series). Other approaches are intrusive (require complex model code to be edited) or have subjective structures. The time and expertise required by practitioners to use them is thus prohibitive. Data-driven surrogates in particular exhibit poor performance at new values of $\mathbf{z}$ far away from the runs used to calibrate the surrogate.

Surrogate approaches are typically developed for a particular setting. For example, Le Gratiet et al. (2014) present a multi-fidelity method specific to the estimation of Sobol sensitivity indices. Giles (2008) presents another multi-fidelity method for uncertainty propagation (estimating statistical moments of model outputs based on varying inputs). An oft overlooked consideration for practitioners is the flexibility of a surrogate method. Practical approaches should be relevant in a number of settings and allow the reuse of model runs.

Bi-fidelity stochastic collocation, the approach employed in this paper, com-

bines techniques from both Hierarchical and Projection based methods. As in the reduced basis method (e.g. Lieberman et al., 2010), the greedy algorithm is used to select a basis which spans the output space of the complex model. High-fidelity outputs can then be approximated in terms of this basis. A defining feature of the algorithm is the use of a low-fidelity model in both the basis selection and the projection required to approximate new outputs. Runtimes are hence reduced, as the vast majority of necessary high-fidelity runs can be approximated at the cost of a low fidelity simulation.

The method outperforms many surrogates in the aforementioned criteria. It is non-intrusive and uses computational resources efficiently by allowing simple parallelization, reuse of low and high fidelity model runs, and iterative addition of more high-fidelity runs to improve accuracy. A further advantage is that the low-fidelity output need not be a direct approximation of the high-fidelity output. For example a low-fidelity spatial field $\mathbf{u}^L = (u_1^L, \cdots u_{Q^L}^L)$ may be used with a high-fidelity timeseries $\mathbf{u}^H = (u_1^H, \cdots u_{Q^H}^H)$, provided the uniqueness of the $\mathbf{z}$ to $\mathbf{u}^H$ relationship is emulated by $u^L$. Where $\mathbf{u}^L$ and $\mathbf{u}^H$ do relate to similar quantities, an important advantage for practitioners, compared with statistical methods, is the ability to compare the surrogate approximation with a low-fidelity output. Our work provides anecdotal evidence that the algorithm deals with correlated inputs, a known shortcoming of methods such as polynomial chaos expansions. Significant aspects of the method we neglect in this work are the extension to more than two fidelities (Zhu et al. (2014) suggest three may be optimal) and its performance in the case of "tensor stratification" (Narayan et al., 2014), where the inputs of the low-fidelity model are a subset of the high-fidelity inputs.

One purpose of this work is to determine whether bi-fidelity stochastic collocation, applied to multiple resolutions of the same groundwater flow model, is a practical surrogate approach. The process of creating discretized numerical models from various data sources is increasingly automated, in common graphical user interfaces (eg. Groundwater Vistas Rumbaugh and Rumbaugh (2007), Visual MODFLOW FLEX Fitzpatrick (2012), Aquaveo GMS Aquaveo (2011)

or ModelMuse Winston (2009)) or purpose built tools (e.g. Vermeulen, 2013; Bhatt et al., 2014). This has created an opportunity for modelers to easily produce multiple fidelities of the same model, making the multi-fidelity methods of real practical interest. Despite the well documented problems with changing the resolution of hydrological parameters (Vermeulen et al., 2006; Neuman and Di Federico, 2003; Wen and Gómez-Hernández, 1996; Mehl and Hill, 2010), multi-fidelity groundwater surrogates have been demonstrated to be effective (Mehl and Hill, 2002; Burrows and Doherty, 2014). We aim here to investigate whether different resolutions of the same groundwater model can be combined effectively using a multi-fidelity surrogate.

Our paper is structured as follows. 2 details the bi-fidelity algorithm. 3 gives several illustrative examples of its application to trivial problems, introducing our analysis methods. 4 demonstrates our development of the method and the application to a groundwater flow model. Conclusions follow in 5.

## 2. Multi-fidelity Stochastic Collocation

Consider again the generic model (1)

$$\mathbf{u} = u(\mathbf{z}) \tag{2}$$

with uncertain or unknown inputs $\mathbf{z} = (z_1, \cdots z_D) \in I_z \subseteq \mathbb{R}^D$ and outputs $\mathbf{u} = (u_1, \cdots u_Q)$. Narayan et al. (2014) developped a bi-fidelity stochastic collocation surrogate which combines a high-fidelity implementation of (2),

$$u^H : I_z \to V^H, \tag{3}$$

and a low-fidelity implementation

$$u^L : I_z \to V^L \tag{4}$$

to produce an approximator of $u^H$ at the cost of $u^L$.

The bi-fidelity surrogate is created as follows.

1. Evaluate the low-fidelity model $u^L$ on a candidate set $\Gamma = \{\mathbf{z}_m\}_{m\in\{1,2,\cdots,M\}} \subset I_Z$, ie. compute

$$u^L(\Gamma) = \{u^L(\mathbf{z}_m)\}_{m\in(1,2,\cdots,M)}. \tag{5}$$

2. Choose an ordered subset $\gamma = \{\mathbf{z}_n\}_{n\in(1,2,\cdots,N)} \subset \Gamma$ which best spans $\Gamma$ using the Greedy algorithm. See 2.1 for details.

3. Evaluate the high-fidelity model $u^H$ on $\gamma$, ie. compute

$$u^H(\gamma) = \{u^H(\mathbf{z}_n)\}_{n\in(1,2,\cdots,N)}. \tag{6}$$

4. For any new input $\mathbf{z} \in I_z$, $\mathbf{z} \notin \gamma$, the bi-fidelity surrogate is found by projecting $u^L(\mathbf{z})$ onto $u^L(\gamma)$, then use these coordinates to estimate $u^H(\mathbf{z})$ in terms of $u^H(\gamma)$. See 2.2 for details.

An example of the above method implemented in Python is given in 19, applied to two resolutions of a one dimensional diffusion equation 20. Together with the below algorithms (8) and 1), these scripts form a complete example which readers can run on any computer with Python and a few common libraries (Numpy and Scipy) installed. We are of the opinion that very little efficiency of expression or clarity is lost by presenting 8 in Python as opposed to the more common "pseudo-code". Using a real language has the advantage of providing a ready to use example for readers to try, and perhaps more importantly, allowing the authors to test the algorithm exactly as it is presented. It is our experience that "pseudo-code" in the literature contains many errors, likely because authors cannot test it in this fashion.

As detailed in Zhu et al. (2014), the simulation cost is dominated by the precomputed $N$ low-fidelity runs making up $u^L(\Gamma)$ and above all by the $M$ high-fidelity runs $u^H(\gamma)$. Each new value $\mathbf{z} \in I_z$, $\mathbf{z} \notin \gamma$ requires the evaluation of $u^L(\mathbf{x})$. The expense of the linear algebra required for the node selection and projection will be insignificant compared to these model evaluations. A key advantage of this algorithm is that if either $M$ or $N$ is increased, the previously computed low and high-fidelity outputs can be reused. Furthermore, as the samples are independent, both $u^L(\Gamma)$ and $u^H(\gamma)$ can be evaluated in parallel

and can be reused. For example, $u^L(\Gamma)$ might be used to select the nodes, but also to provide uncertainty estimates of the outputs of $u^H$ using the multi-fidelity surrogate.

### *2.1. Selection of the collocation nodes*

We aim to choose a set of $N$ basis vectors, $u^H(\gamma)$, which minimises the distance between $\text{span}(u^H(\gamma))$ and $\text{span}(u^H(\Gamma))$. Implicit is the assumption that $\text{span}(u^H(\Gamma)) \approx V^H$. To reduce computational cost, we use $u^L$ instead of $u^H$ and treat $\Gamma$ as a finite set rather than a continuum.

The Greedy Algorithm achieves this as follows. Define a distance function between an output $v \in V^L$ and a set of outputs $W \subset V^L$

$$\begin{aligned} d^L(v, W) &= \inf_{w \in W} \|v - w\|^L \\ &= \|(I - P_W)v\|^L \\ &= \|v - \sum_{w \in W} \frac{\langle w, v \rangle}{\langle w, w \rangle} w\|^L \end{aligned} \tag{7}$$

where $P_W$ is the orthogonal projection operator onto $W$. Start with $\gamma_0 = \{\}$ and iteratively expand the set

$$\begin{aligned} \mathbf{z}_n &= \arg\max_{\mathbf{z} \in \Gamma} d^L \left( u^L(\mathbf{z}), u^L(\gamma_{n-1}) \right) \\ \gamma_n &= \gamma_{n-1} \cup \{\mathbf{z}_n\} \end{aligned} \tag{8}$$

where $u^L(\gamma_{n-1}) = \{u^L(\mathbf{z}_i)\}_{i \in (1,2,\cdots,n-1)}$. The algorithm is implemented as in 1 using the pivoted Cholesky decomposition, also known as an incomplete Cholesky decomposition or partial Gram-Schmidt orthogonalization (Hardoon et al., 2004). As in Zhu et al. (2014), we assume that the continuous inner product in (7) can be computed by a discrete dot product.

```python
from numpy import dot, array, arange, zeros, empty, argmax, finfo, sqrt
def select_nodes(V, N):
    # The columns of V are the M candidate outputs of the low-fidelity model
    # V = u^L(Γ) = [ u^L(z_1), u^L(z_2) ··· u^L(z_M) ].
    # N is number of interpolation nodes/ high-fidelity runs
    M = V.shape[1]; assert N <= M;
    # Elements of w are the norms for corresponding parameter z_m
    w = array([dot(V[:, m], V[:, m]) for m in range(M)])
    # P is the permutation vector
    P = arange(M, dtype=int)
    # L is the Cholesky factor
    L = zeros((M, N))
    r = empty((M))
    for n in range(N):
        # Choose largest norm for the next pivot/ interpolation point
        p = argmax(w[n:M]) + n
        # Avoid ill-conditioning if norm is less than machine precision
        if w[p] < 2*finfo( float ).eps:
            print 'Grammian is numerically singular...The grammian has rank %s and size %s'%(n,M)
            n -= 1
            break
        # Update indices in P and swap columns n and p of V, L, and w
        P[[n, p]] = P[[p, n]]
        V[:, [n, p]] = V[:, [p, n]]
        L[[n, p], :] = L[[p, n], :]
        w[[n, p]] = w[[p, n]]
        # Update L
        for t in range(n+1, M):
            r[t] = dot(V[:, t], V[:, n]) - sum([ L[t, j]*L[n, j] for j in range(n)])
        L[n, n] = sqrt(w[n])
        for t in range(n+1, M):
            L[t, n] = r[t]/L[n, n]
            w[t] = w[t] - L[t, n]**2
    # Truncate the Cholesky factor and permutation vector
    L = L[:n+1, :]
    P = P[:n+1]
    return P, L
    # Note that L L^T = V^T V where V is the input V with columns permuted according to P
```

Figure 1: Python code to select the interpolation nodes $\gamma$ from the candidates $\Gamma$ using a pivoted Cholesky decomposition based Greedy algorithm.

### 2.2. Bi-fidelity surrogate

For any input $\mathbf{z} \in I_z$, $\mathbf{z} \notin \gamma$, the bi-fidelity surrogate is constructed as the projection of $u^H(\mathbf{z})$ onto $u^H(\gamma)$

$$u^H(\mathbf{z}) \approx u^B(\mathbf{z}) = \sum_{n=1}^{N} c_n u^H(\mathbf{z}_n) \tag{9}$$

where $c_n$ solve

$$\langle \sum_{n=1}^{N} c_n u^L(\mathbf{z}_n), \phi \rangle^L = \langle u^L(\mathbf{z}), \phi \rangle^L, \qquad \forall \quad \phi \in u^L(\gamma). \tag{10}$$

Since only $u^L(\mathbf{z})$ is required, the computational cost is reduced by the ratio of $u^H$ to $u^L$ runtime. The discrete form of this equation algorithm is implemented in 2. Again the continuous inner product in (10) can be computed by a discrete dot product.

```
from numpy import dot
from numpy.linalg import inv, cond
def synthesis_operator(LF_selected_values, HF_selected_values, L, LF_new):
        # LF_selected_values is u^L(γ)
        # HF_selected_values is u^H(γ)
        # L is the Cholesky product from the node selection algorithm
        # LF_new is u^L(z)
        G = dot(L,L.T)
        G_inv = inv( G )
        g = dot( LF_selected_values.T, LF_new )
        c = dot( G_inv, g )
        # MF_new approximates u^H(z)
        MF_new = dot( HF_selected_values, c ).squeeze()
        return MF_new, cond(G)
```

Figure 2: Python code to compute multi-fidelity surrogate for new input.

### 2.3. Configurations

Given a high-fidelity model $u^H : I_z \to V^H$, a number of choices must be made when building a bi-fidelity surrogate.

1. A low-fidelity model $u^L : I_z \to V^L$ must be used that replicates the uniqueness of the $I_z \to V^H$ relationship. Note that $V^L$ need not approximate $V^H$.

2. Choose the size $M$ and structure (sampling scheme) of $\Gamma$ such that $u^L(\Gamma)$ spans $V^L$. $M$ is the number of low-fidelity runs.

3. Select the size $N$ of $\gamma$ such that $u^L(\gamma)$ may be selected to span $V^L$. $N$ is the number of high-fidelity runs.

4. Determine the discrete form of the inner product to be used in (7) and (10). Narayan et al. (2014) suggest that it may be possible to improve the accuracy of certain quantities of interest by weighting the inner product. We could not identify examples of this in our investigation.

In this work, we consider a customization to the algorithm whereby the columns of $u^L(\Gamma)$ are normalized for the selection of the collocation nodes. Hence the Grammian $\mathbf{G}$ cannot be computed $LL^T$, as in Narayan et al. (2014), but equivalently as $u^L(\gamma)^T u^L(\gamma)$. Models based on spatially and temporally varying partial differential equations typically result in a large number of outputs. One or more for each point in space and time. Depending on the purpose, the quantities of interest $\mathbf{u} = (u_1, \cdots u_Q)$ are chosen or computed from these. Here, we experiment (in 4) with a number of low-fidelity outputs $\mathbf{u}^L$ based on the same underlying model, in an attempt to approximate a given high-fidelity output $\mathbf{u}^H$.

## 3. Simple numerical examples

To introduce our analysis of the methods, we give a number of simple examples from the literature. We use several error metrics below to compare a set of $P$ bi-fidelity outputs $\{u^B(\mathbf{z}_i)\}_{i=(1,\cdots,P)}$ to a test set of high-fidelity outputs $\{u^H(\mathbf{z}_i)\}_{i=(1,\cdots,P)}$. Where both outputs consist of $Q$ quantities of interest, the mean $L^2$ error based on $P$ samples is calculated

$$\begin{aligned}\mathbb{E}\left(\|u^H - u^B\|_{L^2}\right) &= \frac{1}{P}\sum_{i=1}^{P} \|u^H(\mathbf{z}_i) - u^B(\mathbf{z}_i)\|_{L^2} \\ &= \frac{1}{P}\sum_{i=1}^{P} \sqrt{\frac{\sum_{q=1}^{Q}(u^H_{i,q} - u^B_{i,q})^2}{Q}} \end{aligned} \tag{11}$$

where $u^H_{i,q}$ is the $q^{th}$ quantity of interest of the bi-fidelity model evaluated at $\mathbf{z}_i$. The relative mean $L^2$, which attempts to better account for sampled outputs which vary by orders of magnitude, is defined

$$\begin{aligned}\mathbb{E}\left(\|u^H - u^B\|_{\%L^2}\right) &= \frac{1}{P}\sum_{i=1}^{P}\|u^H(\mathbf{z}_i) - u^B(\mathbf{z}_i)\|_{\%L^2} \\ &= \frac{1}{P}\sum_{i=1}^{P}\sqrt{\frac{\sum_{q=1}^{Q}(u^H_{i,q} - u^B_{i,q})^2}{\sum_{q=1}^{Q}(u^H_{i,q})^2}}. \end{aligned} \tag{12}$$

The $L^\infty$ error is

$$\|u^H - u^B\|_{L^\infty} = \max_{i\in(1,\cdots,P),q\in(1,\cdots,Q)} |u^H_{i,q} - u^B_{i,q}|. \tag{13}$$

### *3.1. Trigonometric example*

Narayan et al. (2014) illustrate the technique with approximations of the function

$$u(x,z) = g(x, z+\epsilon z^2) \triangleq \cos\left(x(z+\epsilon z^2)+1\right), \qquad (x,z)\in[-1,1]\times[0,10\pi].$$

The low-fidelity model is

$$u^L(x,z) = \sum_{k=0}^{Q^L-1} \widehat{g_k}(z)\widetilde{L_k}(x)$$

where

$$\widehat{g_k}(z) = c_k\sqrt{\frac{\pi(2k+1)}{|z|}}J_{k+1/2}(|z|), \qquad z\neq 0$$

$$c_k = \Re\left[e^{i\frac{z}{|z|}}i^k\right]$$

$J_k$ is the order-$k$ Bessel function of the first kind and $\widetilde{L_k}$ is the degree-$k$ Legendre polynomial orthonormal under the unit weight function on $[-1,1]$. Here $Q^L$ and $Q^H$ are the number of terms in the low and high-fidelity expansions respectively. The high-fidelity model has more terms ($Q^L < Q^H$) and includes a fidelity parameter $\epsilon$.

$$u^H(x,z) = \sum_{k=0}^{Q^H-1} \widehat{g_k}(z+\epsilon z^2)\widetilde{L_k}(x).$$

Convergence of the bi-fidelity surrogate is shown in 3. Here we reveal an aspect of the algorithm not obvious in previous work. While the performance of the multi-fidelity surrogate improves with the size, $N$, of $\gamma$ to a point, it degrade rapidly thereafter. We include three more plots which explain this phenomenon. The error at the interpolation nodes $\gamma$ increases almost identically to the condition number of the grammian $u^L(\gamma)^T u^L(\gamma)$. These in turn correspond to the decay of the eigenvalues of the candidates $u^L(\Gamma)$. A key limitation of the algorithm is that if too many interpolation nodes are used, redundancy begins to affect the uniqueness of the solution to the projection 2.

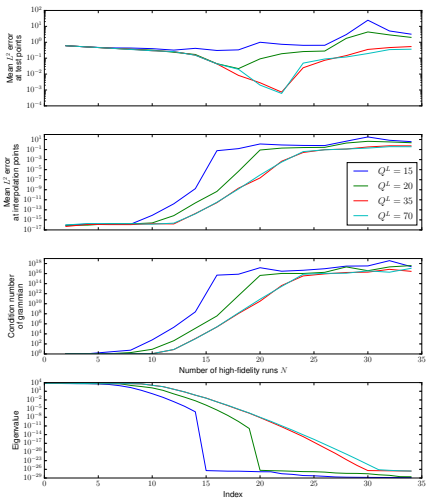


Figure 3: Bi-fidelity example from Narayan et al. (2014). Mean $L^2$ error at test points, collocation points, the condition number of Grammian and the eigenvalues of the $M = 1000$ candidate points. Errors calculated using 1000 MC tests. High and low-fidelity are evaluated at $Q = 100$ points in space. The high-fidelity model has $Q^H = 100$ terms. $\epsilon = 5 \times 10^{-3}$. The top plot is the same as Figure 1a or Narayan et al. (2014).

*3.2. 1D Diffusion with KLE using Chebyshev Collocation*

Another example from the preceding literature (Zhu et al., 2014), involves the steady state 1D diffusion equation

$$\frac{\partial}{\partial x}\left(c(x,\mathbf{z})\frac{\partial u}{\partial x}\right) = -f(x,\mathbf{z}) \tag{14}$$
$$u(0) = u(1) = 0 \qquad 0 < x < 1$$

where the quantity of interest is the density, $u(x)$, and the unknowns are the conductivity, $c(x,\mathbf{z})$, and forcing term, $f(x,\mathbf{z})$.

As in Zhu et al. (2014), we parameterize conductivity using the Karhunen-Loeve expansion

$$c(x,\mathbf{z}) = 1 + \sigma \sum_{d=1}^{D-1} \frac{1}{d\pi}\cos(2\pi dx) z_d \tag{15}$$

with $\sigma = 0.5$ and $(z_1, \cdots, z_{D-1}) \in [-1,1]^{D-1}$ is uniformly distributed. The forcing term is considered constant

$$f(x,\mathbf{z}) = 1. \tag{16}$$

Chebyshev collocation is used to solve (14). The low and high-fidelity models differ only in the number of collocation nodes used. Lagrange interpolation is used to interpolate low-fidelity outputs to the locations of the high-fidelity outputs. Convergence of the bi-fidelity surrogate is shown in 4.

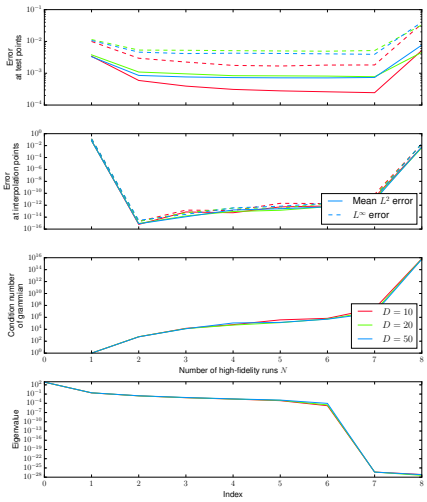


Figure 4: 1D diffusion equation with diffusivity represented by a KLE of $D-1$ terms. LF has $2^4$ Chebyshev collocation nodes, HF has $2^7$. The top figure replicates Figure 2 of Zhu et al. (2014).

*3.3. 1D Diffusion with KLE using FEM*

Another example uses the finite element method to solve (14). The forcing term is given

$$f(x, \mathbf{z}) = 50 z_D^2 \tag{17}$$

where $z_D$ normally distributed on $N(0, 1)$. The same KLE expansions as above (15) is used for conductivity, with coefficients all uniformly distributed on $U(-1, 1)$. Using a $2^9$ node solution as the high-fidelity model, 5 shows the convergence of the multi-fidelity method.

To introduce diagnostics used later, 6 plots the low $u^L(\mathbf{z})$, high $u^H(\mathbf{z})$, and multi-fidelity output $u^B(\mathbf{z})$ for a few random test inputs $\mathbf{z}$. The residuals, $u^H(\mathbf{z}) - u^B(\mathbf{z})$ and $u^H(\mathbf{z}) - u^L(\mathbf{z})$ are also plotted. 7 shows the candidates $u^L(\Gamma)$ along with the chosen $u^L(\gamma)$ and $u^H(\gamma)$. In common with the other examples in this section, and many such "toy" problems from the literature, $u^L$ is itself a close approximation to $u^H$. This motivates the use of a more "complex" example in 4.

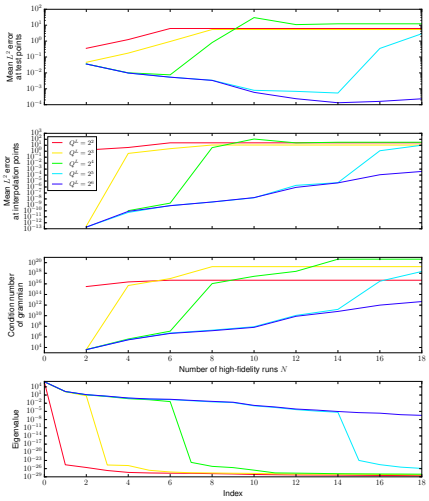


Figure 5: 1D diffusion equation, with diffusivity represented by a KLE of 10 terms and random forcing term. The solution is by the Finite Element method. High-fidelity resolution $Q^H = 2^9$.

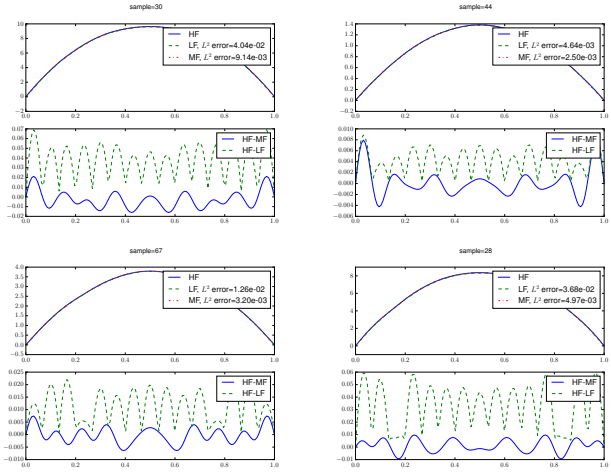


Figure 6: LF, MF and HF for four random inputs. LF resolution=$2^4$, HF runs=6, mean $L^2$ for LF=2.32e-02, mean $L^2$ for MF=7.54e-03, condition number for plots=1.19e+07 HF resolution is $2^9$.

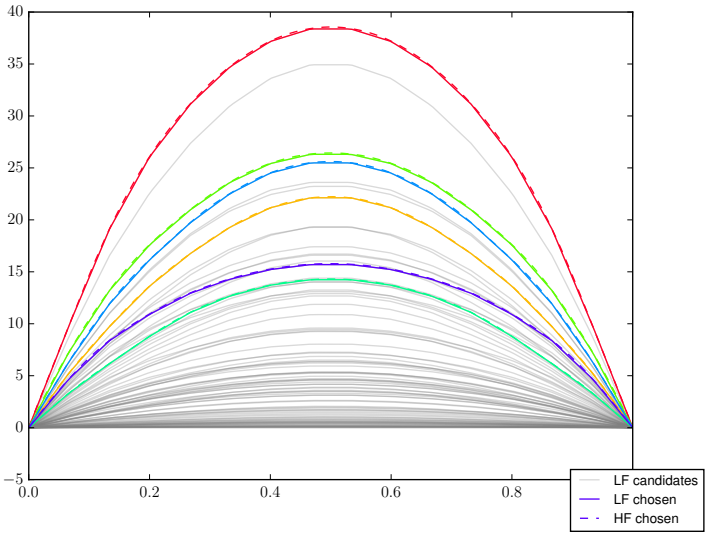


Figure 7: LF candidate outputs, and chosen LF and HF outputs.

## 4. Practical example

In this section we apply the bi-fidelity stochastic collocation algorithm to two fidelities of a groundwater flow model. We describe the model in 4.1, and then present results based on various configurations of the algorithm in 4.2 through 4.4.

### *4.1. Bi-fidelity groundwater model*

The purupose of the model used in this section is to find the maximum drawdown (decrease in pressure) in surface aquifers due to coal seam gas development. It is implemented in the 2.5D finite difference groundwater flow model code, MODFLOW Harbaugh (2005), based on the equation

$$s(\mathbf{x},\mathbf{z})\frac{\partial h(\mathbf{x},t,\mathbf{z})}{\partial t} = \nabla_{\mathbf{x}}\cdot(k(\mathbf{x},\mathbf{z})\nabla_{\mathbf{x}}h(\mathbf{x},t,\mathbf{z}))+f(\mathbf{x},t,\mathbf{z}), \qquad \mathbf{x}\in D, \quad t\in(0,T] \tag{18}$$

where $h$ is pressure, $k$ is conductivity, $s$ is specific storage and $f$ is a source sink term. Discretization is as follows. A 30 year development period with pumping to keep production wells in lower layers dry is divided into 20 timesteps, a further 60 years without pumping is divided into 30 timesteps. Vertically, the model is 1200m thick, composed of 34 layers including coal seams and confining units. A $10 \times 40$km area is divided into horizontal cells of $400 \times 400$m for high-fidelity and $2400 \times 2400$m for low-fidelity model. This results in a HF runtime which is 50 times the LF on average. Our quantity of interest, and therefore $\mathbf{u}^H$, is a drawdown timeseries for a particular location $\mathbf{x}_a$ in the top layer

$$u^H(\mathbf{z}) = \left(h(\mathbf{x}_a,0,\mathbf{z}),\cdots,h(\mathbf{x}_a,T,\mathbf{z})\right). \tag{19}$$

As noted above, $\mathbf{u}^L$ need not be the equivalent output of the low fidelity model. We explore a number of possibilities for $\mathbf{u}^L$ in 4.2 through 4.4

Much of the uncertainty in such models concerns the conductivity values of aquifer and faults. A 2D linear parameterization is employed for the acquifer properties, with $\mathbf{z} = (z_1, z_2, \cdots z_{10})$ made up of

1. anisotropy of conductivity (ratio of conductivity North-South to East-West)
2. conductance of faults
3. slope of conductivity in confining unit
4. intercept of conductivity in confining unit
5. slope of conductivity in coal seem
6. intercept of conductivity in coal seem
7. slope of storage in confining unit
8. intercept of storage in confining unit
9. slope of storage in coal seem
10. intercept of storage in coal seem.

### *4.2. Timeseries LF*

Perhaps the most straightforward configuration of the bi-fidelity algorithm is to take $u^L$ identical to $u^H$, but based on the coarser model:

$$u^L(\mathbf{z}) = (h(\mathbf{x}_a, 0, \mathbf{z}), \cdots, h(\mathbf{x}_a, T, \mathbf{z}))\,. \tag{20}$$

The convergence of this approach is given in 8. Candidates $u^L(\Gamma)$ and chosen nodes $u^L(\gamma)$ and $u^H(\gamma)$ are plotted in 9. A clearer picture of the models performance is given in 10, where we plot low, $u^L(\mathbf{z})$, high, $u^H(\mathbf{z})$, and multi-fidelity output, $u^M(\mathbf{z})$, for a 16 random test inputs $\mathbf{z}$. The residuals, $u^H(\mathbf{z}) - u^M(\mathbf{z})$ are also plotted. These 16 samples were selecte as the quantiles of high-fidelity output from 1000 test samples. As this approach reveals, the mulit-fidelity approximator performs much better where $u^H(\mathbf{z})$ is large. While the mean $L^2$ errors do not vary drastically, the mean relative errors do. This is an artifact of the greedy algorithm, which favours the selection of nodes larger in magnitude.

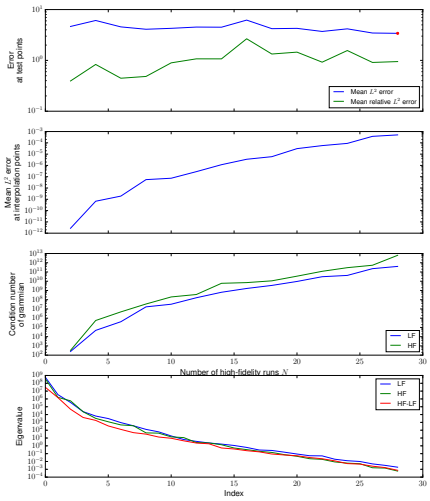


Figure 8: Error at 1000 test samples, error at collocation nodes, condition number of Grammian and eigenvalues of $u^L(\Gamma)$. High-fidelity cell size is $Q^H = 400$ meters. $Q^L = 2400$ meters. Candidate runs $M = 1000$.

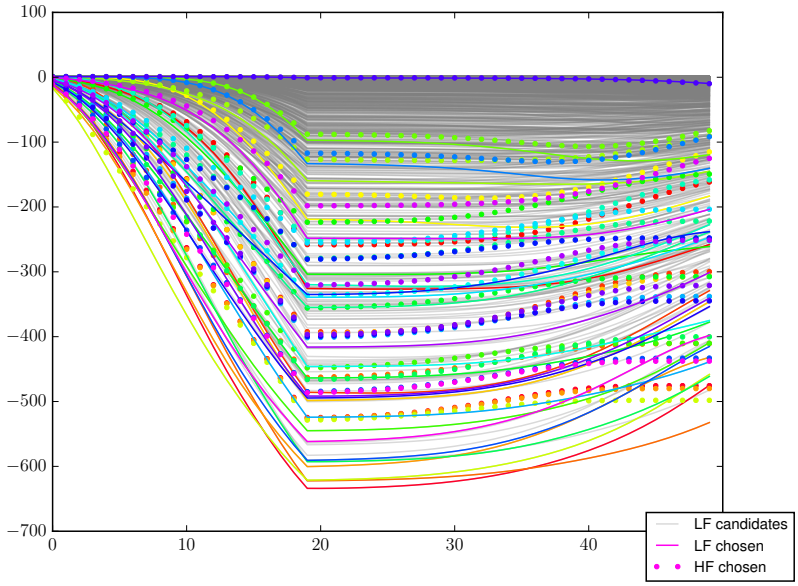


Figure 9: LF candidate outputs, and chosen LF and HF outputs.

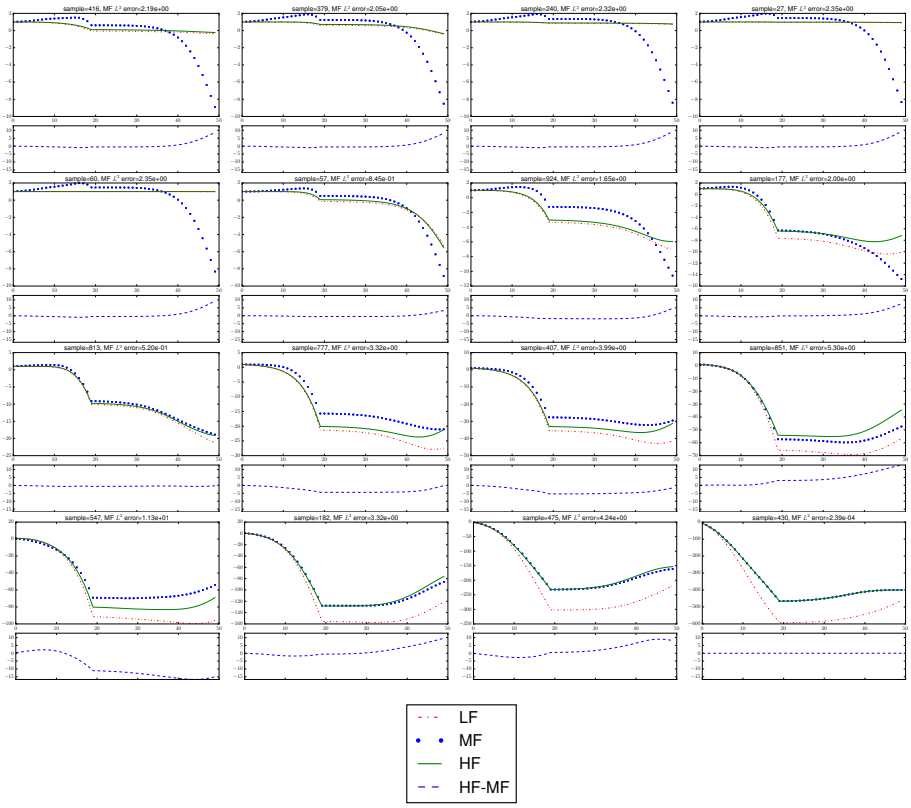


Figure 10: LF, MF and HF for 16 random inputs. HF runs=28, mean $L^2$ for LF= 8.49e+01, mean $L^2$ for MF= 3.41e+00, mean relative $L^2$ for MF= 9.48e-01, condition number for plots=4e+11, HF condition number for plots=6e+12

*4.3. Normalized timeseries LF*

Here we consider an identical formulation to 4.2, with $u^L$ again the coarse model timeseries output. However, to account for the aforementioned shortcoming of the greedy algorithm, the candidates $u^L(\Gamma)$ are normalized for the selection of the collocation nodes. Hence the Grammian $\mathbf{G}$ cannot be computed $LL^T$, but must be computed with the unnormalized $u^L(\gamma)^T u^L(\gamma)$. 11, 12 and 13 correspond to 8, 9 and 10 respectively, with a notable improvement in performance.

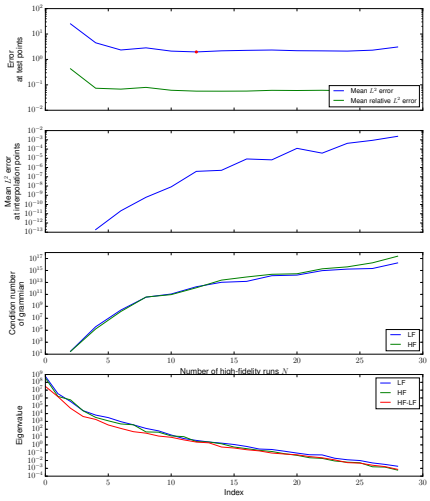


Figure 11: Error at 1000 test samples, error at collocation nodes, condition number of Grammian and eigenvalues of $u^L(\Gamma)$. High-fidelity cell size is $Q^H = 400$ meters. $Q^L = 2400$ meters. Candidate runs $M = 1000$.

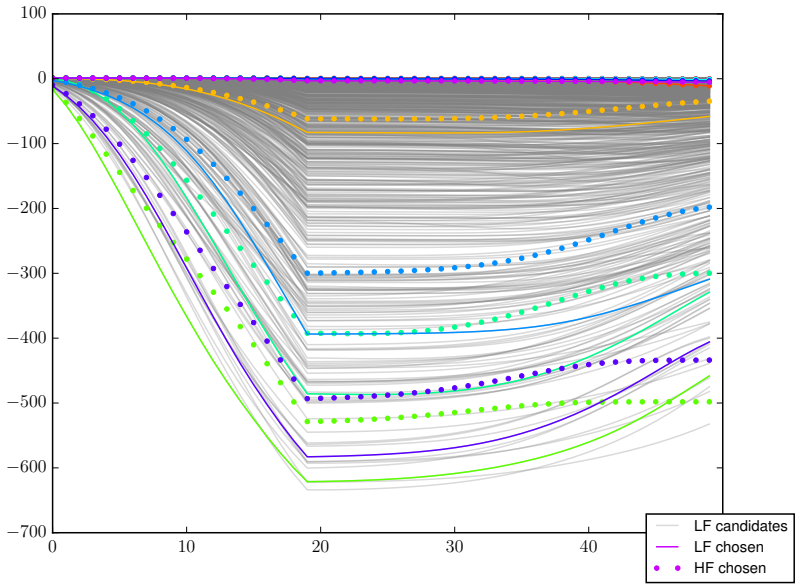


Figure 12: LF candidate outputs, and chosen LF and HF outputs.

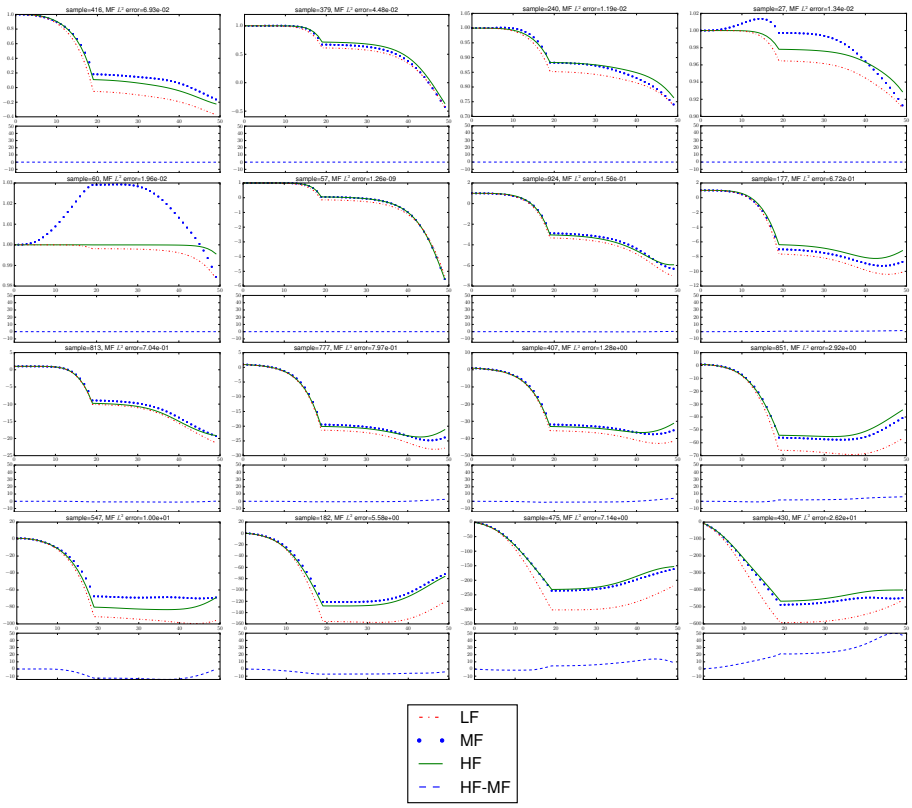


Figure 13: LF, MF and HF for 16 random inputs. HF runs=12, mean $L^2$ for LF= 8.49e+01, mean $L^2$ for MF= 1.97e+00, mean relative $L^2$ for MF= 5.65e-02, condition number for plots=2e+12, HF condition number for plots=1e+12

*4.4. Normalized spatial and timeseries LF*

In addition to normalizing the candidate vectors for node selection, we found that, in general, the algorithm improves when more information is added to $u^L$. An improvement on 4.2 was found by setting $u^L$ as the spatial output of the coarse model in the top layer for a single timestep $t_a$

$$u^L(\mathbf{z}) = (h((0,0,0),t_a,\cdots,h((0,X_0,0),t_a,\mathbf{z}),\cdots,h((0,X_0,X_1),t_a,\mathbf{z})) \quad (21)$$

where the model has $X_0$ rows and $X_1$ columns. Note that this is a different output, but is based on the same underlying model. Thus storage requirements will increase, but computational time should not. Further improvement was noted by take $u^L$ as a combination of 21 and 20, the results of which are given in 14, 15 and 16, with a notable improvement in performance.

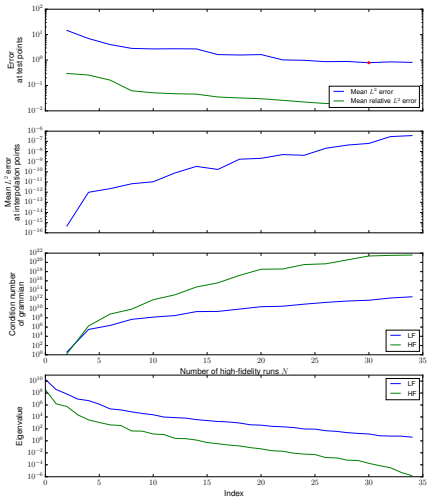


Figure 14: Error at 1000 test samples, error at collocation nodes, condition number of Grammian and eigenvalues of $u^L(\Gamma)$. High-fidelity cell size is $Q^H = 400$ meters. $Q^L = 2400$ meters. Candidate runs $M = 1000$.

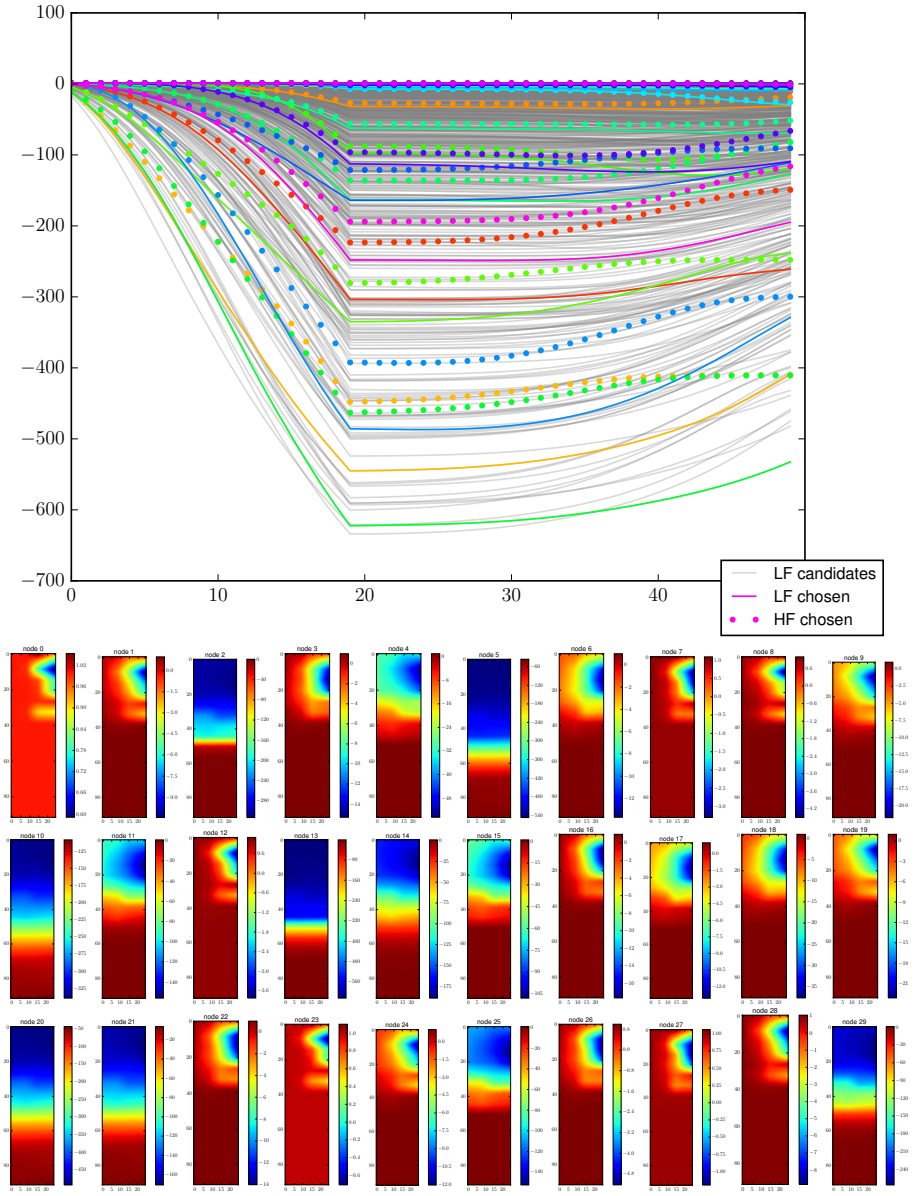


Figure 15: LF candidate outputs, and chosen LF and HF outputs.

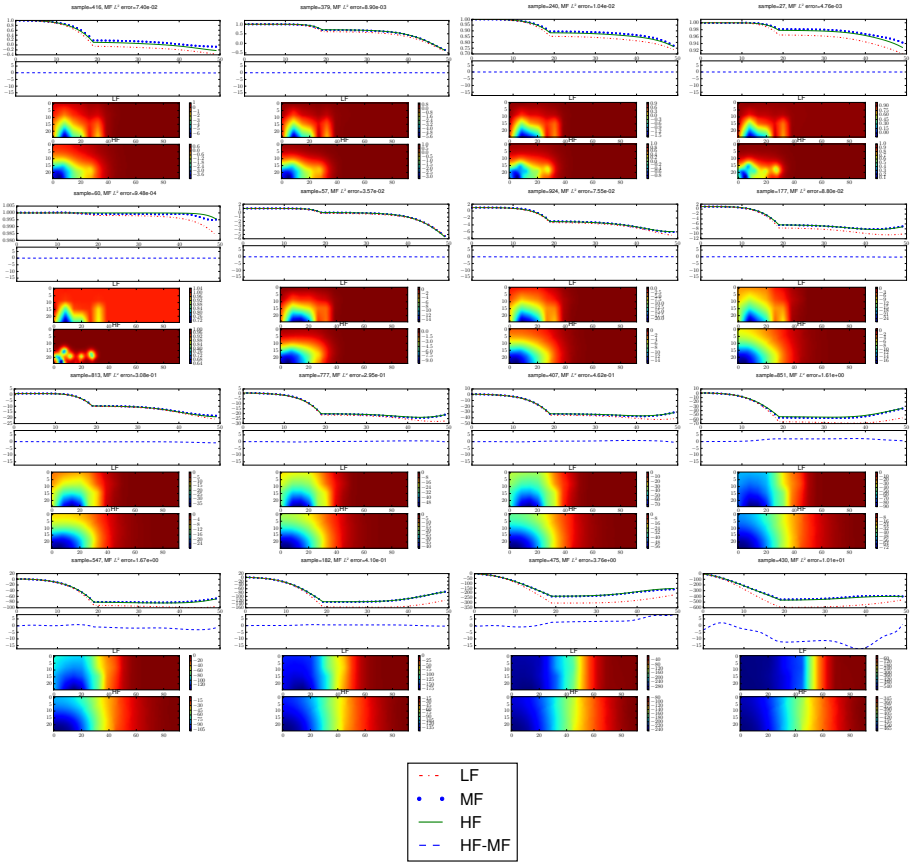


Figure 16: LF, MF and HF for 16 random inputs. HF runs=30, mean $L^2$ for LF= 1.00e+28, mean $L^2$ for MF= 7.83e-01, mean relative $L^2$ for MF= 1.76e-02, condition number for plots=6e+11, HF condition number for plots=2e+21

*4.5. Comparison with LF equal to HF*

To compare the above results with the potential of the method, we repeat the configuration in 4.4, but using the fine resolution simulator for both the low and high-fidelity model. The results are given in 17 and 18. We ommit the basis functions as 82 is an impractically large number to visualize. Note an improvement in the mean $L^2$ error from $7.83e-01$ to $1.14e-03$, and the relative error from $1.76e-02$ to $1.81e-05$.

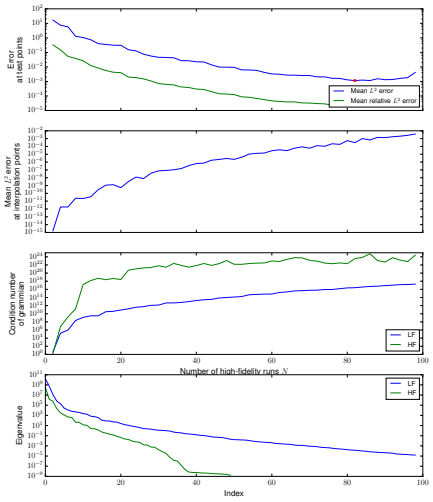


Figure 17: Error at 1000 test samples, error at collocation nodes, condition number of Grammian and eigenvalues of $u^L(\Gamma)$. High-fidelity cell size is $Q^H = Q^L = 400$ meters. Candidate runs $M = 1000$.

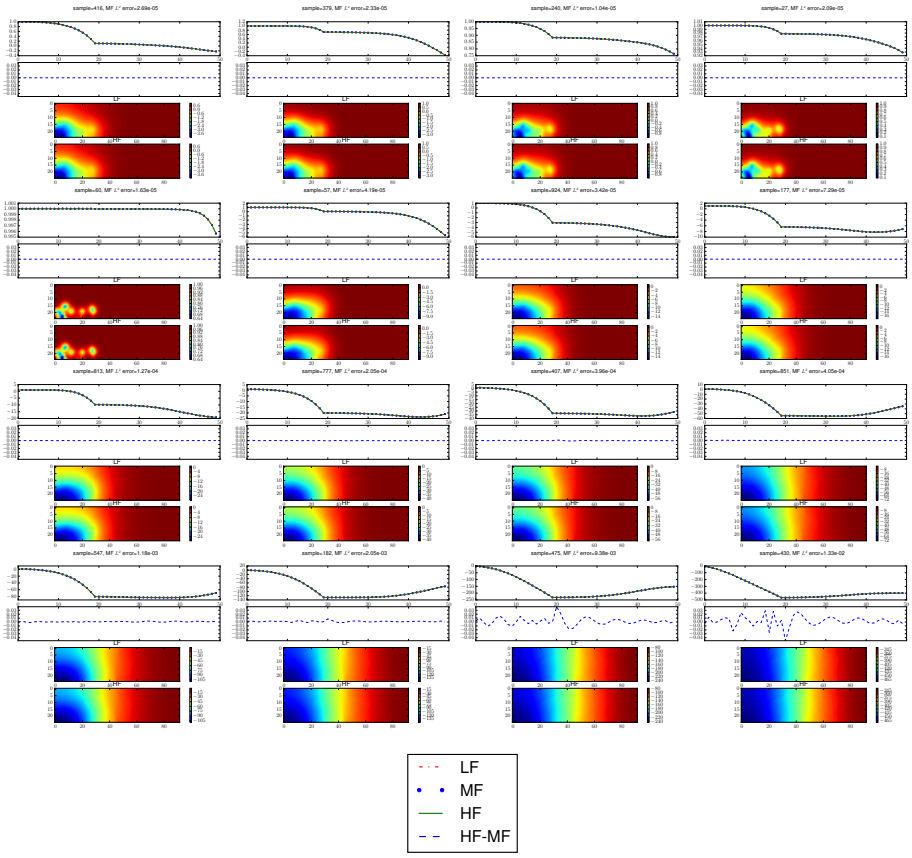


Figure 18: LF, MF and HF for 16 random inputs. HF runs=82, mean $L^2$ for LF= 1.00e+28, mean $L^2$ for MF= 1.14e-03, mean relative $L^2$ for MF= 1.81e-05, condition number for plots=3e+16, HF condition number for plots=4e+23

## 5. Conclusions

Multi-fidelity stochastic collocation is a valuable tool for efficient analysis of complex models, such as groundwater flow simulators. The method is non-intrusive and can be applied to a variety of lower-fidelity models. It allows the efficient use of computational resources, as the required samples are independent and easily parallelizable. As demonstrated by our case study (4)), as few as 10-30 high-fidelity model runs my be required to satisfactorily construct the surrogate, orders of magnitude fewer than would be required for applications such as optimization, uncertainty analysis, or decision support. Computational cost for each run required for such analysis can then be reduced to that of the low-fidelity model.

We have identified two improvements to the algorithm in practical settings. Firstly, low-fidelity outputs should be normalized for node selection so as to prevent bias towards outputs of larger magnitude. Second, as much information from the low-fidelity simulator should be included in the output as is practical. Significant improvement was noted by including more outputs of the same underlying model.

## References


Aquaveo, L., 2011. Gms 8.0 tutorials. Retrieved from Aquaveo Website (URL: http://www. aquaveo. com/gms-learning) .

Asher, M., Croke, B., Jakeman, A., Peeters, L., 2015. A review of surrogate models and their application to groundwater modeling. Water Resources Research .

Bhatt, G., Kumar, M., Duffy, C.J., 2014. A tightly coupled gis and distributed hydrologic modeling framework. Environmental Modelling and Software 62, 70–84.

Burrows, W., Doherty, J., 2014. Efficient calibration/uncertainty analysis using paired complex/surrogate models. Groundwater .

Doherty, J., Christensen, S., 2011. Use of paired simple and complex models to reduce predictive bias and quantify uncertainty. Water Resources Research 47, W12534.

Efendiev, Y., Hou, T., 2007. Multiscale finite element methods for porous media flows and their applications. Applied Numerical Mathematics 57, 577–596.

Fitzpatrick, C., 2012. Schlumberger water services launches visual modflow flex .

Fox, R., Miura, H., 1971. An approximate analysis technique for design calculations. AIAA Journal 9, 177–179.

Giles, M.B., 2008. Multilevel monte carlo path simulation. Operations Research 56, 607–617.

Harbaugh, A.W., 2005. MODFLOW-2005, the US Geological Survey modular ground-water model: The ground-water flow process. US Department of the Interior, US Geological Survey.

Harding, B., Hegland, M., 2014. Robust solutions to pdes with multiple grids, in: Sparse Grids and Applications-Munich 2012. Springer, pp. 171–193.

Hardoon, D.R., Szedmak, S., Shawe-Taylor, J., 2004. Canonical correlation analysis: An overview with application to learning methods. Neural computation 16, 2639–2664.

Le Gratiet, L., Cannamela, C., Iooss, B., 2014. A bayesian approach for global sensitivity analysis of (multifidelity) computer codes. SIAM/ASA Journal on Uncertainty Quantification 2, 336–363.

Lieberman, C., Willcox, K., Ghattas, O., 2010. Parameter and state model reduction for large-scale statistical inverse problems. SIAM Journal on Scientific Computing 32, 2523–2542.

Mehl, S., Hill, M.C., 2002. Development and evaluation of a local grid refinement method for block-centered finite-difference groundwater models using shared nodes. Advances in Water Resources 25, 497–511.

Mehl, S., Hill, M.C., 2010. Grid-size dependence of cauchy boundary conditions used to simulate stream–aquifer interactions. Advances in water resources 33, 430–442.

Narayan, A., Gittelson, C., Xiu, D., 2014. A stochastic collocation algorithm with multifidelity models. SIAM Journal on Scientific Computing 36, A495–A521.

Neuman, S.P., Di Federico, V., 2003. Multifaceted nature of hydrogeologic scaling and its interpretation. Reviews of Geophysics 41, n/a–n/a. URL: http://dx.doi.org/10.1029/2003RG000130, doi:10.1029/2003RG000130. 1014.

Powell, M.J., 1987. Radial basis functions for multivariable interpolation: a review, in: Algorithms for approximation, Clarendon Press. pp. 143–167.

Rumbaugh, J., Rumbaugh, D., 2007. Groundwater vistas.

Sirovich, L., 1987. Turbulence and the dynamics of coherent structures. part i: Coherent structures. Quarterly of applied mathematics 45, 561–571.

Vermeulen, P., 2013. imod version 2.7-interactive modeling. Manual, Deltares .

Vermeulen, P., Te Stroet, C., Heemink, A., 2006. Limitations to upscaling of groundwater flow models dominated by surface water interaction. Water resources research 42.

Wen, X.H., Gómez-Hernández, J.J., 1996. Upscaling hydraulic conductivities in heterogeneous media: An overview. Journal of Hydrology 183, ix–xxxii.

Winston, R., 2009. Modelmuse-a graphical user interface for modflow-2005 and phast: Us geological survey techniques and methods 6-a29. Reston (VA) .

Xiu, D., Karniadakis, G.E., 2002. The wiener–askey polynomial chaos for stochastic differential equations. SIAM Journal on Scientific Computing 24, 619–644.

Zhu, X., Narayan, A., Xiu, D., 2014. Computational aspects of stochastic collocation with multifidelity models. SIAM/ASA Journal on Uncertainty Quantification 2, 444–463.

## 6. Appendix

```
"""
Boilerplate for running the multi-fidelity stochastic collocation algorithm using a simple diffusion example
-------------------------------------------------------------------
authors: John Jakeman (jdjakem@sandia.gov) and Michael Asher (michael.james.asher@gmail.com)
date: August 2015
"""
import numpy
import matplotlib.pylab as plt
"""
Import
    * HF_model and LF_model: high-fidelity and low-fidelity models
    * sample_inputs: function to produce samples from input distributions (for building and testing)
    * select_nodes: function to select collocation nodes
    * synthesis_operator: function to compute surrogate
-------------------------------------------------------------------
"""
from ellip import HF_model, LF_model, sample_inputs
from select_nodes import select_nodes
from synthesis_operator import synthesis_operator
"""
Settings
-------------------------------------------------------------------
"""
# number of initial candidates/snapshots for low-fidelity model
num_lf_candidates = 100
# number of low-fidelity, multi-fidelity, and high-fidelity runs to do for testing
num_test_samples = 100
# number of interpolations nodes/high-fidelity runs
num_hf_runs = 3
"""
1. Evaluate the low-fidelity model £u^L£ on a candidate set £\Gamma£.
-------------------------------------------------------------------
"""
candidate_inputs = sample_inputs(num_lf_candidates)
lf_candidate_values = LF_model( candidate_inputs )
"""
2. Choose an ordered subset of £N£ nodes £\gamma£ using Algorithm 1.
-------------------------------------------------------------------
"""
pivots, L  = select_nodes(V=lf_candidate_values.copy(), N=num_hf_runs)
selected_inputs = candidate_inputs[:,pivots]
lf_selected_values = lf_candidate_values[:,pivots]
"""
3. Evaluate the high-fidelity £u_H£ model on £\gamma£.
-------------------------------------------------------------------
"""
hf_selected_values = HF_model( selected_inputs )
"""
4. Use £u_H(\gamma)£ to construct the interpolation operator
    and evaluate at any £z£ using Algorithm 2 with input data £v = u^L(z)£.
-------------------------------------------------------------------
"""
test_inputs = sample_inputs(num_test_samples)
lf_test_values = LF_model( test_inputs )
mf_test_values, condition_number = synthesis_operator(lf_selected_values, hf_selected_values, L, lf_test_values)
hf_test_values = HF_model( test_inputs )
print('condition number= %.2e, 1/machine eps= %.2e' % (condition_number, 1./numpy.finfo(float).eps))
print("||HF-LF|| = %.2e" % (numpy.mean(numpy.linalg.norm(hf_test_values-lf_test_values, axis=0)) / numpy.sqrt(hf_test_values.shape[0])) )
print("||HF-MF|| = %.2e" % (numpy.mean(numpy.linalg.norm(hf_test_values-mf_test_values, axis=0)) / numpy.sqrt(hf_test_values.shape[0])))
# plot a few random LF, HF, and MF samples
rand_i = numpy.random.randint(num_test_samples, size=(6))
for i in range(len(rand_i)):
  plt.subplot(numpy.ceil(len(rand_i)/3.), 3, i+1)
  plt.plot(lf_test_values[:, rand_i[i]], '--', label="lf")
  plt.plot(hf_test_values[:, rand_i[i]], '-', label="hf")
  plt.plot(mf_test_values[:, rand_i[i]], 'o', label="mf")
  plt.title('sample '+str(rand_i[i]))
plt.legend()
plt.show()
# plot all LF candidates
for l in range(num_lf_candidates):
    plt.plot(lf_candidate_values[:,l], color='grey', alpha=0.3)
# plot chosen LF and HF
cm = plt.get_cmap('gist_rainbow')
for l in range(num_hf_runs):
    plt.plot(lf_selected_values[:,l], 'x', color=cm(1.*l/num_hf_runs))
    plt.plot(hf_selected_values[:,l], color=cm(1.*l/num_hf_runs))
plt.title("Selected nodes")
plt.show()
```

Figure 19: Boilerplate for running the multi-fidelity stochastic collocation algorithm using a simple diffusion example.

```
"""
Multi-fidelity solver for a 1D elliptic PDE with random coefficients and random forcing
-----------------------------------------------------------------------
Originally from Mike Giles' MCMC code.
authors: John Jakeman (jdjakem@sandia.gov) and Michael Asher (michael.james.asher@gmail.com)
date: August 2015
2nd order central difference discretization of (c u')'(x)= - 50 Z^2, u(0)=0, u(1)=0
where Z = N(0,1)                    (Normal with unit variance)
and c(x) = 1 + a*x with a = U(0,1)  (uniformly distributed on (0,1))
"""
import numpy
import scipy.sparse
import scipy.stats
class ellip():
  # set up equation matrices
  def __init__(self, nf, num_QOI):
    hf  = 1./nf
    cf  = numpy.ones((nf))
    AOf = hf**(-2)*scipy.sparse.spdiags([cf[1:], -cf[1:]-cf[0:-1], cf[0:-1]],[-1.,0.,1.],nf-1,nf-1)
    cf  = (numpy.arange(1.,nf+1.)-0.5)*hf
    A1f = hf**(-2)*scipy.sparse.spdiags([cf[1:], -cf[1:]-cf[0:-1], cf[0:-1]],[-1.,0.,1.],nf-1,nf-1)
    self.cf  = numpy.ones((nf-1))
    self.AOf = AOf
    self.A1f = A1f
    self.num_QOI = num_QOI
  # interpolate with grid of num_QOI cells
  def post_process(self, uf):
      nf = len(uf)+1
      hf = 1./nf
      n_HF = self.num_QOI + 1
      h_HF = 1./n_HF
      # note boundary conditions (0.) are left off ends, add them here
      return numpy.interp(x=[h_HF*(i+1) for i in range(n_HF-1)], xp=[hf*(i+1) for i in range(-1, nf)], fp=numpy.concatenate(([0.0], uf, [0.0])))
  # solve equation for given random variables
  def run(self, x):
    U, Z = x
    uf = scipy.sparse.linalg.spsolve(-(self.AOf+U*self.A1f), (50.*Z**2*self.cf))
    return self.post_process(uf)
  # solve equation for each column
  def bulk_run(self, input_samples):
    num_samples = input_samples.shape[1]
    output_values = numpy.empty( (self.num_QOI, num_samples), float )
    for i in range( num_samples ):
      output_values[:,i] = self.run( input_samples[:,i] ).squeeze()
    return output_values
# set up two fidelities
n_HF  = 2**(9)
h_HF  = 1./n_HF
n_LF = 2**(2)
h_LF  = 1./n_LF
num_QOI = n_HF - 1
num_dims = 2
HF_model = ellip(nf=n_HF, num_QOI=num_QOI).bulk_run
LF_model = ellip(nf=n_LF, num_QOI=num_QOI).bulk_run
# should return (num_dims, num_samples)
def sample_inputs(num_samples):
    input_samples = numpy.random.RandomState().uniform(0.,1.,(num_dims, num_samples))
    input_samples[1,:] = scipy.stats.distributions.norm(loc=0., scale=1.).ppf(input_samples[1,:])
    return input_samples
if __name__ == '__main__':
  num_samples = 6
  zs = sample_inputs(num_samples)
  HFs = HF_model(zs)
  LFs = LF_model(zs)
  import matplotlib.pylab as plt
  for i in range(num_samples):
    plt.subplot(numpy.ceil(num_samples/3.), 3, i+1)
    plt.plot(HFs[:,i], label="HF")
    plt.plot(LFs[:,i], '--', label="LF")
  plt.suptitle("Low and high fidelity elliptic equation solutions for a few random inputs.")
  plt.legend()
  plt.show()
```

Figure 20: Multi-fidelity solver for a 1D diffusion equaion with random coefficient and forcing.