**Multilevel multifidelity Monte Carlo methods for assessing uncertainty in coastal flooding**

***Note from the Authors***

*In this document, we have included the relevant revised sections of the manuscript in framed boxes. Note all references to figures, tables and equations in these boxes, unless otherwise stated, are to the figures and equations in the new manuscript and not to those in this response document.*

***Reviewer 2 comments***

*We would like to thank Reviewer 2 for their helpful and thoughtful comments. Below we outline how we have addressed each of their concerns and suggestions.*

1. In all, the methodology, as presented, seems to be more adequate for conducting uncertainty quantification analysis and not for coastal flood risk assessment. Not much evidence is presented in the paper supporting the usefulness of the methodology in risk assessment. The estimation of cumulative distribution functions (CDFs) is just one piece of a proper risk assessment. In areas like Myrtle Beach, SC, and throughout most of the U.S. Atlantic and Gulf coasts, risk assessment must be concerned with understanding and characterizing coastal hazards (e.g., storm surge, waves, tides) due to different storm populations (e.g., tropical cyclones, extratropical storms), and must rely on some form of extreme value analysis. None of these elements are present in this paper. A methodology developed for uncertainty analysis is not necessarily transferable to coastal hazard analysis or risk assessment, therefore, I would consider revising the title of the paper as it might be inadvertently misleading.

   *As suggested, we will remove all references to risk assessment in the manuscript and replace them with references to uncertainty. This includes the title for which we propose the following:*

   > Multilevel multifidelity Monte Carlo methods for assessing uncertainty in coastal flooding

2. Although my background and experience encompass both probabilistic hazard analysis and hydrodynamic modeling, I found this paper to be quite difficult to follow. The abstract states: "Here, we apply the multilevel multi-fidelity Monte Carlo method (MLMF) to quantify uncertainty by computing statistical estimators of key output variables with respect to uncertain inputs, ..."; but there is no discussion about how are these uncertain inputs identified or prioritized. The three cases presented in the manuscript, including 2D real-world case, are highly idealized and seem to consider only one uncertain input per case; this is, Manning's coefficient, beach slope, and offshore water level, respectively.

   *We will rewrite Section 2 to make it simpler to follow the methodology. The proposed new Section 2 is included as an addendum to this document.*

   *We will also add text to each test case explaining why each uncertain parameter was selected. For the first case, we will add the following:*

   > We choose the Manning coefficient as our uncertain parameter because Bates et al. (2010) note that this test case is particularly sensitive to this parameter and thus this is a good test for our MLMF framework.

   *For the second test case, we will add the following:*

   > We choose the slope as our uncertain parameter because it represents a significant source of uncertainty, as discussed in Unguendoli (2018), particularly when simulating run-up and run-down as is the case here.

   *For the third test case we will add the following:*

   > Over the coming decades, climate change will lead to changing water levels but the actual change at specific locations is uncertain, which in turn leads to uncertainty in the impact of flooding from future storms. Thus, in this test case, we consider the offshore water level to be uncertain.

3. In real-world applications, rarely there is just one uncertain input parameter. The paper discusses how to consider multiple output locations, but is the methodology applicable when there are multiple uncertain input parameters?

   *The methodology is the same irrespective of the number of uncertain input parameters. Whilst we agree with the comment that there is normally more than one uncertain parameter, we have chosen to only use one uncertain input throughout, largely because increasing the number of uncertain inputs will increase the general uncertainty of the test case meaning larger numbers of samples are required by the Monte Carlo, the MLMC and the MLMF methods. We will add a new discussion section before the conclusion, where we discuss this, amongst other things (see response to comment 5).*

   **Discussion: Future Extensions to our MLMF methodology**

   This works aims to be a proof-of-concept demonstrating that MLMF can be used for coastal flooding. Thus, whilst in real-world cases there will be more than one uncertain input, to meet this aim it is sufficient to consider only one uncertain input parameter per test case. Adding more uncertain inputs would increase the variance of the outputs and thus all methods would require larger numbers of simulations and be more computationally expensive. Note, however, that the methodology outlined in Section 3 remains the same irrespective of the number of uncertain inputs and thus considering multiple uncertain inputs will be the subject of future work.

4. The initial case (presumably Level 0) does not seem to be well defined. Equation #14 is used to determine the optimal number of samples, but how can the initial number of samples be estimated without prior knowledge of key output variance and the input-to-output relationship?

   *It is indeed necessary to run an initial number of samples to estimate variance and cost. This is already indicated in Step 2 of the MLMC algorithm (Algorithm 1) and Step 1 of the MLMF algorithm (Algorithm 2), but we will add text to highlight this. After equation 14, we will also add the following:*

   However, this formula requires initial estimates of $\mathrm{Var}(\hat{Y}_l)$ and $C_l$ and thus we follow Giles (2008) and run 50 initial simulations (see Step 2 of Algorithm 1). To ensure this provides a good variance estimate, we also calculate the kurtosis (still following Giles (2008)). Following standard practice, if the kurtosis is greater than 100, this indicates that the variance estimate is poor and that the number of initial simulations used is insufficient. In this work, we find 50 is always sufficient but for more complex test cases, a greater number may be required. In our implementation of this algorithm, these initial simulations are stored and used as part of the optimal number of simulations in the final estimator and thus the total cost of running the algorithm is unaffected by these initial simulations (see Step 4 of Algorithm 1).

5. Also, there is no mention of other approaches that are used for similar purposes, this is, to determine the optimal number of events to be subsequently simulated using high fidelity models. Such approaches could leverage methods like Latin hypercube sampling, genetic algorithms, joint probability methods, and even recent machine learning techniques that directly account for input-output relationships. It's difficult to judge the benefits of the proposed methodology without a discussion, at least conceptually, of some of these other approaches.

   *We will add a new discussion section before the conclusion (see response to comment 3), where we will acknowledge other alternative approaches. We note here and in the text that many of these approaches can be combined with MLMF to improve upon existing approaches.*

For all methods in this work, we assess the impact of uncertain input parameters by randomly sampling values from a user-chosen distribution and then running the models with these parameter values. This again meets the aim of this work but is the simplest sampling approach. Nevertheless, the flexibility of MLMC and MLMF means that they can also be combined with other more sophisticated sampling techniques that can further reduce the number of model simulations needed. These complex techniques are out of scope for this work but we remark briefly upon them here. One such technique is Latin hypercube sampling (McKay et al., 2000) which splits the distribution into $n$ equal partitions (where $n$ is the number of samples required) and a sample is then taken from each partition. This sampling approach has been shown to improve computational efficiency when used with both a standard Monte Carlo method (McKay et al., 2000) and with MLMC (Xiong et al., 2022). Another technique is evolutionary algorithms (Vikhar, 2016), which are optimisation algorithms inspired by biological evolution that start with an initial set of samples (population) and evolve towards an optimal set. These have also been successfully combined with MLMC in Pisaroni et al. (2019) to further improve efficiency. There are also other common techniques to improve the efficiency of assessing uncertainty such as the Markov Chain Monte Carlo method (MCMC) and using machine learning techniques as emulators. As with the sophisticated sampling techniques, these can also be combined with MLMC and/or MLMF to improve the methods further: both multilevel Markov Chain Monte Carlo algorithms (Dodwell et al., 2019) and combining multifidelity samples with transfer learning to train machine learning emulators (Chakraborty, 2021) are fast growing areas of research, making them a promising avenue for further work.
We conclude this section by observing that, although there are more sophisticated techniques to assess uncertainty than that applied in this work, the flexibility of the MLMF algorithm means that it can easily be combined with other more complex statistical approaches, leveraging the advantages of both approaches. Whilst these combined approaches are beyond the scope of this work, using these techniques on coastal problems is an interesting and promising avenue for further research.

6. Several numerical models are introduced in the abstract and the manuscript introduction without defining the name (or acronym); e.g., SFINCS is not defined (Super-Fast INundation of CoastS) until the third time that the model is mentioned.

   *We will add a definition of SFINCS to the abstract and move the definition in the Introduction to where the first time SFINCS is mentioned. We will also define the acronyms for the other models where they first appear. The sentence where different high and low fidelity models are introduced in the introduction will read as follows:*

   Coastal flood modelling is therefore an ideal field on which to apply MLMF because there exist a large number of high fidelity but computationally expensive full physics models such as XBeach (Roelvink et al., 2009), SWASH (Simulating WAves till SHore) (Zijlema et al., 2011), or MIKE21 (Warren and Bach, 1992), and lower fidelity computationally cheaper reduced physics models such as SFINCS (Super-Fast INundation of CoastS) (Leijnse et al., 2021), LISFLOOD-FP (Bates et al., 2010) or SBEACH (Storm-Induced BEAch CHange) (Larson and Kraus, 1989).

7. Figure 1 mentions "Level 1", but the concept of what a level constitutes in this methodology has not been established at this stage of the manuscript. Also, in Figure 1, is the Euro symbol meant to describe typical computational costs or time associated with the different levels of fidelity and number of samples? It might help to provide additional context. In terms of order of magnitude, is each "Euro symbol" representing hours, weeks, or months?

   *We had not noticed that the concept of a level had not yet been established by Figure 1 and thus following your helpful suggestion, we will remove all references to levels in this figure. We will also improve the caption to explain the "Euro symbol" better.*

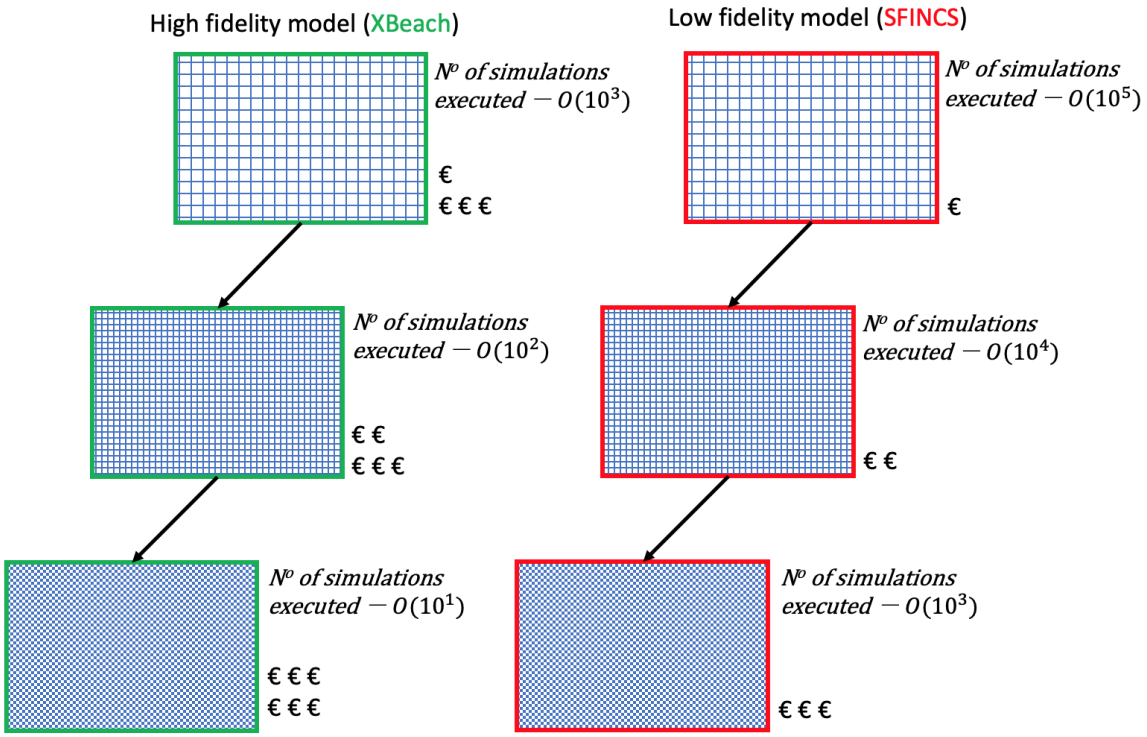


*Figure 1: Example illustration of how MLMF's multifidelity multilevel approach using SFINCS and XBeach models on different grid resolutions results in computational cost savings. Note the € symbol indicates the order of magnitude of the computational cost for a single simulation with this model at this grid resolution i.e. €€ indicates $O(10^2)$ seconds for a single simulation. The orders of time and number of scenarios are approximately those for the Myrtle Beach test case in Section 3.3.*

## References


Bates, P.D., Horritt, M.S., Fewtrell, T.J., 2010. A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. Journal of Hydrology 387, 33–45.

Chakraborty, S., 2021. Transfer learning based multi-fidelity physics informed deep neural network. Journal of Computational Physics 426, 109942.

Dodwell, T.J., Ketelsen, C., Scheichl, R., Teckentrup, A.L., 2019. Multilevel Markov chain Monte Carlo. Siam Review 61, 509–545.

Giles, M.B., 2008. Multilevel Monte Carlo Path Simulation. Operations Research 56, 607–617. doi:10.1287/opre.1070.0496.

Larson, M., Kraus, N.C., 1989. SBEACH: numerical model for simulating storm-induced beach change. Report 1. Empirical foundation and model development. Technical Report. Coastal Engineering research center Vicksburg Ms.

Leijnse, T., van Ormondt, M., Nederhoff, K., van Dongeren, A., 2021. Modeling compound flooding in coastal systems using a computationally efficient reduced-physics solver: Including fluvial, pluvial, tidal, wind- and wave-driven processes. Coastal Engineering 163. URL: https://doi.org/10.1016/j.coastaleng.2020.103796, doi:10.1016/j.coastaleng.2020.103796.

McKay, M.D., Beckman, R.J., Conover, W.J., 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42, 55–61.

Pisaroni, M., Nobile, F., Leyland, P., 2019. Continuation multilevel monte carlo evolutionary algorithm for robust aerodynamic shape design. Journal of Aircraft 56, 771–786.

Roelvink, D., Reniers, A., Van Dongeren, A., De Vries, J.V.T., McCall, R., Lescinski, J., 2009. Modelling storm impacts on beaches, dunes and barrier islands. Coastal engineering 56, 1133–1152.

Unguendoli, S., 2018. Propagation of uncertainty across modeling chains to evaluate hydraulic vulnerability in coastal areas. Ph.D. thesis. Università di Bologna, Bologna, Italy. URL: http://amsdottorato.unibo.it/8599/1/Unguendoli{_}Silvia{_}Tesi.pdf.

Vikhar, P.A., 2016. Evolutionary algorithms: A critical review and its future prospects, in: 2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC), IEEE. pp. 261–265.

Warren, I., Bach, H., 1992. MIKE 21: a modelling system for estuaries, coastal waters and seas. Environmental Software 7, 229–240.

Xiong, M., Chen, L., Ming, J., 2022. Quantify uncertainty by estimating the probability density function of the output of interest using mlmc based bayes method. Discrete and Continuous Dynamical Systems-B .

Zijlema, M., Stelling, G., Smit, P., 2011. Swash: An operational public domain code for simulating wave fields and rapidly varied flows in coastal waters. Coastal Engineering 58, 992–1012.

## 2 Methodology: Applying the multilevel multifidelity Monte Carlo method (MLMF) to assess uncertainty in coastal flooding

As discussed in Section 1, Monte-Carlo type methods can be used to assess uncertainty by estimating the expectations of functions of an input random variable. In our model scenario, the input random variable is some source of uncertainty, such as the friction coefficient, and the function involves running our numerical model and computing values such as the water elevation height at specific locations, from the model output. These estimates could be calculated using the standard Monte Carlo approach, but this is computationally expensive due to the need to run large numbers of model simulations to obtain an appropriate accuracy (see Eq. 1 and the discussion below it). The computational cost of running the model can be reduced by either coarsening the grid resolution or using a less complex model, or, in the case of this work, making use of both approaches by using the multilevel multifidelity Monte Carlo method (MLMF).

Using a coarse grid and/or simpler model gives an estimate which is cheap to compute but (more) incorrect and thus has an error. This error can be corrected by estimating the difference between the low and high fidelity models and/or the different resolutions, and adding these on to the cheaply computed expectation. Key to the approach is the observation that estimating the difference requires fewer simulations than computing the full estimate, because the variance of the correction is (hopefully) smaller than the variance of the outputs. For the different grid resolutions, the correction is done by the telescoping sum of the multilevel Monte Carlo method (MLMC), while for the different fidelity models, the correction is done by control variate formulae. The challenge is composing these approaches so that we can do both, which is what MLMF seeks to do.

The theory for MLMF is the focus of Section 2.1, whilst details on the control variate multifidelity approaches and MLMC can be found in Appendix A and B respectively. As described in Geraci et al. (2015), the standard MLMF approach cannot estimate the probability of an output variable exceeding a certain value. The latter is often also of significant interest for flooding problems and thus in Section 2.2, we present novel theory to extend MLMF for the estimation of probabilities. The implementation and application of the MLMF method in this work is then described in Section 2.3 and we conclude this methodology section with a brief remark on different methods to assess uncertainty in Section 4.

### 2.1 Multilevel multifidelity Monte Carlo method (MLMF)

MLMF seeks to improve the efficiency of uncertainty analyses by running fewer simulations at the more expensive finer resolutions than at the cheaper coarser resolutions and by running fewer high fidelity model simulations than low fidelity ones (see Figure 1). In this section, we describe the theory for the standard MLMF approach, following Geraci et al. (2015) throughout. A pictorial representation of this algorithm is shown in Figure 2 and a full statement of the algorithm is included at the end of the section.

To fix ideas, we consider a hypothetical scenario, where the variable of interest is the water elevation height at a given location after a given time and the uncertain parameter is the friction coefficient which we assume follows a normal distribution. The desired grid resolution in our model is $\Delta x = 5000/2^{10}\ (\approx 5)\,\mathrm{m}$. Note that this hypothetical scenario is similar to the example

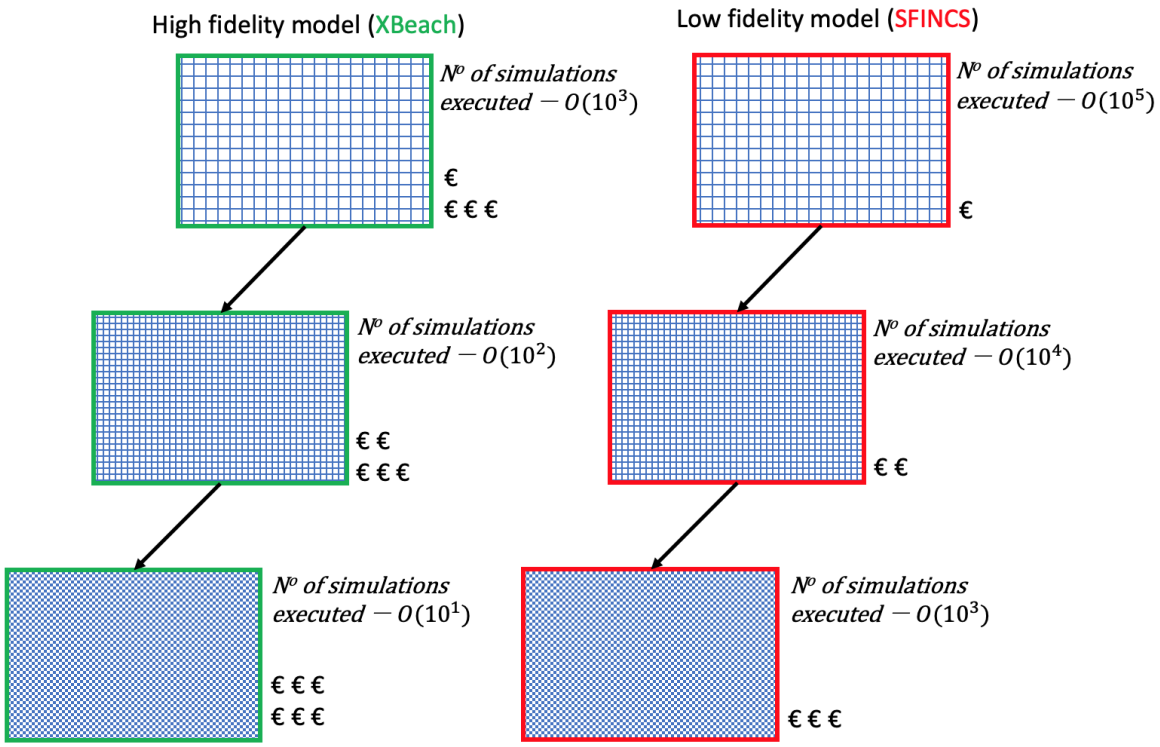


**Figure 1.** Example illustration of how MLMF's multifidelity multilevel approach using SFINCS and XBeach models on different grid resolutions results in computational cost savings. Note the € symbol indicates the order of magnitude of the computational cost for a single simulation with this model at this grid resolution *i.e.* €€ indicates $O(10^2)$ seconds for a single simulation. The orders of time and number of scenarios are approximately those for the Myrtle Beach test case in Section 3.3.

used as the first test case in this work. Moreover, throughout this work, we use HF and LF to denote the high fidelity XBeach model and low fidelity SFINCS model respectively.

We denote the MLMF estimator for the water depth at the finest grid resolution $L$ as $\hat{Q}^{HF,CV}_{M_L}$. Here the finest grid resolution is the grid resolution we would like to evaluate our model at; for our hypothetical scenario the finest grid resolution is $\Delta x = M/2^L = 5000/2^{10}\ (\approx 5)$ m. Note that following standard notation, $\hat{\cdot}$ denotes that $\hat{Q}^{HF,CV}_{M_L}$ is an estimator. An estimator represents the rule for calculating an estimate of a variable of interest given data. In our hypothetical scenario, the estimator is the rule, the variable of interest is the water elevation height, the data is our model runs and the estimate is then the numerical approximation of the mean water elevation that we obtain using our model runs. For MLMF, the rule for the estimator is a combination of the multilevel MLMC estimator (B3) with the multifidelity control variate (A1). The multilevel part of the

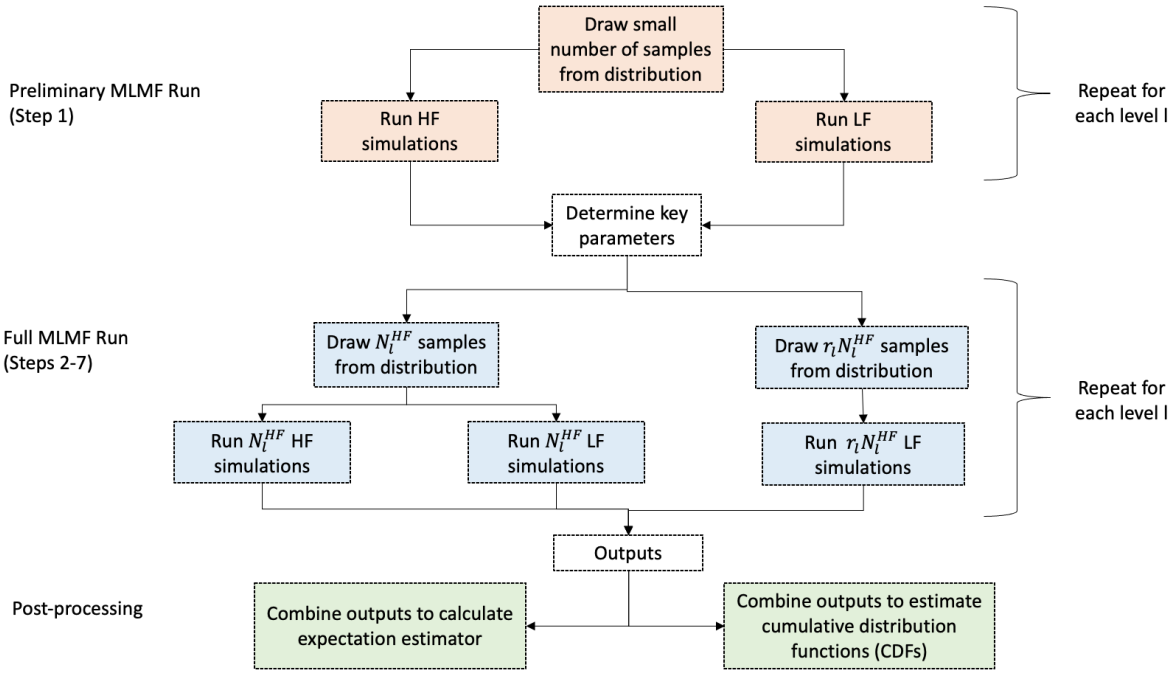


**Figure 2.** Flow chart of multi-model approach to MLMF using HF (XBeach) and LF (SFINCS).

estimator uses linearity of expectations (see (B1)) to construct the following telescoping sum

$$\hat{Q}_{M_L}^{HF,CV} = \hat{Q}_{M_{l_\mu}}^{HF,CV} + \sum_{l=l_\mu+1}^{L} \left[ \hat{Q}_{M_l}^{HF,CV} - \hat{Q}_{M_{l-1}}^{HF,CV} \right], \tag{2}$$

where $M_l$ denotes different resolutions at which the estimator is evaluated, with $l_\mu$ being the coarsest resolution. In our hypothetical scenario, the estimator is evaluated at resolutions of $[5000/2^4, 5000/2^5, 5000/2^6, 5000/2^7, 5000/2^8, 5000/2^9, 5000/2^{10}]$m. Eq. (2) finds the multilevel multifidelity estimate of water elevation at the finest resolution by calculating the multifidelity estimate at the coarsest resolution ($5000/2^4$), adding to this the difference between the multifidelity estimates at the coarsest resolution ($5000/2^4$) and the slightly finer resolution ($5000/2^5$) etc., up to and including the second finest and finest resolutions pair of $5000/2^9$ and $5000/2^{10}$. By the linearity of expectations, the sum of these differences is an estimate for the expected value of the water elevation on the finest resolution that is as accurate as simply calculating a standard Monte Carlo estimate on the finest resolution. The advantage is that calculating the estimate using this approach is less computationally expensive than using the standard Monte Carlo approach because the width of the distribution of the model outputs at each resolution $X_l$ is much larger than the width of the distribution of the *difference* between the outputs ($X_l - X_{l-1}$). Figure 3 illustrates this for two resolutions of the hypothetical scenario computed using XBeach. The narrower the distribution (*i.e.* the smaller the variance) the fewer samples are needed to estimate its mean (see Figure 9 for example). Note that the distribution of

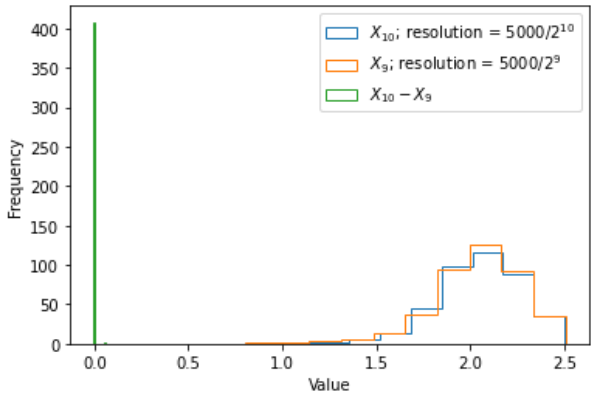


**Figure 3.** Distribution of outputs generated by XBeach for the hypothetical scenario at the finest resolution and at the second finest resolution considered, as well as the distribution of the difference between these output values. Note that the distribution for the difference between the output values is much narrower, meaning fewer samples are required to get a good estimate of the mean. For reference, in this hypothetical scenario the variance at resolution $\Delta x = 5000/2^9$m is 0.0587; the variance at resolution $\Delta x = 5000/2^{10}$m is 0.0580; and the variance of the difference between outputs at $[5000/2^9, 5000/2^{10}]$ is 9.82e-06.

the difference is very narrow in this example; for more complex cases it may be wider, although it should still remain narrower than the distribution of the individual outputs.

Combining multilevel estimators with the multifidelity control variate (A1), the full rule for the MLMF estimator is

$$\hat{Q}_{M_L}^{HF,CV} = \sum_{l=l_\mu}^{L} \left( \hat{Y}_{M_l}^{HF} + \alpha_l \left( \hat{Y}_{M_l}^{LF} - \hat{E}\left[ Y_{M_l}^{LF} \right] \right) \right), \tag{3}$$

where

$$\hat{Y}_{M_l}^{*} = \begin{cases} N_{l_\mu}^{-1} \sum_{i=1}^{N_{l_\mu}} X_{l_\mu}^{(i)} & l = l_\mu, \\ N_l^{-1} \sum_{i=1}^{N_l} \left( X_l^{(i)} - X_{l-1}^{(i)} \right) & l > l_\mu, \end{cases} \tag{4}$$

where the superscript $*$ here can indicate results from either XBeach (HF) or SFINCS (LF). In our hypothetical scenario, $X_l$ is the water elevation height from the model run using a grid resolution of $\Delta x = 5000/2^l$ m, with $X_{l-1}$ being the same but for a grid resolution of $\Delta x = 5000/2^{l-1}$ m. For each difference pair, $(i)$ denotes that the value sampled from the normal distribution for the uncertain friction coefficient is the same for both the finer resolution $X_l^i$ simulation and the coarser resolution $X_l^{i-1}$ simulation. Thus, $\hat{Y}_{M_l}^{*}$ is the mean of the difference between two model runs conducted at different resolutions with the same random number (*i.e.* value sampled from the distribution) used for friction for each pair. Moreover, for each $(i)$, the same random number is used for the XBeach model pair and the SFINCS model pair, *i.e.* the same random numbers are used to construct both $\hat{Y}_{M_l}^{HF}$ and $\hat{Y}_{M_l}^{LF}$. Note constructing estimators like this means that the coarsest level, $l_\mu$ is left without a pair and therefore $\hat{Y}_{M_{l_\mu}}^{*}$ is just the mean of the model runs conducted at the coarsest resolution. Note further that, although not strictly

necessary, here we choose to run both SFINCS and XBeach at the same resolutions, as it seems sensible to assume that this will maximise correlation between the outputs at each level.

The other terms in (3) come from the multifidelity estimator. The notation $\hat{E}[\cdot]$ denotes the estimator for the expected value – statistically speaking we cannot know the actual expected value ($\mathbb{E}$) of $Y_{M_l}^{LF}$ because this would require knowing the exact distribution of $Y_{M_l}^{LF}$. Thus, the best we can do is calculate an estimate of the expected value using data from SFINCS runs at different resolutions, *i.e.* use an estimator. This subtlety is discussed in more detail in Appendix A. Finally $\alpha_l$ is a coefficient which weights the SFINCS model outputs and is defined as

$$\alpha_l = -\rho_l \sqrt{\frac{\mathrm{Var}(\hat{Y}_{M_l}^{HF})}{\mathrm{Var}(\hat{Y}_{M_l}^{LF})}}, \tag{5}$$

where $\rho_l$ is the Pearson's correlation coefficient. In our hypothetical scenario, $\rho_l$ is the correlation between the water elevation calculated by XBeach and that calculated by SFINCS at each resolution $l$. We refer the reader to Appendix A for more details on multifidelity estimators.

To make calculating the variance of the water depth estimator simpler, we follow standard practice throughout and independently sample the values for the friction coefficient for each $\hat{Y}_{M_l}$. Hence the variance of the MLMF estimator is

$$\mathrm{Var}\left[\hat{Q}_{M_L}^{HF,CV}\right] = \sum_{l=l_\mu}^{L} \left(N_l^{HF}\right)^{-1} \mathrm{Var}\left[\hat{Y}_l^{HF}\right] \left(1 - \frac{r_l}{1+r_l}\rho_l^2\right), \tag{6}$$

using independence. Here $N_l^{HF}$ is the number of XBeach (HF) simulations required at level $l$ to compute $\hat{Y}_{M_l}^{HF}$ (which is also the number of SFINCS (LF) simulations required to compute $\hat{Y}_{M_l}^{LF}$), and $r_l$ is the factor of extra SFINCS simulations required to compute $\hat{E}\left[Y_{M_l}^{LF}\right]$. Note that, throughout this work and for simplicity, we refer to $N_l^{HF}$ as the number of XBeach simulations required, because the total number of SFINCS simulations required is the combined quantity $(1+r_l)N_l^{HF}$, and not just $N_l^{HF}$.

Because $\rho_l^2 < 1$ by definition of a correlation coefficient, equation (6) shows that the greater the correlation between the two models, the greater the reduction in the variance of the estimator. We thus seek to maximise this correlation. Geraci et al. (2017) show that, because the multifidelity control variate is unbiased, correlation can be artificially increased by modifying the estimator $\hat{Y}_l^{LF}$ using

$$\mathring{Y}_l^{LF} = \gamma_l \hat{X}_l^{LF} - \hat{X}_{l-1}^{LF}, \tag{7}$$

where the modification factor $\gamma_l$ adds an extra degree of freedom to maximise the correlation. Therefore, instead of (3), we use

$$\hat{Q}_{M_L}^{HF,CV} = \sum_{l=l_\mu}^{L} \left(\hat{Y}_{M_l}^{HF} + \alpha_l \left(\mathring{Y}_l^{LF} - \hat{E}\left[\mathring{Y}_l^{LF}\right]\right)\right), \tag{8}$$

and the new correlation coefficient $\mathring{\rho}_l^2$ is dependent on $\gamma_l$ and is equal to

$$\mathring{\rho}_l^2 = \rho_l^2 \frac{\mathrm{Cov}^2\left(\hat{Y}_l^{HF}, \mathring{Y}_l^{LF}\right)}{\mathrm{Cov}^2\left(\hat{Y}_l^{HF}, \hat{Y}_l^{LF}\right)} \frac{\mathrm{Var}\left[\hat{Y}_l^{LF}\right]}{\mathrm{Var}\left[\mathring{Y}_l^{LF}\right]}, \tag{9}$$

where we correct a typographical error in the formula given in Geraci et al. (2017). By differentiating (9) with respect to $\gamma_l$, we find the correlation is maximised when

$$\gamma_l = \frac{\mathrm{Cov}\left(\hat{Y}_l^{HF}, X_{l-1}^{LF}\right)\mathrm{Cov}\left(X_l^{LF}, X_{l-1}^{LF}\right) - \mathrm{Var}\left[X_{l-1}^{LF}\right]\mathrm{Cov}\left(\hat{Y}_l^{HF}, X_l^{LF}\right)}{\mathrm{Var}\left[X_l^{LF}\right]\mathrm{Cov}\left(\hat{Y}_l^{HF}, X_{l-1}^{LF}\right) - \mathrm{Cov}\left(\hat{Y}_l^{HF}, X_l^{LF}\right)\mathrm{Cov}\left(X_l^{LF}, X_{l-1}^{LF}\right)}. \tag{10}$$

Note that when using the modified estimator (7) the formulae previously stated in this section remain the same but $\hat{Y}_l^{LF}$ and $\rho_l$ are replaced with $\mathring{Y}_l^{LF}$ and $\mathring{\rho}_l$, respectively, in all formulae.

Finally, using (A3) and (B6), the overall cost of the MLMF algorithm (*i.e.* finding the water elevation at grid resolution $L$) is

$$C = \sum_{l=l_\mu}^{L} N_l^{HF}\left(C_l^{HF} + C_l^{LF}(1+r_l)\right). \tag{11}$$

In order to obtain the optimum values for $N_l^{HF}$ and $r_l$ in (6), we minimise this cost with respect to the variance constraint

$$\mathrm{Var}\left[\hat{Q}_{M_L}^{HF,CV}\right] < \epsilon^2/2. \tag{12}$$

which results in the following optimum formula for the factor of extra SFINCS simulations

$$r_l = -1 + \sqrt{\frac{\mathring{\rho}_l^2}{1-\mathring{\rho}_l^2}\omega_l}, \tag{13}$$

where $\omega_l = C_l^{HF}/C_l^{LF}$ is the ratio of the cost of running XBeach and SFINCS, and the following optimum formula for the number of XBeach simulations

$$N_l^{HF} = \frac{2}{\epsilon^2}\left[\sum_{k=l_\mu}^{L}\left(\frac{\mathrm{Var}\left[\mathring{Y}_k^{HF}\right]C_k^{HF}}{1-\mathring{\rho}_l^2}\right)^{1/2}\Lambda_k(r_k)\right]\sqrt{(1-\mathring{\rho}_l^2)\frac{\mathrm{Var}\left[\mathring{Y}_l^{HF}\right]}{C_l^{HF}}}, \tag{14}$$

where

$$\Lambda_k(r_k) = 1 - \frac{r_k}{1+r_k}\mathring{\rho}_k^2, \tag{15}$$

and as in (B7), $\epsilon$ should be viewed as a user-defined accuracy tolerance.

Calculating (14) requires estimates of the variance and cost. Therefore we run 50 initial simulations for each model at each resolution (see Step 1 of Algorithm 1) and use the kurtosis to check whether this provides a good enough estimate of the variance. Following Giles (2008), if the kurtosis is less than 100, then we consider our estimate of the variance to be good enough. Note further that if we are interested in the value of the variable of interest at multiple locations, $N_l^{HF}$ must be calculated separately for each location. In the algorithm, we run $\max N_l$ over all locations and then when calculating the estimator (3) at each location, subsample the optimum number for that specific location from the full output.

### 2.1.1 MLMF algorithm

Given the theory outlined above, the MLMF algorithm used in this study is summarised in Algorithm 1.

**Algorithm 1** Multilevel Multifidelity Monte Carlo method.

1: Estimate the variance and cost of the MLMF estimator, as well as the correlation and cost ratio between the HF and LF models at user-specified levels using an initial estimate for the number of simulations. The same set of random numbers must be used for the HF and LF models
2: Start with $L = l_\mu$
3: Define optimal $N_l^{HF}$ using (14) and $r_l$ using (13) with increased correlation factor (9) when required
4: If the optimal $N_l^{HF}$ is greater than the number of simulations of the HF and LF models from Step 1, evaluate the extra simulations required
5: If the optimal $r_l N_l^{HF}$ is greater than the number of simulations of the LF model after Step 4, evaluate the extra simulations of LF required
6: If the algorithm has not converged and $L < L_{\max}$, set $L$ equal to $L + 1$ and return to Step 3
7: If algorithm converged, or $L \geq L_{\max}$, STOP

## 2.2 Cumulative distribution functions

In this section so far, we have described the standard MLMF framework outlined in Geraci et al. (2015), the objective of which is to find the expectation of the output variable of interest. However, the probability of a variable exceeding a certain value is often of significant value in the study of natural hazards. This probability is complicated to estimate because MLMF computes very few values at the finest resolution from which we could build the distribution.

To resolve this, in this work we develop our own novel technique to find the cumulative distribution function (CDF) from the MLMF outputs, using a modified version of the inverse transform sampling method from Gregory and Cotter (2017). The output of a cumulative distribution function, $\mathbb{P}(X \leq x)$, is some value between 0 and 1. Returning to the hypothetical example used throughout this section, $X$ is the water elevation as a variable and $x$ is its value. For evaluating uncertainty, we would like to know the value of $x$ which the water elevation at a given location after a given time is below for $25\%$ of cases, $50\%$ of cases etc. In other words, we are interested in the inverse cumulative distribution function $F^{-1}(u)$, where $u \sim \mathcal{U}[0,1]$ and $F(x) = \mathbb{P}(X \leq x)$. If $F$ is strictly increasing and absolutely continuous, then $x \equiv F^{-1}(u)$ is unique. A simple consistent estimate for $x$ can then be found by sorting the values such that $X^1 < X^2 < ... < X^N$ and then

$$\hat{F}^{-1}(u) = X^{\lceil N \times u \rceil}. \tag{16}$$

In other words, suppose in our hypothetical scenario we have 100 values for the water elevation at a given location after a given time. Then this expression simply says that the value $x$ which the water elevation does not exceed 25% of the time, is the 25th largest value. Gregory and Cotter (2017) show that this estimate is consistent because it converges in probability to $x$

as $N \to \infty$. Note that here converges in probability means that the probability of $X^{\lceil N \times u \rceil}$ being more than a small distance $\epsilon$ from $x$ tends to zero as $N \to \infty$. In Gregory and Cotter (2017), they then use a formula to approximate $F_L^{-1}(u)$ from the MLMC outputs. In this work, we modify that formula to make it applicable for MLMF outputs so that the inverse cumulative distribution function for MLMF is approximated by

$$\begin{aligned} F_L^{-1}(u) \approx & R^{HF}(X)_{l_\mu}^{\lceil N_{l_\mu}^{HF} \times u \rceil} + \alpha_{l_\mu} \left( \mathring{R}^{LF}(X)_{l_\mu}^{\lceil N_{l_\mu}^{HF} \times u \rceil} - \hat{E}\left[ \mathring{Y}_l^{LF} \right] \right) \\ & + \sum_{l=l_\mu+1}^{L} \left( R^{HF}(X)_l^{\lceil N_l^{HF} \times u \rceil} - R^{HF}(X)_{l-1}^{\lceil N_{l-1}^{HF} \times u \rceil} \right) \\ & + \sum_{l=l_\mu+1}^{L} \alpha_l \left( \mathring{R}^{LF}(X)_l^{\lceil N_l^{HF} \times u \rceil} - \mathring{R}^{LF}(X)_{l-1}^{\lceil N_l^{HF} \times u \rceil} - \hat{E}\left[ \mathring{Y}_l^{LF} \right] \right), \end{aligned} \tag{17}$$

where $R^{HF}(X)_l^i$ and $\mathring{R}^{LF}(X)_l^i$ represent the $i^{\text{th}}$ order statistic of $X_l$ on each level $l$ of XBeach and modified correlation SFINCS (see 7), respectively. In other words, suppose that in our hypothetical scenario we want to know the value $x$ which the water elevation does not exceed 25% of the time. We then pick the lower quartile value (*i.e.* the value not exceeded 25% of the time at each resolution for both models) and add them together following the rule of the MLMF estimator. Note that, unlike with (B1), there cannot be exact cancellation because using this method means the approximations at each level are no longer unbiased.

### 2.3 Implementation

In this work, we construct our own Python MLMF wrapper around both SFINCS and XBeach to implement the MLMF algorithm. This wrapper can be shared on distributed cores of an HPC cluster to increase efficiency. Given the use of distributed cores, any times quoted in this work are the total simulation times multiplied by the number of cores used. The different steps performed when running the models in the wrapper are illustrated in the flow chart of Figure 2. Note, in particular, that in this wrapper, the models are run and post-processed separately, meaning there is no issue with different input or output formats. Therefore, our MLMF wrapper is model-independent meaning it can be easily applied to other models and applications in further work.

For the models themselves, we use XBeach version 1.23.5526 from the XBeachX release and use the surfbeat mode to simulate the waves approaching the beach (Roelvink et al., 2018). SFINCS is not yet released in the public domain, but we use a version similar to that used in Leijnse et al. (2021).