A Principled Approach to Design Using High Fidelity Fluid-Structure Interaction Simulations

Wensi Wu\textsuperscript{a,*}, Christophe Bonneville\textsuperscript{a,*}, Christopher Earls\textsuperscript{a,b}

\textsuperscript{a}School of Civil & Environmental Engineering, Cornell University, Ithaca, NY 14850, United States
\textsuperscript{b}Center for Applied Mathematics, Cornell University, Ithaca, NY 14850, United States

Abstract

A high fidelity fluid-structure interaction simulation may require many days to run, on hundreds of cores. This poses a serious burden, both in terms of time and economic considerations, when repetitions of such simulations may be required (e.g. for the purpose of design optimization). In this paper we present strategies based on (constrained) Bayesian optimization (BO) to alleviate this burden. BO is a numerical optimization technique based on Gaussian processes (GP) that is able to efficiently (with minimal calls to the expensive FSI models) converge towards some globally optimal design, as gauged using a black box objective function. In this study we present a principled design evolution that moves from FSI model verification, through a series of Bridge Simulations (bringing the verification case incrementally closer to the application), in order that we may identify material properties for an underwater, unmanned, autonomous vehicle (UUV) sail plane. We are able to achieve fast convergence towards an optimal design, using a small number of FSI simulations (a dozen at most), even when selecting over several design parameters, and while respecting optimization constraints.

Keywords: Bayesian Optimization, Fluid-Structure Interaction, Gaussian Process, Machine Learning, Design Optimization

\*Equal contribution
1. Introduction

Fluid-structure interaction (FSI) analyses have been successfully employed in increasing numbers of engineering application areas over the past decades. A few examples of such efforts include FSI modeling of heart valves [1], prediction of aerodynamics flutter around an AGARD 445.6 wing [2], parachute inflation simulation in turbulent supersonic flows [3], and slamming-induced structural response on a large container ship [4]. While the idea of effectively employing FSI analyses in design is attractive because of the enhanced insight it offers (i.e. enabling study of the interplay of a solid deforming due to forcing from a fluid medium), correctly capturing the physical response of a multi-physics system, where each sub-system is described using a distinct set of governing equations, is a nontrivial endeavor. Particularly, in a partitioned FSI approach (i.e. where the analysis context is such that the fluid and structural systems are considered using separate numerical descriptions, with external coupling strategies imposed to ensure satisfaction of salient transmission conditions along the fluid-structure interface) [5, 6, 7], a very fine temporal and spatial discretization is usually required to properly resolve the flow fields, and to correctly reproduce the physical mechanisms along the fluid-structure interface. In addition, in order to mitigate instabilities arising from the presence of the so-called added mass effects (i.e. the two-way influence of the adjacent fluid mass contacting the structure, which is especially pronounced in the context of incompressible fluids) [8, 9, 10], implicit coupling strategies are required to properly satisfy the conservation equations along the FSI interface. Hence, performing FSI analyses is rather computationally expensive: a serious problem when such simulations are required within the design cycle of some engineering artifact.

This computational cost often arises as an obstacle for optimizing structural designs subject to heavy fluid loads. Indeed, optimizing design parameters through random search, with the need to evaluate the design candidate with FSI analysis, at each new design point, would be very costly, both in time and
in computational resources. Furthermore, trying random guesses offers very little hope for effectively finding the optimal candidate, especially as the number of design parameters increases. Standard numerical optimization methods (e.g. gradient-based methods, etc.) are typically not applicable in the present case because they require convexity in the design objective function, and further, usually require many objective function evaluations. Consequently in this paper, we present design optimization strategies based on Bayesian optimization (BO): a powerful optimization method that is well suited for optimizing black box objective functions that are potentially noisy. In addition, BO is typically able to achieve quick convergence to an admissible candidate with minimal objective function evaluations. BO has been successfully applied to multiple fields of science and engineering, such as material design [11][12], aerospace [13], biology [14], drug discovery [15], and hyperparameter tuning in data sciences [16]. In the context of FSI, we particularly focus on BO with inequality constraints [17], which allows to handle more realistic design problems (e.g. if we want to reduce the drag of a hydrodynamic profile, we might want to do so while maintaining a minimal lift, maximum admissible structural deformation, etc.)

In this paper, we are proposing a principled approach to design when high-fidelity FSI simulations are required as part of the design loop. We begin with the identification of a verification problem that is “close” within the design space of our ultimate application (in our case, the Turek & Hron FSI3 benchmark [18] fits our needs). We subsequently incrementally extend the verification context to bring it ever closer to our application, using what we call Bridge Simulations. We then apply our, now trusted, FSI simulation capability to treat the case of composite material design for use in the sail plane of a UUAV, as a demonstration. If, when extending the proposed approach to other problems, the application in question does not have a suitable benchmark verification case in the literature, then one must resort to experimental validation of a simplified version of the design context: this may be advisable in all cases, if feasible.
The outline of this manuscript is as follows: Section 2 provides details concerning the FSI solution framework used in the present work, along with some elements of Bayesian optimization theory; FSI validation results, as well as bridging simulation results, are presented in Section 3 and Section 4 Section 5 provides a demonstration on how the proposed BO/FSI method can be applied to the composite design for a UUAV sail plane; and Section 6 concludes the work.

2. Methods

A depiction of the computational domain for a representative fluid-structure interaction problem is presented in Figure 1. Within the computational domain, a structural cantilever member, denoted as $\Omega_s$, is enclosed in a fluid domain, denoted as $\Omega_f$. The fluid domain (driven by an inflow defined by $u_f(t)$) interacts with the structural domain through the moving boundary $\Gamma$. Mesh deformation within the FSI system is governed by a mapping function $\chi$, which maps the mesh motion from its reference configuration $\Sigma_0$ to its deformed configuration $\Sigma(t)$.

![Figure 1: A representative computational domain within an FSI system](image)

The present work adopts a partitioned approach to FSI: relying to two open source analysis tools when treating the fluid and structural domains, respectively. The open source computational structural dynamics (CSD) finite element software used in this work is CU-BENs [19], and the open source computational
fluid dynamics (CFD) finite volume software, Open Field Operation and Manipulation (OpenFOAM 1.6-ext) [20, 21, 22, 23], is used to treat the fluid domain. Information transfer between the CFD and CSD solvers is managed using a coupling library, denoted herein as the “Coupler” [24].

2.1. CU-BENs

CU-BENs [19] is an open source structural modeling finite element solver, written in C and developed at Cornell University. CU-BENs exploits high order structural elements for use in fully nonlinear (i.e. both material and geometric contexts) transient dynamic analysis. The CU-BENs discrete Kirchhoff triangular (DKT) shell element is used in the present work, when modelling the structural responses within the FSI system. The Newmark implicit time integration scheme, augmented with generalized-α method [25], for numerical stability purposes (artificial, numerical damping can improve solution behavior), is used for time integration within our structural analyses.

2.2. OpenFOAM 1.6-ext

OpenFOAM version 1.6-ext is a comprehensive CFD solver based on the finite volume method. In many ways, OpenFOAM functions as an object-oriented application programming interface (API) for C++. In this work, we employ OpenFOAM 1.6-ext, since it provides a FEM decomposition framework [20, 23] (also known as “mini-element” method), along with a Laplacian-based mesh motion equation technique [20, 27, 26], to govern the mesh motion within the fluid domain. We found the finite element vertex-based mesh motion solver to produce superior numerical results, as compared with some of the other popular mesh motion methods (e.g. the spring analogy [28, 29] and its variants [30, 31, 32, 33]), as it guarantees boundedness in the Laplacian operator (the Laplace equation governs mesh vertex motion), even when the fluid cells approach a degenerate state [34, 20]. This feature is vital for FSI simulations
involving large structural displacements; since such structural responses could lead to considerable deformation and skewness of the attached fluid control volumes.

2.3. Coupler

The coupling library (the Coupler), initially developed by the Navel Surface Warfare Center Carderock Division (NSWCCD) \cite{24}, was modified for use in the present work. Due the extreme importance of the Coupler in FSI analyses, more detail is offered in what follows. In terms of functionality, the Coupler ensures that the salient transmission conditions are satisfied along the fluid-structure interface. Also, the Coupler manages data transfer between the CSD and CFD models. To effect communication with the Coupler, solver interface classes (i.e. CFD interface and CSD interface) are written for OpenFOAM 1.6-ext and CU-BENs. Specifically, these C++ interface classes work with the respective CSD and CFD solvers to: prepare necessary data structures; adapt cross domain variables; and translate native input/output to a form that is compatible with the Coupler. The interface classes are subsequently translated to “Python-ized” modules using the Python wrapper generator: SWIG. The resulting python files are then integrated into the Coupler. This allows the interface classes to communicate with the Coupler using pointers to memory; thus eliminating the need for writing and reading input/output files. A schematic representation of our implicit partitioned FSI solver coupling framework is presented in Figure \ref{fig:coupler}.

Within a given FSI simulation, it is the CFD solution that is most demanding, both in terms of time step size (time increment) and spatial discretization. The CSD solver can accommodate any time step size (due to its implicit time integration) and requires much coarser meshes (as a result of the high order elements). The orchestration of the FSI solutions, within each time step, follows the general scheme:
1. OpenFOAM computes the hydrodynamic loading, given the structural deformation from the previous time step, using an *Arbitrary Lagrangian Eulerian (ALE)* scheme.

2. The CFD interface object takes in the resulting fluid fields and prepares necessary data for the Coupler.

3. The Coupler interpolates the hydrodynamic loading, and subsequently maps the loading onto the structural mesh.

4. CU-BENs then computes the corresponding structural deformation.

5. The CSD interface class takes in the resulting FSI boundary deformation and prepares necessary data for the Coupler.

6. The Coupler checks for convergence by monitoring out of balance displacement and/or pressures between the fluid and structural domains: comparing to a set of user specified tolerances. If the response along the interface converges, the simulation advances to the next time step. Otherwise, the predicted displacements are passed to OpenFOAM, to update the mesh configuration, and the process is repeated.

![Diagram](image)

Figure 2: Implicit partitioned FSI coupling framework with OpenFOAM as the fluid solver, CU-BENs as the structural solver, and a Coupler that facilitates all necessary communications between the two solvers to ensure the fluid-structure interface conforms to the transmission conditions

Nonconforming mesh projection \cite{35} is utilized to guide the grid-to-grid mesh mapping and updating along the fluid-structure interface. This technique al-
allows for an optimal choice of mesh, for resolving the physics within the respective structural and fluid domains, as it provides flexibility for load and motion transferring on the FSI interface, when the mesh sizes in each subsystem are incongruous. Figure 3 provides a visual representation of nonconforming mesh in a FSI analysis. The structural domain is discretized with DKT shell elements; thus thickness is implied as a shell property (and not explicitly modeled). In such a context, the shell elements (green) are situated at the mid plane of the physical structure, and enclosed by the fluid boundary (grey). Orthogonal projection mapping is initiated at the beginning of a FSI simulation, to identify appropriate coupling pairs within the structural and fluid node sets; thus enabling mesh motion tracking throughout the analysis. It is typically the case in FSI analyses that the number of CFD nodes, occurring along the FSI interface, far exceeds the number of CSD nodes; thus leading to non-conforming mesh geometries that complicate data transfer in the FSI transmission conditions. The Coupler uses a standard approach when handling this case.

Within our Coupler, inverse distance weighting interpolation (IDW) is used to convert hydrodynamic pressure, located at the center of each fluid patch (one face of the finite volume), along the interface, into point loads, before they are projected onto the structural mesh. The IDW procedure used in the current work, employs the following relation:

\[
F(x) = \sum_{i=1}^{n} \omega_i(x) F(x_i), \quad \omega_i(x) = \frac{1}{d(x, x_i)^\alpha},
\]

where \(d(x, x_i)\) is the distance between points \(x\) and \(x_i\).
where \( x_i \) denotes the spatial coordinate of cell center \( i \), \( F(x_i) \) represents the fluid pressure at \( x_i \), \( n \) is the total number of nodal points used in the interpolation, \( \omega_i \) stands the weighting factor, \( d(x, x_i) \) corresponds to the distance between the node \( x \) and cell center \( x_i \), \( \alpha \) is the distance-decay parameter, and \( F(x) \) is the estimated point load at location \( x \). Typically, \( \alpha = 2 \) is used in the standard IDW. A simple representation of the fluid interface is shown in Figure 4 to help visualize the IDW interpolation process.

Time step advancement, and solution convergence on the fluid-structure interface, are controlled by the Coupler using an iterative method: the IQN-ILS iteration scheme [37, 7]. The IQN-ILS is a quasi-Newton root finding method, within which Newton-Raphson iterations are used, within a given time step, to solve for the interface displacement \( R(d) = S(F(d)) = 0 \). In the equation mentioned, \( F() \) represents a CFD solver for a given displacement \( d \), while, \( S() \) is a CSD solver for a given loading resulting from the CFD solver. The Newton-Raphson iteration formulation within each time step is shown in Equation [2] and [3]

\[
R^k + R^k \Delta d^k = 0
\]  
\[ \text{(2)} \]

\[
d^{k+1} = d^k + \Delta d^k
\]  
\[ \text{(3)} \]

Combining Equation [2] and [3] a new expression for \( d^{k+1} \) is derived

\[
d^{k+1} = d^k - \left( \frac{\delta R^k}{\delta d} \right)^{-1} R^k,
\]  
\[ \text{(4)} \]
where $R'_{k}$ represents the Jacobian of the interface displacement residual operator. Moreover, the residual vector $R^k$, which is evaluated at $d^k$, is described in Equation 5

$$R^k = S(F(d^k)) - d^k = \tilde{d}^{k+1} - d^k. \quad (5)$$

Herein, $\tilde{d}^{k+1}$ represents the predicted FSI interface displacement at iteration $k + 1$, whereas $d^{k+1}$ is the corrected FSI interface displacement at iteration $k + 1$. The Newton-Raphson iteration step is repeated until $R^k$ satisfies a user specified convergence criterion $\|R^k\| \leq \epsilon$, where $\epsilon$ is a convergence tolerance (standard Euclidean norm, implied).

The procedure of IQN-ILS may seem easy enough when the exact Jacobian of $R$ is known. However, in many cases, the Jacobians of the fluid and structural domains are not readily available; as in the case of coupling black box CFD/CSD pairs. In such cases, inverting $R'_{k}$, to compute $\Delta d^k$, is no longer a trivial task. Fortunately, we can circumvent the problem by constructing a set of basis vectors on $R$ and $\tilde{d}$, such that the solution to $(\frac{\partial R^k}{\partial d})^{-1}R^k$ can be approximated (i.e. the inverse of the Jacobian does not need to be formed explicitly). A sketch of this procedure follows next.

When approximating the required Jacobian, a series of simple steps are used, such that their collective forms the required linearization within the IQN-ILS algorithm. First, the difference of residual and interface displacement, from the previous iteration (denoted as superscript $k - 1$) and the current iteration (denoted as superscripted $k$), are computed for iteration $i = 0, 1, 2, 3, ..., k - 1$.

$$\Delta R^{k-1} = R^k - R^{k-1} \quad (6)$$

$$\Delta \tilde{d}^{k+1} = \tilde{d}^{k+1} - \tilde{d}^k \quad (7)$$

Each $\Delta R^i$ corresponds to a $\Delta \tilde{d}^{i+1}$, and these vectors are stored as columns
within the matrices $V^k$ and $W^k$

$$V^k = \begin{bmatrix} \Delta R^{k-1} & \Delta R^{k-2} & \cdots & \Delta R^1 & \Delta R^0 \end{bmatrix}$$ (8)

$$W^k = \begin{bmatrix} \Delta \tilde{d}^k & \Delta \tilde{d}^{k-1} & \cdots & \Delta \tilde{d}^2 & \Delta \tilde{d}^1 \end{bmatrix}$$ (9)

where the number of columns, determined by the number of iterations, $q$, is limited by the degrees-of-freedom, $p$, on the FSI interface, such that the system is over-determined. In the event that the number of iterations exceeds the cardinality in degrees-of-freedom along the FSI interface, the rightmost column of $V^k$ must be truncated, to preserve the over-determined characteristic of the system: this allows the system to be solved as a least square problem [38].

Subsequently, a set of orthogonal basis vectors is constructed, where the value of $\Delta R^k$ is approximated as a linear combination of the known $\Delta R^i$, as:

$$\Delta R^k \approx \sum_{i=0}^{k-1} \alpha_i^k \Delta R^i = V^k \alpha^k$$ (10)

with $\alpha_i^k \in \mathbb{R}^{q \times 1}$. The matrix $V^k$ is then decomposed via economy QR decomposition, using Householder transformations

$$V^k = Q^k R^k$$ (11)

into an orthogonal matrix $Q^k \in \mathbb{R}^{p \times q}$, and an upper triangular matrix $R^k \in \mathbb{R}^{q \times q}$. Substituting Equation (11) into Equation (10) yields

$$R^k \alpha^k = (Q^k)^T \Delta R^k$$, (12)

where the coefficient vector $\alpha^k$ in Equation (12) can be obtained by backward substitution.

Similarly, the corresponding $\Delta \tilde{d}^{k+1}$ can also be calculated as a linear combination of the $\Delta \tilde{d}^i$, from previous iterations,

$$\Delta \tilde{d}^{k+1} \approx \sum_{i=0}^{k-1} \alpha_i^k \Delta \tilde{d}^i = W^k \alpha^k.$$ (13)
From Equations 3, 5, 6 and 7, \( \Delta R^k = \Delta \tilde{d}^{k+1} - \Delta d^k \) is derived, this leads to the relation

\[
\Delta d^k = W^k \alpha^k - \Delta R^k.
\]  

(14)

Given that \( \Delta d^k = -(\frac{\partial R^k}{\partial d})^{-1} R^k \) and \( \Delta R^k = -R^k \) (assuming in an ideal case \( R^{k+1} = 0 \)), the quasi-Newton Raphson iteration step in Equation 4 is then re-expressed as

\[
d^{k+1} = d^k + W^k \alpha^k + R^k.
\]  

(15)

2.4. Bayesian Optimization

We now discuss the second critical piece within our FSI design “puzzle”: Bayesian optimization. Bayesian optimization (BO) is a powerful global optimization framework that permits the minimization of complex, non-convex objective functions [39, 40, 41]. Unlike gradient based optimization methods, BO does not require calculation of gradients to the objective function, and typically only requires few evaluations of the objective function, in order to achieve satisfactory convergence towards the global minimum. Consequently, it is a method well suited for minimizing (noisy) black box functions that are expensive to evaluate (e.g. experimental evaluations, computationally intensive evaluations, etc.) The general optimization problem may be stated as follows:

\[
\text{find } \hat{f} = \min_{x \in \Omega} f(x) \quad \text{s.t. } c_k(x) \leq \lambda_k \quad (k \in [1, n_c])
\]  

(16)

Where \( f \) is the objective function, \( \hat{f} \) its global minimum, \( \Omega \) the domain of definition of \( f \), and \( c_k \) is a set of \( n_c \) inequality constraints (\( c_k \) may be a black box function that is evaluated at the same time as \( f \)) [17].

The main idea of BO is to approximate \( f \) with a Bayesian regression surrogate that interpolates on a set of already known values of \( f \). The Bayesian regression is vital, since it outputs “confidence intervals” on its prediction; thus providing useful hints concerning the veracity of our interpolations of the objective function. Additionally, this built-in uncertainty quantification on the
objective function allows for a principled formulation for the requisite acquisition function; used in guiding our search for a minimum towards the most promising regions of the design space. Evaluating $f$ within the regions of design space that are specified by the acquisition function provides new data that are subsequently used to refine the Bayesian regression surrogate, and update the acquisition function prediction [41]. Iterating this process usually yields convergence with only a small number of objective function evaluations.

In the following subsections, we first provide more details about Gaussian processes (GP) [42], which is the most widely used Bayesian regression method in BO. Then, we provide more insights concerning acquisition functions and constrained Bayesian optimization (cBO) [17].

2.4.1. Gaussian Process Regression

In this brief introduction of GP regression, we employ the same notations as in [43]. Let us consider a machine learning training dataset made of $n$ input-output couples (training data): $D = (X, y) = (x_i, y_i)_{i \in [1,n]}$. $x_i$ is an input vector (e.g. the optimization/design parameters in the case of BO) and $y_i$ a collection of output scalars (e.g. the corresponding value of the objective function in the case of BO). $y_i$ is assumed to be contaminated with Gaussian white noise (other noise models can be accommodated) and modeled as follows:

\[
\begin{align*}
    y_i &= f(x_i) + \epsilon_i \\
    \epsilon_i &\sim \mathcal{N}(0, \sigma^2)
\end{align*}
\]  

In machine learning (of which GP regression is a type), the main goal is to learn a representation of the unknown function $f$, given the available training data, in order to make prediction for new (yet unobserved) input points $x_*$ [43] [45] [46]. Assumptions must be made about the functional form of $f$, and in GP regression it is taken as a stochastic vector sampled from a multivariate
Gaussian distribution \[42\] :

\[ f \sim \mathcal{GP}(0, k(X, X)) \] (18)

The covariance matrix \( k(X, X) \) depends on the input training data, and is computed using an arbitrary covariance kernel function. The kernel function, \( k \), is a design choice that reflects prior knowledge one may have about the nature of \( f \) (smoothness, linearity, etc.) In BO, it is often hard to have any intuition about how the objective function behaves (especially in black-box optimization). A popular choice, that we will adopt in the following, is the \( 5/2 \) Matérn kernel, described in Equation \[19\]. This kernel is well suited for handling both smooth and non-smooth functions \[41\].

\[
k(x_i, x_j; \theta) = \frac{\theta_1}{\Gamma(\nu)2^{\nu-1}} \left( \frac{\sqrt{2\nu}||x_i - x_j||}{\theta_2} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu}||x_i - x_j||}{\theta_2} \right) \] (19)

where \( \nu = 5/2 \), \( \Gamma \) is the standard gamma function, \( K_\nu \) is the modified Bessel function of the 2\(^{nd} \) kind, and \( \theta = (\theta_1, \theta_2) \) is a set of tunable hyperparameters. These hyperparameters can be learned directly from the data, by maximizing the marginal likelihood:

\[
p(y|X, \theta) = \int p(y|f)p(f|X, \theta)df \] (20)

The likelihood \( p(y|f) \) is an immediate consequence of Equation \[17\] and \( p(f|X, \theta) \) is the prior distribution defined in Equation \[18\]. In the present case, the marginal likelihood is analytically tractable and exact Bayesian inference can be performed \[42\]. In order to make predictions at any location, a new input point \( x_\ast \) is appended to the training data \( X \), and the noise variance is incorporated within the upper left block of the covariance matrix. Equation \[18\] therefore becomes:

\[
\begin{bmatrix} y \\ f_\ast \end{bmatrix} \sim \mathcal{N} \left( 0, \begin{bmatrix} k(X, X) + \sigma^2 I_n & k(X, x_\ast) \\ k(x_\ast, X) & k(x_\ast, x_\ast) \end{bmatrix} \right) \] (21)

The joint distribution above can then be combined with the marginal likelihood to find the predictive distribution, conditioned on the training data:

\[
p(f_\ast|X, y, x_\ast) = \frac{p(y, f_\ast|X, x_\ast)}{p(y|X)} = \mathcal{N}(f_\ast|\mu_\ast(x_\ast), \Sigma_\ast(x_\ast)) \] (22)
In the present case, the distribution over the predictive output, $f_*$, is Gaussian, and analytical expressions can be found for its mean value $\mu_*$ and variance $\Sigma_*$. More details on GPs and practical implementation considerations can be found in the reference book from Rasmussen & Williams, *Gaussian Processes for Machine Learning* [42].

### 2.4.2. Acquisition Functions

Gaussian processes are a powerful tool for building a surrogate of the objective function that we want to minimize. As a Bayesian method, it typically avoids the caveat of overfitting [47], and the predictive distribution offers insight on how much is truly known about the objective function. To find out where the objective function should be evaluated next, acquisition functions are used. The acquisition function embodies a principled means to quantify, for any $x \in \Omega$, how likely it is that evaluating $f(x)$ will yield improvement towards the global minimum of our objective function. Many types of acquisition functions can be found in the literature, but in the current work we adopt Expected Improvement (EI): a popular method that is both analytically tractable and can easily handle constrained BO [41] [17].

A Bayesian optimization iteration loop must be initialized using a few evaluations of the objective function (typically at random input locations). Among all of the evaluations, the (currently known) minimum is denoted as $\tilde{f}$. For any $x \in \Omega$, the potential improvement toward the global minimum, given $\tilde{f}$ is:

$$ I(x) = \max(0, \tilde{f} - f(x)) $$

where $f$ is unknown, but approximated with a GP, so the expected improvement can be derived and expressed in term of the GP predictive mean and
variance \[41\].

\[
EI(x) = E_{p(f_*|X,y,x)}[I(x)] \\
= \int_{-\infty}^{+\infty} \max(0, \tilde{f} - f_*)N(f_*|\mu_*(x), \Sigma_*(x))df_* \\
= (\tilde{f} - \mu_*(x))\Phi(\tilde{f}|\mu_*(x), \Sigma_*(x)) + \Sigma_*(x)N(\tilde{f}|\mu_*(x), \Sigma_*(x))
\]

(24)

\(\Phi\) refers to the univariate normal cumulative distribution function (CDF). The expected improvement can now be maximized using standard optimization methods (e.g. BFGS, gradient descent, etc.) \[48\], and the objective function can be evaluated at the new input \(x_{next} = \arg \max EI(x)\). Finally, the evaluated value \(f(x_{next})\) can be used to update the GP predictions, and if a better global minimum candidate value is obtained, the value of \(\tilde{f}\), as well. This process can be iterated until convergence (as gauged by a user specified tolerance) or until the budget for evaluating the objective function has been exhausted. Note that each complete BO iteration comprises two optimization sub-iterations: The first one for maximizing the marginal likelihood when fitting the GP to the data, and the second one for maximizing the acquisition function. As a result, each BO iteration is fairly computationally intensive (but also worth it, when the cost of evaluating the objective function is itself significantly more expensive).

In order to deal with constrained optimization context, expressed in Equation \[16\] the standard expected improvement acquisition function in Equation 24 must be modified \[17\]. To account for the fact that some inputs \(x \in \Omega\) are simply not feasible with respect to constraint \(c_k(x) \leq \lambda_k\), the modification results in:

\[
cI(x) = \Delta(x)I(x)
\]

(25)

\[
\Delta(x) = \begin{cases} 
0 & \text{if } c_k(x) \geq \lambda_k \\
1 & \text{if } c_k(x) \leq \lambda_k 
\end{cases}
\]

(26)
Since $c_k$ may be a black box function, evaluated at the same time as the objective function, it may be approximated with a GP as well. As a result, $\Delta(x)$ is a random variable following a Bernoulli distribution of parameter $\rho(x)$ \[17\]. Furthermore:

$$\rho(x) = p(c_k(x) \leq \lambda_k) = \int_{-\infty}^{\lambda_k} p(c_k,|X,y,x)dc_k$$

(27)

Where $p(c_k,|X,y,x)$ is the Gaussian predictive distribution of $c_k$, in Equation 22 and conveniently $\rho$ is a univariate Gaussian cumulative distribution function \[17\]. Finally, the expected constrained improvement becomes:

$$c\text{EI}(x) = E[\Delta(x)I(x)]$$

$$= E_{\mathcal{B}(1,\rho(x))}[\rho(x)]E_{p(f,|X,y,x)}[I(x)]$$

(28)

2.4.3. Bayesian Optimization Implementation

To implement our Bayesian optimization codes, we use \textbf{botorch} \[49\], a dedicated python library. \textbf{botorch} relies on \textbf{gpytorch} \[50\] for performing all the relevant GP inferences, and both of these libraries are built on top of the deep learning framework \textbf{pytorch} \[51\].

3. FSI Validation

As part of our design approach, we carefully searched the literature for a canonical FSI verification problem that was similar enough to our ultimate design application \textit{i.e.} fiber reinforced plastic UUAV sail plane, to be useful is building or confidence in the veracity of our FSI modeling approach. To that end, the Turek & Hron’s benchmark cases \[18\] are considered here as we judge the accuracy of our implicit partitioned FSI approach using CU-BENs and OpenFOAM solvers. The general problem setup for our verification context is shown in Figure 5. Within the structural domain, an elastic cantilever beam is attached to a fixed cylinder whose radius is 5 cm; the center of the cylinder is positioned at
(0.2 m, 0.2 m). The cantilever beam is 0.35 m long, 0.02 m thick, and 0.01 m wide.

No-slip boundary conditions are imposed at the top and bottom walls of the channel, as well as the interface along the cylinder and the cantilever. The pressure outflow condition along the right edge of the channel is prescribed to atmospheric pressure. The channel inflow condition is defined along the left edge of the domain as a parabolic velocity profile, described in Equation 29, where \( U \) is the mean inflow velocity. Equation 30 describes the evolution of the inflow velocity as a function of time.

\[
u_f(0, y) = 1.5U \frac{4}{0.1681} y(0.41 - y)
\]

\[
u_f(t, 0, y) = \begin{cases} 
\frac{u_f(0, y)[1 - \cos\left(\frac{\pi t}{2}\right)]}{2} & \text{if } t < 2.0 \\
 u_f(0, y) & \text{otherwise}
\end{cases}
\]

The verification procedure sets out to verify three test cases (i.e. the CSM3 test, the CFD3 test, and the FSI3 test) detailed in [18], each test case evaluates the behavior of the corresponding component response within our CU-BENs/OpenFOAM implicit FSI solver, so that comparisons can be made with the ground truth data reported in [18]. Mesh refinement studies are undertaken at this point, so that we may know what spatiotemporal discretizations are suitable as we move towards our ultimate design case.
3.1. CSM3 Test

An elastic cantilever beam subjected to gravitational force, only, with \( g = 2 \text{ m/s}^2 \) is carried out in the CSM3 test. The elastic beam has a Young’s modulus of 1.4 MPa, Poisson ratio of 0.4, and mass density of 1000 kg/m\(^3\). Implicit Newmark time integration scheme, coupled with Newton Raphson sub-iteration within each time step, is used to capture the geometric nonlinearity effects within the CU-BENs structural model. The displacements at the free end of the cantilever are recorded. Figure 6 and Table 1 presents the x– and y– tip displacements over the time span \( t = [8, 10] \) seconds. As shown, that the tip displacements tend to the reference values provided by Turek & Hron, as the number of shell elements increases; negligible differences among the displacements are observed.

![Figure 6: x– and y– displacements at the free end of the cantilever at time \( t = [8, 10] \) seconds: \( \delta t = 0.005 \text{ sec} \)](image)

| Number of shells | x– displacement [centerline ± amplitude] | y– displacement [centerline ± amplitude] | Frequency [x– disp] | Frequency [y– disp] |
|------------------|------------------------------------------|------------------------------------------|---------------------|---------------------|
| 20               | -15.462 ± 15.509                         | -67.940 ± 66.600                         | 1.087               | 1.0989              |
| 40               | -15.062 ± 14.908                         | -66.441 ± 65.500                         | 1.0989              | 1.0989              |
| 80               | -14.552 ± 14.547                         | -65.152 ± 64.848                         | 1.0989              | 1.0989              |

Table 1: Peak x– and y– displacement at the free end of the cantilever at time \( t = [8, 10] \) seconds: \( \delta t = 0.005 \text{ sec} \)
3.2. CFD3 Test

Next, the fluid dynamic aspects of the coupled system are verified using the CFD3 test case, within which the cantilever is modeled as a rigid beam. A parabolic inflow described in Equation 30 with 2 m/s mean inflow velocity, is prescribed along the left boundary edge of the channel. The fluid flow is simulated with OpenFOAM’s incompressible solver: PimpleDyMFoam. The total drag and lift forces along the cylinder, as well as along the rigid beam, are recorded. The drag and lift over the time span \( t = [9, 10] \) seconds can be found in Figure 7 and Table 2. We see that the drag and lift forces computed using PimpleDyMFoam are converging to Turek & Hron’s values with increasing number of control volumes. At the finest level of refinement, with 606,208 control volumes, the percentage differences for the peak drag and lift values are well below 2 %.

Figure 7: Drag and lift forces along the cylinder and the cantilever at time \( t = [9, 10] \) seconds: \( \delta t = 0.005 \) sec

| Number of control volumes | Drag force [centerline ± amplitude] | Frequency [drag] | Lift force [centerline ± amplitude] | Frequency [lift] |
|---------------------------|------------------------------------|------------------|-------------------------------------|------------------|
| 37,888                    | 470.29 ± 6.915                     | 3.42             | -1.882 ± 574.86                     | 1.9216           |
| 151,552                   | 451.98 ± 8.285                     | 4                | -22.998 ± 496.76                    | 4                |
| 606,208                   | 431.90 ± 4.3290                    | 4                | -42.279 ± 461.54                    | 4                |
| Turek & Hron reference    | 439.45 ± 5.6183                    | 4.3956           | -11.893 ± 437.81                    | 4.3956           |

Table 2: Peak drag and lift forces along the cylinder and the cantilever at time \( t = [9, 10] \) seconds: \( \delta t = 0.005 \) sec
3.3. FSI3 Test

The complete FSI modeling context is treated in FSI3 test case, where an elastic structure (Young’s modulus of 5.6 MPa, Poisson ratio of 0.4, and mass density of 1000 kg/m$^3$) is submerged in a fluid channel that is 2.5 m long, 0.41 m tall, and 0.01 m wide; a summary of the material properties for the fluid and structural domains is provided in Table 3. The channel inlet, on the left, is prescribed with a parabolic inflow speed with mean inflow velocity of 2 m/s. The fluid flow is modeled with OpenFOAM’s PimpleDyMFoam solver, while, the structural deformation is estimated using the Generalized-α integration method within CU-BENs (a spectral radius of 0.8 is applied to ensure numerical stability.) Deformation along the fluid and structural interface is governed by the IQN-ILS scheme. Displacements along the moving boundary are iterated until the residual vector ($R^k$) satisfies the convergence criterion $\|R^k\| \leq \epsilon$, where the convergence tolerance is $\epsilon = 10^{-5}$.

| Fluid Properties | Structural Properties |
|------------------|-----------------------|
| Density, $\rho_f$ | $10^3$ kg/m$^3$       |
| Viscosity, $\mu_f$ | $10^{-3}$ m$^2$/s    |
| Reynolds number, Re | 200                  |
| Density, $\rho_s$ | $10^3$ kg/m$^3$       |
| Poisson ratio, $\mu_s$ | 0.4                |
| Young’s modulus, E | 5.6 MPa               |

The full analysis time (not wall clock) is 20 sec with $\delta t = 0.0005$ sec. The x– and y– tip displacements over the time span $t = [19.5, 20]$ seconds as well as the peak x– and y– tip displacements are provided in Figure 8 and Table 4. The analysis is repeated using three levels of mesh refinement in the fluid domain, meanwhile the discretization in the structural domain is fixed at 80 DKT shell elements. As shown, the implicit FSI solver is able to predict the structural deformation reasonably well compared to the reference values provided by Turek & Hron. The peak displacements are slightly higher than the benchmark values.
with the peak x– displacement converging towards \(-2.89 \pm 2.75\) mm and the peak y– displacement converging towards \(1.36 \pm 35.97\) mm. Nonetheless, the discrepancies among the peak displacements are within 1 mm in x– direction and 2 mm in y– direction, approximately 12.5% difference for x– displacement and 4.1% difference for y– displacement.

![Graph showing x– and y– displacements at the free end of the cantilever during time \(t = [19.5, 20]\) seconds: \(\delta t = 0.0005\) sec](image)

**Figure 8:** x– and y– displacements at the free end of the cantilever during \(t = [19.5, 20]\) seconds: \(\delta t = 0.0005\) sec

| Number of shells | Number of control volumes | x– displacement (centerline ± amplitude) | Frequency [x– disp] | y– displacement (centerline ± amplitude) | Frequency [y– disp] |
|------------------|---------------------------|-----------------------------------------|---------------------|-----------------------------------------|---------------------|
| 80               | 9,472                     | -2.46 ± 2.35                            | 11.11               | 1.61 ± 32.75                            | 15.8                |
| 190              | 26,160                    | -2.87 ± 2.73                            | 11.11               | 1.46 ± 36.0                            | 15.6                |
| 280              | 37,888                    | -2.89 ± 2.75                            | 11.11               | 1.36 ± 35.97                            | 15.6                |
| Turek & Hron       |                           | -2.69 ± 2.55                            | 13.9                | 1.48 ± 34.38                            | 15.1                |

**Table 4:** Peak x– and y– displacements at the free end of the cantilever during time \(t = [19.5, 20]\) seconds: \(\delta t = 0.0005\) sec

The drag and lift responses over the time span \(t = [19.5, 20]\) seconds are presented in Figure 9 and Table 5. Minor nonphysical oscillations can be observed due to the different spatial discretization in the fluid and structural systems as well as nonconforming mesh along the fluid-structure interface. The implicit FSI solver estimates the peak drag converge towards \(489.2 \pm 30.44\) N, and the peak lift converge towards \(11.084 \pm 181.21\) N. The discrepancies among the peak drag and lift are within 40 N and 170 N respectively, with percentage difference of 8% for drag and 26.5% for lift. The discrepancies are due to the difference in fluid solver used in the present work (i.e. different spatial temporal discretization scheme, different approach to solving the pressure-velocity equations, etc.)
as well as the different FSI approach compared to Turek & Hron. As specifically noted in [52], with different discretization, solver, and coupling mechanism, differences in different approach could lead to discrepancies of up to 50% for the drag and lift values and 10% difference for the displacement values. With this in mind, we are confident in our FSI predictions, as we are well below these thresholds.

Figure 9: Drag and lift forces along the cantilever during time $t = [19.5, 20]$ seconds: $\delta t = 0.0005$ sec

| Number of shells | Number of control volumes | Drag force [centerline ± amplitude] | Frequency [drag] | Lift force [centerline ± amplitude] | Frequency [lift] |
|------------------|---------------------------|------------------------------------|-----------------|-----------------------------------|-----------------|
| 80               | 20,160                    | 488.3 ± 31.24                      | 11.11           | 8.041 ± 180.31                    | 5.56            |
|                  | 37,888                    | 488.2 ± 31.44                      | 11.11           | 11.064 ± 181.21                   | 5.56            |
| Turek & Hron reference | 437.3 ± 22.66            | 10.9                               | 2.22 ± 149.76   | 5.3                              |

Table 5: Peak drag and lift forces during time $t = [19.5, 20]$ seconds: $\delta t = 0.0005$ sec

4. Bridging Simulations

In the previous section, we identified canonical verification models to furnish ground truth results for assessing the performance of our proposed FSI simulation framework. These verification analyses allowed us to identify suitable spatiotemporal discretization for our application space, along with gauging the efficacy of our selected numerical schemes. As it is that we need to extend our analyses to be more realistic with respect to our ultimate design application (fiber reinforced polymer UUAV sail plane), we now incrementally increase our
modeling complexity using what refer to as *bridging simulations*. We perform Bayesian optimization (both constrained and unconstrained) using our bridging simulations. The geometric configuration and boundary conditions of the FSI3 benchmark, outlined in Section 3 are leveraged in the bridging simulations, as detailed in Section 4.1 to 4.3. As it is that our ultimate design application involves the material property design for a composite fiber reinforced structural component of fixed geometry, in our bridging simulations the elastic modulus of our structure is specified as a function of its cantilevered geometry within our bridging simulations: as linear, uniform, and box functions, respectively.

A schematic representation of the BO-FSI procedure is presented in Figure 10 where \( x \) is the optimization parameter (also referred to as input), \( f(x) \) is the objective function, \( c_k(x) \) is the set of constraint functions, and \( cEI(x) \) is the acquisition function. The BO process goes as follows:

1. We initialize the optimizer by defining a set of initialization inputs (chosen randomly except in section 4.3 and 5).
2. We pass the initialization inputs to the FSI solver, which outputs corresponding data on the objective function and the constraints.
3. The BO solver computes the GP surrogates, maximizes the acquisition function, and provide a new input point.
4. The new input point is passed to the FSI solver, and the process it repeated until the desired design is uncovered.

Figure 10: A schematic representation of the interplay between the FSI and Bayesian optimization solvers

24
Note that in Bayesian optimization, there is no obvious, *a priori* stopping criterion; though there may be a hard limit on the number of feasible iterations (*e.g.* available computational budget for evaluating the objective function): knowing when to stop may be difficult. There is a trade-off to balance between objective function evaluation cost and the likelihood of finding a better (the global) minimum. The optimizer is typically stopped when no significant progress has been made for the last few iterations, or if the uncovered design is simply “good enough” for engineering purposes.

4.1. Example 1: Minimizing Tip Displacement with Outlet Velocity constraint

In this first example, we aim to minimize the vertical tip displacement of the cantilever beam, used in the FSI3 verification, subject to an outlet velocity constraint. In this section, and those subsequent, the tip displacement corresponds to the maximum displacement attained at any time point within the stabilized FSI run. The beam elasticity modulus, $E$, is assumed to vary linearly along the length of the cantilever (with zero coordinate corresponding to the fixed end, at the cylinder):

$$
\begin{align*}
E(x) &= Ax + B & x &\in [0, l] \\
\int_{0}^{l} E(x) dx &= 1.96 \text{ MPa}
\end{align*}
$$

The integral of the elastic modulus function, taken along the beam, remains constant for all time; this results in $B$ being dependant on $A$. In addition, to ensure $E(x) > 0$, $A$ is bounded in the interval $[-31.9088, 31.9088]$. Our goal is to determine the value of $A$, such that the vertical beam tip displacement, $\delta$, is minimized, while also making sure that we do not exceed our prescribed maximum outlet fluid velocity, $v_c = 1.4 \text{ m/s}$, where the velocity is taken at mid-height. This example scenario typically relates to situations where making the beam stiffer is costly, and thus choices must be made on where to concentrate the stiffness throughout the beam in an optimal manner. Formally, the optimization
The problem is:

\[
\text{Find } \min_A \delta \quad \text{s.t. } v_x \leq v_c \tag{32}
\]

The Bayesian optimizer is initialized using four different FSI runs; instantiated with random values for \(A\). Table 6 shows the successive input values proposed by the optimizer, as well as the corresponding tip displacement and outlet velocity. Figure 11 shows plots of the surrogate GPs for \(\delta(A)\) and \(v_x(A)\), along with the constrained expected improvement at the initialization, and after each BO iteration. Ultimately, the tip displacement, and the outlet velocity, both appear to have a monotonic trend; thus the optimizer pushes us towards the lower bound of \(A\) (i.e. where the stiffness is concentrated at the beam’s fixed end).

Once \((A_{\min} = -31.9088, \delta_{\min} = 0.0262)\) has been reached, no progress towards a better global minimum is being made, and given the likely monotonic trend of the objective function, future progress is deemed unlikely, and so the optimizer is stopped (Figure 12).

| Iteration # | \(A\) (MPa) | \(\delta\) (m) | \(v_x\) (m/s) |
|-------------|-------------|--------------|-------------|
| Init        | -2.565      | 0.0347       | 1.3035      |
| Init        | -4.957      | 0.035        | 1.2794      |
| Init        | 29.033      | 0.048        | 1.5844      |
| Init        | 14.764      | 0.0410       | 1.4338      |
| 1           | -10.5329    | 0.0335       | 1.2258      |
| 2           | -16.7519    | 0.0319       | 1.171       |
| 3           | -27.0711    | 0.0299       | 1.1016      |
| 4           | \(-31.9088\) | \(0.0262\)  | \(1.0337\) |
| 5           | -31.565     | 0.0271       | 1.0651      |

Table 6: Successive BO iterations - tip displacement with outlet velocity constraint
Figure 11: Successive BO iterations - tip displacement with outlet velocity constraint. The solid blue and orange lines represent the predictive mean of the two surrogate GPs ($\delta$ and $v_x$ respectively) with their 95% confidence interval (shaded areas), given the already known tip displacements and outlet velocity values. The acquisition function indicates what value of $A$ is the most likely to yield a better global minimum (i.e. the value that should be use in the next FSI run).

Figure 12: Best minimum recorded at each BO iteration - tip displacement with outlet velocity constraint
4.2. Example 2: Unconstrained Uniform Stiffness

In this second example, we now aim to minimize the vertical tip displacement of the cantilever with respect to $B$ in Equation 31, assuming that the beam has an uniform stiffness ($A = 0$) (i.e. integral condition on $E(x)$ is no longer enforced). $B$ is bounded in the interval $[0, 10]$, but no additional constraints are used. This leads to an unconstrained BO, with the standard expected improvement acquisition function:

$$\text{Find } \min_B \delta$$

This optimization problem may seem overly simple, because one might intuitively expect that larger stiffness values always yield a smaller tip displacement. However, the initialization data shows the opposite, as $B = 2$ leads to a smaller $\delta$ than $B > 3.1401$. To justify the results, velocity profiles of the test case are provided in Figure 13 where the beam deformation for constant elasticity functions, with $B = 5.6$ and $B = 2$, are compared. As shown in Figure 13, under the same conditions, the cantilever with lower stiffness exhibits a higher mode of oscillation, under the action of the complex vortex shedding within the von Kármán vortex street. As a result of this more complex motion, the less stiff beam exhibits a smaller tip deflection than the stiffer case. This highlights some interesting, and counter intuitive, results that may arise within FSI contexts.

Figure 14a shows the GP surrogate after 11 BO iterations and indicates that $\delta(B)$ follows a highly non-monotonic and non-convex behavior, with a significant drop in tip displacement for $B \approx 2$. This drop is sharp, which makes the search for the global minimum difficult, but after 6 iterations, $(B_{\text{min}} = 1.9907, \delta_{\text{min}} = 0.0217)$ is found. This value is taken as the global minimum, since performing 5 additional iterations did not allow for further improvements (Figure 14b).
Figure 13: Velocity profiles for simulations with beam stiffness $B = 5.6$ and $B = 2$ at time $t = 6.25 \text{ sec}$ and $12.5 \text{ sec}$, respectively.

Figure 14: Successive BO iterations - unconstrained uniform stiffness.
4.3. Example 3: Unconstrained Non-Uniform Stiffness

In this third example, we consider the stiffness function along the beam to be described by a box-function as illustrated in Figure 15. Formally, the stiffness is defined by:

$$E(x) = \begin{cases} 
7.0 & \text{if } x \in [x_b - l/6, x_b + l/6] \\
5.6 & \text{else}
\end{cases}$$ (34)

Again, we are looking to minimize the vertical tip displacement, this time with respect to the box mid-point, $x_b$. Figure 16a shows the surrogate GP after 11 iterations. The tip displacement tends to decrease, overall, when the zone of concentrated material stiffness (i.e the box) is positioned close to the beam’s fixed end. Interestingly, the minimum tip deflection value, or $\delta_{\text{min}} = 0.0401$, is

| Iteration # | $B$ (MPa) | $\delta$ (m) |
|-------------|-----------|-------------|
| Init        | 8.0408    | 0.0379      |
| Init        | 3.1401    | 0.0418      |
| Init        | 5.6781    | 0.0422      |
| Init        | 2.0000    | 0.0220      |
| 1           | 1.5689    | 0.0385      |
| 2           | 2.2160    | 0.0371      |
| 3           | 1.9078    | 0.0244      |
| 4           | 1.9774    | 0.0221      |
| 5           | 2.0303    | 0.0225      |
| 6           | 1.9907    | 0.0217      |
| 7           | 1.8059    | 0.0288      |
| 8           | 9.9826    | 0.0336      |
| 9           | 4.3898    | 0.0429      |
| 10          | 2.0727    | 0.0218      |
| 11          | 2.0599    | 0.0220      |

Table 7: Successive BO iterations - unconstrained uniform stiffness
not obtained for a box fully shifted to the far left (i.e. corresponding to the fixed end), but rather for a box located at about a third of the beam length \((x_b,\text{min} = 0.1199)\). After 11 iterations, the next \(x_b\) value proposed by the optimizer is \(x_b = 0.1199\), which has already been tried and yields the current best minimum. Therefore, it is unlikely to improve, and so the optimization is stopped there.

\(\begin{align*}
E & \quad l/3 \\
x_b & \quad 7 \text{ MPa} \\
l & \quad 5.6 \text{ MPa}
\end{align*}\)

Figure 15: Stiffness along the beam given the box mid-point, \(x_b\)

\(\begin{align*}
\text{(a) Tip displacement with box function stiffness (11}\text{th iteration)} & \quad \text{(b) Best minimum recorded at each BO iteration}
\end{align*}\)

Figure 16: Successive BO iterations - non-uniform stiffness
Table 8: Successive BO iterations - non-uniform stiffness

| Iteration # | $x_b$ (m) | $\delta$ (m) |
|-------------|-----------|--------------|
| Init        | 0.0585    | 0.0403       |
| Init        | 0.1755    | 0.0413       |
| Init        | 0.2925    | 0.0416       |
| 1           | 0.0856    | 0.0405       |
| 2           | 0.0675    | 0.0412       |
| 3           | 0.1238    | 0.0402       |
| 4           | 0.2368    | 0.0417       |
| 5           | 0.2062    | 0.0407       |
| 6           | 0.1159    | 0.0402       |
| 7           | 0.1008    | 0.0406       |
| 8           | 0.2675    | 0.0414       |
| 9           | 0.1453    | 0.0410       |
| 10          | 0.1199    | 0.0401       |
| 11          | 0.1928    | 0.0407       |

5. Composite Material Properties Selection for UUAV Sail Plane

Fortified with our experience with verification and bridging simulations, we now turn our attention towards a realistic application: a UUAV sail plane made from FRP (fiber reinforced polymer), as shown in Figure 17. The sail plane geometry is fixed: 0.4 m long and 0.01 m wide, with some adjustable angle of attack, $\theta$ (with respect to the horizontal). The leading edge of the sail plane is 0.038 m thick; reducing linearly with length, to assume a value of 0.004 m thickness at the trailing edge. The sail plane has a density of 2000 kg/m$^3$. A uniform inflow velocity of 5 m/s is prescribed along the left boundary. We wish to find the optimal (uniform) stiffness, $E$, and angle of attack $\theta$, that minimizes the drag over time, while imposing conditions on the lift, the vertical tip displacement, as well as the outlet pressure differential. Unlike previous cases, this is a 2-
parameters optimization problem:

\[
\begin{align*}
\text{Find} \quad & \min_{\theta, E} F_D \\
\text{s.t.} \quad & F_L \geq F_{L,c} \\
& \delta \leq \delta_c \\
& \Delta p \leq \Delta p_c
\end{align*}
\] (35)

The drag \( F_D \) and lift \( F_L \) values are respectively taken as the sum of the total pressure acting on the sail plane in the tangential and normal direction, with respect to the orientation of the sail plane, over the time span \([0.5 \text{ sec}, 3.5 \text{ sec}]\). \( \delta \) is the plane vertical tip displacement. The pressure differential, \( \Delta p \), is taken as \( \Delta p = p_{\text{max}} - p_{\text{min}} \), where \( p_{\text{max}} \) and \( p_{\text{min}} \) are the pressures recorded at the outlet, at a height of \( 5/8H \), during the time interval \([0.5 \text{ sec}, 3.5 \text{ sec}]\). The constraints are \( F_{L,c} = 12000 \text{ N}, \delta_c = 2.5 \text{ mm}, \) and \( \Delta p_c = 120 \text{ kPa} \). The Bayesian Optimizer is initialized with eight data points (since the input space is now in two dimensions, more data are required for fitting the surrogate GPs properly).

Table 9 shows the successive input parameters proposed by the optimizer, along with the corresponding FSI analysis results. Figure 19 shows the predictive mean of the drag force predicted by the surrogate GP at each BO iteration, with the feasible/unfeasible inputs regions (given the constraint set for the surrogate GPs). Figure 18 shows the best minimum found after each iteration. The drag appears to follow a very smooth and fairly monotonic behavior, and the optimizer converges quickly toward a minimum. As seen in Figure 19, the feasible regions provide significant freedom for \( E \) to vary, whereas the feasible regions for \( \theta \) are much more narrow. This is intuitive since one might expect the angle of the sail plane to influence the drag more than the stiffness. Yet,
the optimal stiffness value is not trivial. Given the optimizer progress after 8 iterations (Figure 18), it seems unlikely that a significantly better minimum drag value can be achieved, and the iteration loop is stop there. The global minimum is taken as $(\theta_{\text{min}} = 3.74, E_{\text{min}} = 35.39, F_{D,\text{min}} = 2303)$.

| Iteration # | $\theta$ ($^\circ$) | $E$ (MPa) | $F_D$ (N) | $F_L$ (N) | $\delta$ (mm) | $\Delta p$ (kPa) |
|-------------|----------------------|------------|-----------|-----------|---------------|-----------------|
| Init        | 0.00                 | 40.00      | 350       | 3540      | 0.565         | 9.50            |
| Init        | 2.50                 | 40.00      | 1482      | 9214      | 1.206         | 39.90           |
| Init        | 5.00                 | 40.00      | 3419      | 13736     | 2.146         | 89.70           |
| Init        | 7.50                 | 40.00      | 5042      | 16183     | 2.593         | 141.90          |
| Init        | 10.00                | 40.00      | 7148      | 18333     | 3.399         | 165.60          |
| Init        | 2.50                 | 30.00      | 1452      | 10051     | 1.813         | 42.40           |
| Init        | 7.50                 | 50.00      | 4767      | 16336     | 1.884         | 139.60          |
| Init        | 10.00                | 30.00      | 7026      | 17716     | 4.840         | 173.80          |
| 1           | 3.85                 | 39.93      | 2454      | 12025     | 1.739         | 74.53           |
| 2           | 3.66                 | 30.00      | 2256      | 11891     | 2.355         | 61.83           |
| 3           | 3.75                 | 30.00      | 2247      | 11320     | 2.183         | 71.42           |
| 4           | 4.02                 | 35.77      | 2466      | 12467     | 1.749         | 89.06           |
| **5**       | **3.74**             | **35.39**  | **2303**  | **12184** | **1.956**     | **50.37**       |
| 6           | 3.63                 | 35.12      | 2172      | 11961     | 1.939         | 77.97           |
| 7           | 3.65                 | 35.10      | 2285      | 11858     | 1.941         | 76.85           |
| 8           | 3.69                 | 35.11      | 2164      | 11547     | 1.934         | 78.80           |

Table 9: Successive BO iterations - UUAV sail plane
Figure 18: Best minimum recorded at each BO iteration - UUAV sail plane

Figure 19: Successive BO iterations - UUAV sail plane. The color surface plots represent the GP predictive mean for the drag, and the shaded dark areas represent the unfeasible input regions imposed on the surrogate GPs for the lift, tip displacement, and pressure.
6. Conclusion

This paper has demonstrated a principled approach to design using high fidelity FSI simulations. Our proposed approach consists of three stages. Firstly, verify and/or validate the FSI computational tool using canonical cases and/or experiment results from the literature with design configurations that are close to the ultimate application. This helps identify the optimal spatiotemporal discretization, as well as numerical schemes, for the design application. Subsequently, incrementally and systematically increase the complexity of the verification case to bring it closer to the design context, using what we term bridge simulations. This step provides information to gauge whether adjustments need to be made to further improve the computational model. Finally, apply the verified FSI framework, thus now trusted, to the design problem of interest.

Our experience with BO has shown it to be able to handle complicated design scenarios that have multiple design parameters, and multiple constraints. Furthermore, it is able to effectively optimize design cases that exhibit both smooth and non-smooth objective functions, even with counter-intuitive optimal parameters directions.

7. Acknowledgement

The support of the Office of Naval Research (ONR), under the grant N00014-19-1-2034 and contract N6833518C0217, is gratefully acknowledged.

References

[1] Sarah C. Vignostad, Holavanahalli S. Udaykumar, Jia Lu, and Krishnan B. Chandran. Fluid–structure interaction methods in biological flows with special emphasis on heart valve dynamics. *International Journal for Numerical Methods in Biomedical Engineering*, 26(34):435–470, 2010.
[2] Ramji Kamakoti and Wei Shyy. Fluid–structure interaction for aeroelastic applications. *Progress in Aerospace Sciences*, 40(8):535 – 558, 2004.

[3] Zhengyu Huang, Philip Avery, Charbel Farhat, Jason Rabinovitch, Armen Derkevorkian, and Lee D. Peterson. *Simulation of Parachute Inflation Dynamics Using an Eulerian Computational Framework for Fluid-Structure Interfaces Evolving in High-Speed Turbulent Flows*.

[4] J T Tuitman and Š Malenica. Fully coupled seakeeping, slamming, and whipping calculations. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 223(3):439–456, 2009.

[5] Hermann G. Matthies and Jan Steindorf. Partitioned strong coupling algorithms for fluid–structure interaction. *Computers and Structures*, 81(8):805 – 812, 2003. K.J Bathe 60th Anniversary Issue.

[6] Hermann G. Matthies, Rainer Niekamp, and Jan Steindorf. Algorithms for strong coupling procedures. *Computer Methods in Applied Mechanics and Engineering*, 195(17):2028 – 2049, 2006. Fluid-Structure Interaction.

[7] Joris Degroote, Robby Haelterman, Sebastiaan Annerel, Peter Brugeman, and Jan Vierendeels. Performance of partitioned procedures in fluid–structure interaction. *Computers and Structures*, 88(7):446 – 457, 2010.

[8] P. Causin, J.F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194(42):4506 – 4527, 2005.

[9] Christiane Förster, Wolfgang A. Wall, and Ekkehard Ramm. The artificial added mass effect in sequential staggered fluid–structure interaction algorithms. TU Delft, The Netherlands, 2006. European Conference on Computational Fluid Dynamics.

[10] Christiane Förster, Wolfgang A. Wall, and Ekkehard Ramm. Artificial added mass instabilities in sequential staggered coupling of nonlinear struc-
tures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 196(7):1278 – 1293, 2007.

[11] Yichi Zhang, Daniel Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables, 2019.

[12] Peter I. Frazier and Jialei Wang. Bayesian optimization for materials design. *Springer Series in Materials Science*, pages 45–75, Dec 2015.

[13] Remi Lam, Matthias Poloczek, Peter Frazier, and Karen Willcox. Advances in bayesian optimization with applications in aerospace engineering. 01 2018.

[14] Doniyor Ulmasov, Caroline Baroukh, Benoit Chachuat, Marc Peter Deisendorf, and Ruth Misener. Bayesian optimization with dimension scheduling: Application to biological systems, 2015.

[15] E. O. Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2:1–2:7, 2018.

[16] V. Nguyen. Bayesian optimization for accelerating hyper-parameter tuning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 302–305, 2019.

[17] Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, pages II–937–II–945. JMLR.org, 2014.

[18] Stefan Turek and Jaroslav Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In Hans-Joachim Bungartz and Michael Schäfer, editors, *Fluid-Structure Interaction*, pages 371–385, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
[19] Wensi Wu, Justyna Kosianka, Heather Reed, Christopher Stull, and Christopher Earls. Cu-bens: A structural modeling finite element library. *SoftwareX*, 11:100485, 2020.

[20] Zeljko Tukovic Hrvoje Jasak, Aleksandar Jemcov, editor. *OpenFOAM: A C++ Library for Complex Physics Simulations*, Dubrovnik, Croatia, September 2007. IUC, International Workshop on Coupled Methods in Numerical Dynamics.

[21] H.G. Weller and G. Tabor. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620, 1998.

[22] Hrvoje Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows*. PhD thesis, Imperial College of Science, Technology and Medicine, 1996.

[23] Bernhard Gschaider, Hakan Nilsson, Henrik Rusche, Hrvoje Jasak, Martin Beaudoin, and Vanja Skuric. OpenFOAM-1.6-ext.

[24] Ronald Miller, Sung-Eun Kim, and Sung Won Lee. A computational framework for fluid–structure interaction on flexible propellers involving large deformation. Pasadena, California, September 2010. 28th Symposium on Naval Hydrodynamics.

[25] J. Chung and G.M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of applied mechanics*, 60(2):371–375, 1993.

[26] Hrvoje Jasak. *Dynamic Mesh Handling in OpenFOAM*.

[27] Christophe Kassiotis. Which strategy to move the mesh in the computational fluid dynamic code openfoam. 2008.

[28] JOHN BATINA. *Unsteady Euler airfoil solutions using unstructured dynamic meshes*.
[29] Frederic J. Blom. Considerations of the spring analogy. volume 32, pages 647–668. International Journal for Numerical Methods in Fluids, 2000.

[30] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. Computer Methods in Applied Mechanics and Engineering, 163(1):231 – 245, 1998.

[31] Y. Zhao, J. Tai, and F. Ahmed. Simulation of micro flows with moving boundaries using high-order upwind fv method on unstructured grids. Computational Mechanics, 28(1):66–75, 2002.

[32] Blair Perot and Ramesh Nallapati. A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows. Journal of Computational Physics, 184(1):192 – 214, 2003.

[33] Christoph Degand and Charbel Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. Computers and Structures, 80(3):305 – 316, 2002.

[34] Hrvoje Jasak and Zeljko Tuković. Automatic mesh motion for the unstructured finite volume method, 2004.

[35] C. Farhat, M. Lesoinne, and P. Le Tallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. Computer Methods in Applied Mechanics and Engineering, 157(1):95 – 114, 1998.

[36] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In Proceedings of the 1968 23rd ACM National Conference, ACM ’68, pages 517–524, New York, NY, USA, 1968. Association for Computing Machinery.

[37] Joris Degroote, Klaus-Jürgen Bathe, and Jan Vierendeels. Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure
interaction. *Computers & Structures*, 87(11):793 – 801, 2009. Fifth MIT Conference on Computational Fluid and Solid Mechanics.

[38] Gene H Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[39] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[40] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[41] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[42] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Mass. [u.a.], 2006.

[43] Christophe Bonneville, Maxwell Jenquin, Juan Lundono, Alex Kelly, Jeffrey Cipolla, and Christopher Earls. Gaussian processes for shock test emulation. *In Review*, 2020.

[44] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[45] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[46] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[47] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, CAN, 1995. AAINN02676.
[48] Jungtaek Kim and Seungjin Choi. On local optimizers of acquisition functions in bayesian optimization, 2019.

[49] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: Programmable Bayesian Optimization in PyTorch. arXiv e-prints, page arXiv:1910.06403, October 2019.

[50] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In Advances in Neural Information Processing Systems, 2018.

[51] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[52] S. Turek, J. Hron, M. Razzaq, H. Wobker, and M. Schäfer. Numerical benchmarking of fluid-structure interaction: A comparison of different discretization and solution approaches. In Hans-Joachim Bungartz, Miriam Mehl, and Michael Schäfer, editors, Fluid Structure Interaction II, pages 413–424, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.