Thesis Proposal

An efficient optimization framework for gray-box conservation law simulation

---

Han Chen
Ph.D Candidate

Massachusetts Institute of Technology
Department of Aeronautics & Astronautics

Aeronautics Computational Design Laboratory

Submitted on: Apr 21, 2015

Thesis Committee: Qiqi Wang, Karen Willcox, Youssef Marzouk

External evaluator: Hector M. Klie

Abstract

Gray-box simulations of conservation laws are present in many engineering applications. We call a conservation law simulation *gray-box* if it can output the space (or space-time) solution of the flow-variables. My thesis is interested in the optimization based on gray-box conservation law simulations, which is a common task in the engineering community. In many scenarios such simulations do not have adjoint implemented, so we can not compute efficiently the gradients of the optimization's objective function with respect to the design variables. So conventional optimization practices generally treat the gray-box simulation as only a calculator for the objective function. When the dimension of design space is high, such conventional practices may require large amount of simulations which can be computationally prohibitive.

My thesis proposes an efficient methodology for high-dimensional optimization problems by leveraging the limited insights and the space or space-time solution of the gray-box simulation. My method constructs a PDE surrogate, called twin model, to serve as a low-fidelity avatar of the gray-box simulation. The innovation of my research is: the twin model mimics the gray-box simulation's space or space-time solution, instead of mimicing just the objective function. The twin model first constructs a parameterized surrogate PDE, using the fact the gray-box simulation simulates a conservation law. Then it tunes the parameters to match the gray-box simulation's space or space-time solution. Generally the space or space-time solution contains much more information about the gray-box simulation than the objective function.

Adjoint can be implemented in the twin model to estimate the gradient of the objective function with respect to design variables efficiently. Optimization can be performed in a multi-fidelity framework using the low-fidelity twin model and the high-fidelity gray-box simulation. My thesis proposes a provably convergent Bayesian optimization scheme that combines the evaluations of the gray-box' objective function with the twin model's gradient. For completeness, we extend the optimization framework to constraint optimization.

We propose to demonstrate our method by a turbulent flow optimization in a turbine airfoil internal cooling problem. The gray-box simulation is performed by a large eddy simulation (LES). The twin model is a Reynolds-averaged Navier-Stokes (RANS) model with an adaptive Reynold stress. We optimize a return bend geometry to minimize the pressure loss, and compare our result with state-of-art optimal designs.

Contents

1 Background

1.1 Optimization based on gray-box conservation laws

Optimization problems is of great interest in the engineering community. We consider minimizaing an objective function:

$$J : \mathcal{C} \in \mathbb{R}^d \times [0, T] \to \mathbb{R}, \qquad c(t) \to J(c)\,,$$

where $c(t) \in \mathbb{R}^d$ for $\forall t$. $c(t)$ is the design variable to be optimized. In many cases, the objective function is an ouput of a PDE based spatial-time simulation of conservation laws. For example, oil reservoir simulations may employ PDEs of the black-oil model, in which gas, water, and oil phases satisfy a set of conservation laws [28]. We may optimize the water injection logistics to maximize total oil production in a time window. Another example is the DNS simulation for turbulent internal flow, in which mass, momentum, and energy satisfy a set of conservation laws [103]. We may optimize the wall geometry to minimize the pressure loss. The simulations employed for optimization are generally implementations of PDEs for conservation laws, and can provide accurate predictions for the output quantity of interest.

In many cases, such simulations implement complicated PDEs and exotic solution techniques. Besides, the PDEs and their numerical implementations may not be accessible. For example, many industrial reservoir simulators are proprietary, such as *Eclipse* (Schlumberger), *VIP* (Halliburton), and *PSim* (ConocoPhillips). During an internship in ConocoPhillips, I got a chance to look at *PSim*'s source code. It turns out to be a million-line, *Fortran77*, legacy code developed for decades. Therefore, even when the source code and the documentation are availble, it can still take tremendous amount of man power to access, modify, and maintain the code. When we have no insights of the PDEs or can not access its space-time solution, the only duty of such simulations is to calculate the objective function

$$J = J(c; \kappa) \in \mathbb{R}\,, \tag{1}$$

where $\kappa \in \mathbb{R}^n$ are some pre-defined model properties (for example, porosity and permeability in oil reservoir simulation). c is the design variable to be optimized. We call such simulations *black-box*.

Fortunately, although it may be difficult to access the details of the PDEs, it's often easy to write down the abstract form of the PDE. For example, many reservoir simulations can be written as [28]

$$\begin{aligned} \eta \frac{\partial u}{\partial t} + \nabla \cdot F(u, \kappa) &= q(u, c) \\ J &= \int_0^T \int_\Omega j(u, c)\ \mathrm{d}t\mathrm{d}\mathbf{x}\,, \end{aligned} \tag{2}$$

with an impermeable reservoir boundary condition and known boundary conditions at well locations. $u \in \mathbb{R}^n$ includes field quantities like pressure and saturation. $\eta, \kappa \in \mathbb{R}^n$ are geology properties like porosity and permeability. $q \in \mathbb{R}^n$ is a known model for the volumetric change of u. $F \in \mathbb{R}^n$ is an unknown flux term. c is the space-time-dependent design variable such as water injection rates. Besides, many simulators can output the discretized $u(t, x)$ in addition to J. For each evaluation of J given κ and c, the discretized space-time solution is generally a large file. We will call such simulations *gray-box* [49] , in order to distinguish *black-box* simulations where the PDE and its implementation is unknown and the space-time solution is unavailable. Besides, in many cases, the simulations do not implement adjoint; this property is shared by gray-box and black-box simulations. We will discuss the role of adjoint in more detail in section 1.3. If the PDE and its implementation is known, and if adjoint is implemented, we call such simulations *open-box*. We summarize their differences in Table 1.

Table 1: Comparison of black-box, gray-box, and open-box simulations

	PDE	Implementation	Space (or space-time) solution	Adjoint
Black-box	✘	✘	✘	✘
Gray-box	✔✘	✘	✔	✘
Open-box	✔	✔		✔

Thanks to the complicated physics models in the PDEs and large space-time scale involved, running a PDE-based simulation can be computationally expensive in many cases. In addition, optimization problems may entail a large number of PDE simulations, especially when the design space is high-dimensional. Therefore, optimization based on PDE-simulations can be costly.

We will restrict our attention to optimization based on gray-box simulations with a high-dimensional design space.

1.2 Prototypes for conservation laws and objective functions

We will consider two prototypes of conservation laws governing the gray-box simulation: time-dependent and time-independent conservation laws.

For time-dependent conservation laws, the prototype is

$$\frac{\partial}{\partial t}\left(\eta_i u_i\right) + \nabla \cdot F_i(\mathcal{D}u, \kappa) = q_i(u, c)\,, \qquad i = 1, \cdots, n\,, \tag{3}$$

where $t \in [0, T]$ is time; $x \in \Omega \subseteq \mathbb{R}^n$ is the spatial coordinate. Ω may depend on the design c. q_i's are the source terms that may also depend on c. c is the design variable and may be space-time dependent. The boundary condition is known. F_i's are the flux functions. The flow variables are $u = \{u_1, \cdots, u_n\}$. $\mathcal{D}u = \left\{u_1, \nabla u_1, \cdots, \nabla^{i_1} u_1; \cdots; u_n, \nabla u_n, \cdots, \nabla^{i_n} u_n\right\}$, where ∇^j indicates the jth order spatial derivative tensor. We assume $i_1, \cdots, i_n$, i.e. the maximum order of derivatives, are known. $\eta_i = \eta_i(x)$, $\kappa = \kappa(x)$ are problem dependent spatial variables, and are assumed known. The discretized space-time solution of Eqn(3) given by a gray-box simulation is written as $\hat{u}(t_i, \mathbf{x}_i; c)\,,\ i = 1, \cdots, N$, where $t = \{t_1, \cdots, t_N\}$ indicates the discretized time, and $\mathbf{x}_i$ indicates the spatial discretization at time t_i.

The objective function is defined by

$$J = \int_0^T \int_\Omega j(u, c) \mathrm{d}\mathbf{x}\mathrm{d}t \tag{4}$$

Notice j can depend explicitly on c, while u depends implicitly on c. We will consider unconstraint optimization unless otherwise specified.

Similarly, for time-independent conservation laws, the prototype is

$$\nabla \cdot F_i(\mathcal{D}u, \kappa) = q_i(u, c)\,, \qquad i = 1, \cdots, n\,, \tag{5}$$

and the objective function is defined by

$$J = \int_\Omega j(u, c) \mathrm{d}\mathbf{x} \tag{6}$$

Notice the time-independent prototype can be seens as a special case of the time-dependent prototype. In many cases, the solution of time-independent PDEs can be seens as the converged solution of time-dependent PDEs. If we can evolve Eqn(3) until $\frac{\partial u_i}{\partial t} \to 0$, the converged solution is a solution of Eqn(5). In numerical implementations, this corresponds to the pseudo time marching scheme for solve time-independent PDEs. Therefore, we will mostly consider the time-dependent prototype.

For illustration purposes, consider an example for the time-dependent prototype. Consider a simulator modeling two phase flow in porous media. One of the simplest yet classical model for two-phase porous media flow is the Buckley-Leverett model [31, 32]. It models the displacement process of two-phase flow due to capillary pressure and Darcy's law. The PDE, Buckley-Leverett equation, is

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{1 + A(1-u)^2}\right) = c\,, \tag{7}$$

where $x \in [0, 1]$ is the space domain; $u = u(t, x)$, $0 \le u \le 1$, is the saturation of phase I (e.g. water), and $1 - u$ is the saturation of phase II (e.g. oil); $A > 0$ is a parameter dependent on the physical property of

the two phases; $c = c(t, x)$ is the design variable. $c > 0$ models the injection of phase I replacing phase II; and $c < 0$ vice versa. Suppose A is an unknown, we can use the prototype Eqn(3) to model the PDE.

As an example, we may want to control the flow through c, such that the saturation at $t = T$ is close to a target saturation $u^*(x)$. We may introduce a Tikhonov regularization into the objective function to model the control cost [33]. Hereby the objective function

$$\begin{aligned} J &= \int_x |u(T,x) - u^*(x)|^2 \,\mathrm{d}x + \eta \int_t \int_x c^2(t,x) \mathrm{d}x\mathrm{d}t \\ &= \int_t \int_x |u(t,x) - u^*(x)|^2 \,\delta_T(t) + \eta c^2(t,x) \mathrm{d}x\mathrm{d}t\,, \end{aligned} \tag{8}$$

where $\eta > 0$ models the price of control, and $\delta_\cdot(\cdot)$ is the Direc delta function. The optimization problem is

$$c^* = \arg\min_{c \in \mathcal{C}} J \tag{9}$$

where $\mathcal{C}$ can be the L_2-function Hilbert space.

1.3 Gradient-free and gradient-based optimization

Existing methods for optimization can be classed into two categories: using gradient information (gradient-based optimization) and not using gradient information (gradient-free optimization) [1, 37]. Gradient-free optimization methods require only the availability of objective function values but not derivative information[22]. Because of its mild requirement of the simulation, gradient-free optimization methods are suitable for optimization problems based on *black-box*, *gray-box*, and *open-box* simulations, thus enjoy a wide range of applicability. However, when the dimension of the design space increases, these methods generally suffer from the *curse of dimensionality*. The term *curse of dimensionality* refers to problems caused by the rapid increase in the search volume associated with adding extra dimension in the search space[23]. It's not uncommon to encounter tens or hundreds of dimensions in real life engineering problems, limiting the applicability of gradient-free optimization in these cases.

Many strategies can be adopted to reduce the design space dimension [48], such as dimensional reduction [50], system decomposition [51], and variable selection [52]. However, these methods generally have additional assumptions or requirements to work properly [48]. For example, some reduced order models reduce the design space to a lower-dimensional subspace called *active subspace* [2, 3]. Optimization may be performed only in the active subspace. The active subspace is generally estimated by sampling the relationship between the design variables and the objective function. However, when the original design space is high-dimensional, estimating the active subspace may still require an astronomical number of objective function evaluations. Some reduced order models considers the discretized-in-space ODE of the flow equations, and reduces the dimension of the ODE using techniques such as proper orthogonal decomposition [4]. But such methods require the ODE of the flow equations, which is not suitable under the gray-box assumption.

In contrast, gradient-based optimizations use gradient information to locate a local optimum. A well-known example is the quasi Newton's methods [24]. Generally gradient-based methods require less number of simulations to converge than gradient-free methods, and are more efficient at finding local optimum for high-dimensional problems. In addition of requiring $J(c)$ from the simulation, these methods also require $\frac{\partial J}{\partial c}$, the sensitivity. Adjoint methods are efficient methods for sensitivity analysis[26] for open-box models. Continuous adjoint method develops the continuous adjoint equations from the continuous PDE of the simulation through Lagrange multiplier; and it requires the PDE of the simulation. Discrete adjoint methods applys variational methods directly to the discretized PDE; and it requires the discretized PDE (i.e. the numerical implementation) of the simulation. Another popular method to compute sensitivity is automatic differentiation (AD)[27]. AD exploits the fact that every computer program can be broken down into a sequence of elementary arithmetic operations and elementary functions. By applying the chain rule repeatedly to these operations, derivatives can be computed automatically.

Because adjoint methods require the accessibility of the PDE, and/or its implementation details, it can not be used to compute the sensitivity of a gray-box simulation either.

To sum up, gradient-free optimization methods are suitable for gray-box simulations, but suffer from the curse of dimensionality; gradient-based optimization methods are more efficient for high-dimensional problems, but are not suitable for gray-box simulation. This dilemma motivates the development of a new optimization strategy. The objective of my thesis is to design a new optimization strategy that: 1. not require $\frac{dJ}{dc}$ from the gray-box simulation, and 2. be suitable for high-dimensional design space.

1.4 Turbulent flow optimization

Turbulent flow optimization may be a great playground to motivate and demonstrate my proposed method. There are many turbulence models ranging from low to high in terms of complexity and fidelity. Turbulence flow optimization generally considers objective functions of time averaged flow field. RANS model simulates a time averaged flow field directly, but the result can be inaccurate. Optimization based solely on high-fidelity simulations (e.g. LES, DNS) yields the most credible result. However, such optimization is computationally challenging. The challenge is two-folded: Firstly, these simulations generally require a tremendous amount of spatial-time discretization grid for sufficient resolution, and each objective function evaluation can be both lengthy in time and costly in computational resources. Secondly, these high-fidelity simulators generally do not have adjoint capability; so estimations of the gradient, a key to efficient optimization, is difficult to obtain especially for high dimensional design space.

High-fidelity simulations such as LES are generally gray-box. Firstly, LES models a chaotic dynamical systems. It is not trivial to obtain the correct adjoint sensitivity [104]. Secondly, LES employs a large number of space-time grid points, thus it can be CPU and memory expensive to implement the adjoint. Thirdly, LES may involve exotic boundary conditions, so the PDE and their implementations may not be easily accessible to engineers who are not experts in computational fluid dynamics. Fourthly, LES simulators can generally produce the space-time solution of all relevant flow fields such as velocity and pressure.

Low-fidelity simulations such as steady RANS can be open-box. Steady RANS solves a time-independent problem. They are generally much cheaper to solve, and easier to implement adjoint. However, they may have worse accuracy than the high-fidelity simulations.

We consider optimization problems based on turbulent flow simulation. When the design space is low, optimization can be performed directly with high-fidelity optimization [105]. For higher dimensional design space, the state-of-art optimizations can only employ low-fidelity models such as RANS. However, there is no guarantee for the low-fidelity model's optimum to be close to the high-fidelity model's optimum. Multifidelity optimization may be applied to turbulence optimization. However, low-fidelity models may not be good representation of high-fidelity models, thus limiting computational savings.

2 Thesis objective / Expected contribution / Timeline

2.1 Thesis objective

1. Develop a method for inferring a surrogate conservation law from the space time solution of an unknown conservation law's PDE simulation.
2. Assess how much computational saving can be achieved using the proposed framework in a return bend optimization problem.

2.2 Expected contribution

1. Enable adjoint gradient computation even if the governing PDE is not available.
2. Demonstrate that efficient gradient-based optimization is possible, even if the underlying simulator does not implement an adjoint.
3. Demonstrate my method's superiority in a high-fidelity turbulent flow optimization with a high-dimensional design space.

2.3 Timeline

- Apr: Write proposal defense slides. Wrap up the proof of theorem 3.
- May: Defend proposal. Demonstrate the complete algorithm on a the 1D Buckley-Leverett example.
- Jun: Setup an LES solver for the return bend testcase in OpenFoam, write a RANS solver with adjoint for the return bend.
- Jul: Hold a committee meeting for progress report.
- Jul-Oct: Implement optimization on the return bend example.
- Aug-Nov: Write thesis.
- Jan: Defend thesis.

3 Twin model

3.1 Review of surrogate methods

As mentioned in chapter 1, straightforward optimization of $J(c)$ using gray-box simulations is not always practical. A well-studied topic to address this problem is *surrogate-based optimization* [37, 38]. It is proposed to achieve optimization at reduced computational cost [39]. A *surrogate* gives a reasonably accurate representation of the high-fidelity model's $J(c)$. We will call the high-fidelity gray-box simulation the *primal model*, and write its objective function as $J(c)$; we will write the surrogate model's objective function as $\tilde{J}(c)$. In surrogate-based optimization, the optimization using the primal model can be replaced by iteratively updating and re-optimizing the surrogate. The design point generated by optimizing the surrogate is verified by evaluating the primal model. The primal model evaluation is then used to update/correct the surrogate. Surrogate-based optimization generally proceeds in this prediction-correction manner until termination.

Surrogate-based optimization methods can be distinguished by how the surrogate is constructed. Surrogate construction techniques can be categorized into *physics-based surrogate* and *functional surrogate* [37]. Similar to the primal model, physics-based surrogate simulates the underlying physics. The surrogate is still a representation of the underlying system's physics, but has lower fidelity and is cheaper to evaluate: generally because it uses simplified physics [41, 39] and/or uses coarse discretization [42]. For example, reduced order models based on balanced truncation [4] can be viewed as a physics surrogate, where a reduced dimensional ODE derived from the original ODE is used to describe the system dynamics. A functional surrogate, on the other hand, are generally constructed using only the sample values of $J(c)$ obtained from the primal model (and gradient of $J(c)$ if the primal model can also evaluate the gradient [43]). Functional surrogate does not require insights of the dynamics of the system. Popular functional surrogate techniques include polynomial approximation [44], Kriging [45], artificial neural network [46], etc.

Generally, physics-based surrogates are more expensive to evaluate but more accurate, mainly because they already embed some knowledge of the system's physics. However, physics-based surrogates are dedicated to the specific systems of interest. The construction of physics-based surrogates often require expert insights and significant coding manpower [37]. Functional surrogates, on the other hand, are generic to a wide class of problems. The price paid is they may require considerable amount of primal model evaluation to achieve the same accuracy as physics-based surrogates.

Designing a good physics-based surrogate requires expert insight (e.g. the choice of empirical formulas or the negligence of unimportant physics components), thus limits its applicability. Besides, many physics-based surrogates are only good approximations of the primal model within a certain range of physics settings. For example, the thin airfoil theory is only valid before stall. If one is to optimize the attack angle, the thin airfoil models may no longer provide good estimation of the primal model when the attack angle increases [53]. Another example is from turbulent modelling. Some simplified fluid models are derived under the assumption of low Reynolds number [55], while some are derived under high Reynolds number [54]. In optimization, however, design variables move at each iteration. A low-fidelity model may gradually lose its accuracy as the design variables move away from the surrogate's applicable region. If one is to optimize the attack angle using a surrogate of thin airfoil theory, or optimize flow speed using a low or high Reynolds number fluid model, the quality of the optimum can be questionable.

3.2 Infer physics-based surrogate by the space-time solution

To alleviate the drawbacks in surrogates, we can *correct* the surrogate's objective function iteratively online to improve its accuracy. Generally, existing methods use $J(c)$ to correct the surrogate's $\tilde{J}(c)$. One method is *Bayesian model calibration* [10, 5]. This method models the error, $J(c) - \tilde{J}(c)$, as a deterministic but unknown realization of a stochastic process. Hereby infer the error at unsampled c via a Bayesian framework, using existing samples of $J(c)$ and $\tilde{J}(c)$. The inferred error is then added to $\tilde{J}(c)$ to provide a corrected surrogate. Another method is *multi-fidelity trust-region method*, i.e. to bound

the search step inside a *trust region* during each optimization iteration [17]. The surrogate is corrected to guarantee good estimation of the primal model within a trust region (e.g. satisfying the *fully-linear* property [16]), before performing optimization using the surrogate within the trust region.

These research efforts (Bayesian model calibration, multi-fidelity trust-region methods, etc) can be viewed as overlaying the existing surrogate model with an additional *functional* surrogate for correction. In order for the correction to improve the existing surrogate to a desirable accuracy, a large number of primal model evaluations may be required. As we mentioned earlier, this is a common problem suffered by functional surrogate.

The problem roots in using $J(c)$ to correct a functional surrogate. For a single-objective optimization problem, each high-fidelity simulation yields only a single $J(c)$ to correct the surrogate. When the design space is high, the correction is clearly inefficient. Instead, we may infer an adaptive physics-based surrogate. The physics of the surrogate should be adaptive. In other words, it should be able to morph its representation of the underlying system's physics on-the-fly, so as to mimic the behavior of the primal model for a wider range of physics settings. According to our gray-box assumptions in chapter 1, the only knowledges available in adapting the physics-based surrogate are the sampled $J(c)$'s and corresponding space-time solutions $u(t, x; c)$'s.

Is it feasible to infer a physics-based surrogate? Consider a general dynamical system

$$\dot{u} = \mathcal{L}(u)\,, \tag{10}$$

where $u = \{u_1, \cdots, u_n\}$, $u_i = u_i(t, x)$, $i = 1, \cdots, n$, $x \in \mathbb{R}^n$. $\mathcal{L}$ is a differential operator known as the Hamiltonian of the system [57]. Suppose an observer can take snapshots of u at any t. It has been shown that inferring $\mathcal{L}$ from the snapshots is an intractable problem, no matter how many snapshots are observed [56]. Although Eqn(10) offers the most general description of the primal model, we may not be able to infer a physics-based surrogate just based on Eqn(10).

However, inferring the differential operator $\mathcal{L}$ is not necessary. As discussed in chapter 1, the PDEs for the conservation laws can be written as prototypes Eqn(3) or Eqn(5). Therefore, the problem of adapting the physics of the physics-based surrogate reduces to the problem of adjusting a set of functions F_i or q_i, for $i = 1, \cdots, n$.

We **use the space-time solution** of the gray-box simulations to infer these functions. There are several benefits to use the space-time solution [1]. Firstly, in conservation law simulations, the flow quantities only depend on the flow quantities in an older time inside a *domain of dependence*. When the timestep is small, the domain of dependence can be small too. For example, for scalar conservation laws without exogenous control, we can view the conservation law for one timestep as a mapping $\mathbb{R}^\omega \to \mathbb{R}$, where $\omega \in \Omega$ is the domain of dependence. Loosely speaking, by applying such mapping repeatedly to all $x \in \Omega$ and $t \in [0, T]$ (as well as applying the boundary condition), we perform a space-time simulation of the conservation law. Generally, the size of ω is be very small, making it feasible to infer the mapping. In my master thesis, such a property is termed *stencil locality* [1].

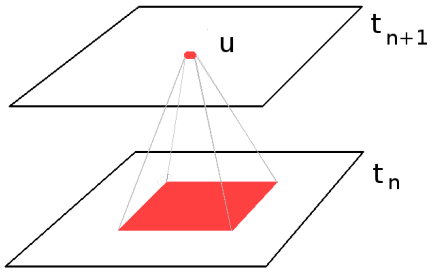

Figure 1: Domain of dependence: in conservation law simulations, the flow quantities at a given location depends on the flow quantities at an older time only within its domain of dependence ω. The domain of dependence can be much smaller then the overall spatial domain Ω when the timestep is reasonably small.

Secondly, the space-time solution at almost *every* space-time grid point can be viewed as a sample for the mapping $\mathbb{R}^{\omega} \to \mathbb{R}$. Because the number of space-time grid points in gray-box simulations is generally large, we have a large number of samples to infer the mapping. In the prototype PDEs Eqn(3) and Eqn(5), such a mapping is determined by the adaptive function F_i's or q_i's. Therefore, the adaptive functions can be inferred with a large number of samples, which makes the inference potentially accurate. In my master thesis, such a property is termed *physics invariance* [1].

Thirdly, in many optimization problems, the design space is high only because the design is space and/or time dependent. In order to parameterize the space-time dependent design, a large number of design variables will be employed. However, the flow quantities only depends on the design variables in the domain of dependence. Therefore, even if the overall design space is high, it can still be feasible to infer the mapping.

To sum up, we propose to infer a new type of surrogate using the prototype equations Eqn(3) or Eqn(5) and the space-time solution of the gray-box simulation. The proposed surrogate is called *twin model*.

3.3 Twin model as an inverse problem

Conventionally, we have a given PDE, and want to compute its space-time solution. However, in twin model we want to infer the governing PDE to match a given space-time solution. Finding a suitable prototype equation can be viewed as an inverse problem. Inverse problems typically involve the estimation of certain variables based on indirect measurements of these quantities. A model that computes indirect measurement given the estimation variables is called a *forward model*. If we parameterize the functions F_i's or q_i's in the prototype equations, we get an inverse problem of estimating such parameters.

Methods for solving inverse problems can be classed into two categories: statistical estimation and parameter identification. Statistical estimation methods adopts a statistical description of the estimation variables. Generally the measurement is assumed noisy, with the noise described by a probability measure. By computing the likelihood of the measurements given the estimation variables, Bayesian inference can be performed to obtain a point estimate or a probability description of the estimation variables [96]. Such methods generally keep track of an ensemble of estimation variables. In our problem, this is equivalent to maintain an ensemble of twin models with different function F_i's or q_i's. Because each twin model is a PDE simulation; it can be expensive to maintain such an ensemble. In addition, it is inappropriate to model the gray-box simulation's space-time solution as noisy.

Another category is parameter identification. Parameter identification infers the estimation variables from the measurements, so the measurement computed from the forward model and the actual mea-

surement can match [96]. Loosely speaking, in twin model we infer the prototype equations so that a metric of the space-time solution mismatch is minimized. Parameter identification treats the estimation variables deterministically. Gradient-based optimization can be used in parameter identification to efficiently infer the optimal estimation variables. We will employ parameter estimation to infer the twin model.

Twin model is an open-box. Therefore, adjoint method can be implemented for efficient parameter identification. Let the metric of the space-time solution mismatch be L, the flow variables solved by twin model be $\tilde{u}$, the variables parameterizing the twin model be ξ, and the forward model be $\tilde{R}(\tilde{u}, \xi) = 0$. Adjoint method proceeds by linearizing L and $\tilde{R}$,

$$\delta L = \left(\frac{dL}{d\tilde{u}}\right)^T \delta\tilde{u} \tag{11}$$

$$\frac{\partial \tilde{R}}{\partial \tilde{u}}\delta\tilde{u} + \frac{\partial \tilde{R}}{\partial \xi}\delta\xi = 0 \tag{12}$$

Introduce a Lagrange multiplier λ, a.k.a. *adjoint state*, we have

$$\begin{aligned}\delta L &= \left(\frac{dL}{d\tilde{u}}\right)^T \delta\tilde{u} - \lambda^T \left(\frac{\partial \tilde{R}}{\partial \tilde{u}}\delta\tilde{u} + \frac{\partial \tilde{R}}{\partial \xi}\delta\xi\right) \\ &= -\left(\lambda^T \frac{\partial \tilde{R}}{\partial \xi}\right)\delta\xi + \left(\left(\frac{dL}{d\tilde{u}}\right)^T - \lambda^T \frac{\partial \tilde{R}}{\partial \tilde{u}}\right)\delta\tilde{u}\end{aligned} \tag{13}$$

Therefore, the adjoint state satisfies the *adjoint equation*

$$\left(\frac{\partial \tilde{R}}{\partial \tilde{u}}\right)^T \lambda = \frac{dL}{d\tilde{u}}\,, \tag{14}$$

and the gradient is given by

$$\frac{dL}{d\xi} = -\left(\frac{\partial \tilde{R}}{\partial \xi}\right)^T \lambda \tag{15}$$

3.4 Formulation of twin model with fixed structure

In the parameter identification, we need a metric for the mismatch of the space-time solutions. Given the same inputs (design variables, initial conditions, boundary conditions, and exogenous inputs), suppose the space-time solution generated by the primal model is u. An ideal twin model should provide a space-time solution $\tilde{u}$ such that

$$\frac{1}{T}\int_{t=0}^{T}\int_{\mathbf{x}\in\Omega} w^2(t, \mathbf{x}, u, \tilde{u})(\tilde{u} - u)^2\, dt d\mathbf{x}\,, \tag{16}$$

is minimized, where $w^2 > 0$ is a weight possibly depending on t, $\mathbf{x}$, u, and $\tilde{u}$. We will delay the discussion of w to section 3.8. Right now we assume $w \equiv 1$ for simplicity. Notice the square on $\tilde{u} - u$: it is a differentiable expression whose gradient is smooth.

The space-time solution is discretized rather than continuous. If the twin model and the primal model use the same space-time grid, then Eqn(16) can be approximated by

$$\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}(\tilde{u}_{ik} - u_{ik})^2\, \Delta t_k\, |\Delta\mathbf{x}_i|\,, \tag{17}$$

where $|\Delta\mathbf{x}_i|$ indicates the size of the grid. If the grids are different, then a mapping P from u to $\tilde{u}$ is required. In this case, Eqn(17) would translate to

$$\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}(\tilde{u}_{ik} - P(u)_{ik})^2\, \Delta t_k\, |\Delta\mathbf{x}_i|\,, \tag{18}$$

Right now, we assume the grids are the same for simplicity.

In the following we assume F_i's are unknown in the prototype equations (extension to problems with unknown q_i's is straightforward), the problem of constructing a twin model (with fixed structure) is converted to the following problem. Notice we added a Tikhonov regularization to avoid ill-posedness.

Solve

$$\xi^* = \arg\min_{\xi} L(\tilde{u}(\xi)) = \arg\min_{\xi} \left\{ \frac{1}{T} \sum_{i=1}^{N} \sum_{k=1}^{T} \left(\tilde{u}_{ik} - u_{ik}\right)^2 \Delta t_k \left|\Delta \mathbf{x}_i\right| + \lambda \|\xi\|^2 \right\}, \tag{19}$$

where u is the discretized space-time solution of the primal model, and $\tilde{u}$ is the discretized space-time solution of

$$\frac{\partial \eta_s \tilde{u}_s(t,x)}{\partial t} + \nabla \cdot \left\{ \sum_{k=1}^{m_s} \xi_{sk} g_{sk}(\mathcal{D}\tilde{u}, \kappa) \right\} = q_s(\tilde{u}, c(t,x)), \qquad s = 1, \cdots, n, \tag{20}$$

g_{sk}, $k = 1 \cdots m_s$ are the library of basis functions for F_s. ξ's are the coefficients for the basis functions. $\lambda\|\xi\|^2$ is the regularization with $\lambda > 0$. $\|\cdot\|$ is the L_2 vector norm. The grids of the primal model solver and the twin model solver are assumed the same. Also, the primal model and the twin model use the same design, initial condition, boundary condition, and exogeneous parameters η and κ.

Similarly, for time-independent prototype equations, we have

Solve

$$\xi^* = \arg\min_{\xi} L(\tilde{u}(\xi)) = \arg\min_{\xi} \left\{ \sum_{i=1}^{N} \left(\tilde{u}_{ik} - u_{ik}\right)^2 \left|\Delta \mathbf{x}_i\right| + \lambda \|\xi\|^2 \right\}, \tag{21}$$

where u is the discretized spatial solution of the primal model, and $\tilde{u}$ is the discretized spatial solution of

$$\nabla \cdot \left\{ \sum_{k=1}^{m_s} \xi_{sk} g_{sk}(\mathcal{D}\tilde{u}, \kappa) \right\} = q_s(\tilde{u}, c(x)), \qquad s = 1, \cdots, n, \tag{22}$$

g_{sk}, $k = 1 \cdots m_s$ are the library of basis functions for F_s. ξ's are the coefficients for the basis functions. $\lambda\|\xi\|^2$ is the regularization with $\lambda > 0$. $\|\cdot\|$ is the L_2 vector norm. The grids of the primal model solver and the twin model solver are assumed the same. Also, the primal model and the twin model use the same design, boundary condition, and exogeneous parameters κ.

3.5 A candidate basis library for twin model

Next, we consider choosing an appropriate basis library g. We will focus on twin models with unknown flux functions, although the extension to unknown source terms is straightforward. For simplicity, we discuss one dimensional function $F(\tilde{u})$, $\tilde{u} \in \mathbb{R}$.

Many candidate basis library may be considered, such as polynomial basis functions, Fourier basis functions, radial basis functions, etc. In this section we will focus on a set of basis functions from the viewpoint of multiresolution analysis and wavelet theory.

Wavelet approximation can be explained by Mallat's *multiresolution analysis (MRA)* [29]. In one dimension, MRA is an increasing sequence of closed function spaces $\{V_j\}_{j\in\mathbb{Z}}$ which approximates $L^2(\mathbb{R})$. V_j approximates functions with increasing resolutions as j increases. The spaces satisfy the following

properties:

$$\begin{aligned}
&\text{MRA 1} \qquad \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots \\
&\text{MRA 2} \qquad \bigcup_{j\in\mathbb{Z}} V_j \ \text{ is dense in } \ L^2(\mathbb{R}) \\
&\text{MRA 3} \qquad f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1},\ j \in \mathbb{Z} \\
&\text{MRA 4} \qquad f(x) \in V_j \Leftrightarrow f(x - \frac{k}{2^j}) \in V_j,\ j \in \mathbb{Z},\ k \in \mathbb{Z}
\end{aligned}$$

In wavelet theory, the orthonormal basis for V_j is given by

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)\,, \quad k \in \mathbb{Z} \tag{23}$$

$\phi_{0,0}$ is called the *scaling function*. Projecting a function $f \in L^2(\mathbb{R})$ on the V_j gives

$$\begin{aligned}
P_{V_j} f &= \sum_{k\in\mathbb{Z}} \langle f, \phi_{j,k}\rangle\, \phi_{j,k} \\
&= \sum_{k\in\mathbb{Z}} \left(\int_{\mathbb{R}} f(x)\phi_{j,k}(x)\,\mathrm{d}x \right)\ \phi_{j,k}
\end{aligned} \tag{24}$$

Scaling functions can be either discontinuous (e.g. Haar wavelet [35]) or continuous (e.g. Meyer wavelet [36]). Because we desire differentiability of the twin model, we will focus on continuous scaling functions. As an example, we show the Meyer wavelet scaling function and its spectrum.

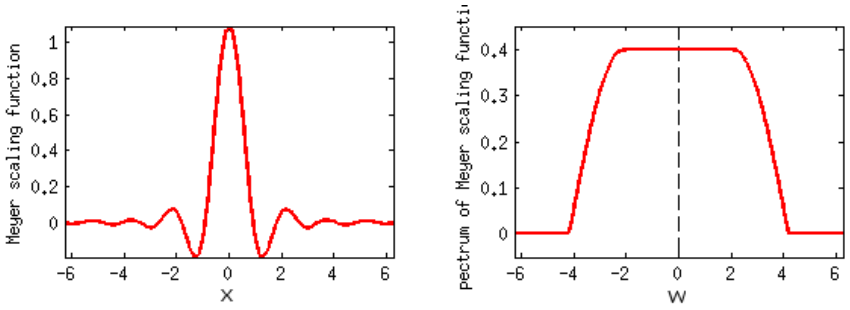

Figure 2: Meyer wavelet scaling function and its spectrum

$\phi_{j,k}(x)$ is localized in the frequency domain, and can be seen as a low pass filter. Projecting a function $f \in L^2(\mathbb{R})$ onto V_j gives a bandlimited approximation of f. As j increments by 1, the bandwidth of the low pass filter will increase by a factor of 2, so does the approximation resolution. Besides, $\phi_{j,k}(x)$ is localized in x, i.e. $\phi_{j,k}(x) \to 0$ as $x \to \pm\infty$. Given j and k, $\langle f, \phi_{j,k}\rangle$ effectively controls the shape of $P_{V_j} f$ only within a bounded domain centered at $\frac{k}{2^j}$. This will be an important property when we discuss the concept of *excited parameter domain* in section 3.7. In practice, it's infeasible and unnecessary to include all $\phi_{j,k}$ $k \in \mathbb{Z}$ into the basis library for a given j. For example, in the 1D water-oil displacement process [31], the water saturation u satisfies $0 \le u \le 1$. Therefore, it's unnecessary to consider k for which $\frac{k}{2^j} \gg 1$ or $\frac{k}{2^j} \ll 0$. We will discuss this topic in greater detail in section 3.7.

In Eqn(3) and Eqn(20), we notice that adding a constant to F's and g's does not change the equations. In other words, it suffices to approximate the flux functions' gradients instead of the flux function value. Therefore, we consider the following functions:

$$\bar{\phi}_{jk}(x) = \int_{-\infty}^{x} \phi_{j,k}(\tau)\,\mathrm{d}\tau \tag{25}$$

The gradient of $\bar{\phi}_{jk}(x)$ is local in frequency and local in x. In this section, we will use $\{\bar{\phi}_{jk}\}$, $k \in \mathbb{Z}$ with a fixed j as the basis library for the flux function.

For flux functions $F(\tilde{u})$, $\tilde{u} \in \mathbb{R}^n$, $n > 1$, the analogy of basis functions is straightforward. The MRA for $L^2(\mathbb{R}^n)$ is $V_{j_1} \otimes V_{j_2} \otimes \cdots \otimes V_{j_n}$, with $\{j_1, j_2, \cdots, j_n\} \in \mathbb{Z}^n$ [29, 69]. Given $\mathbf{j} = \{j_1, \cdots, j_n\}$, The basis is

$$\phi_{\mathbf{j},\mathbf{k}} = \phi_{j_1,k_1}(x_1) \cdots \phi_{j_n,k_n}(x_n), \quad \mathbf{k} \in \mathbb{Z}^n \tag{26}$$

Therefore, the basis for the twin model flux function is

$$\bar{\phi}_{\mathbf{j},\mathbf{k}} = \bar{\phi}_{j_1,k_1}(x_1) \cdots \bar{\phi}_{j_n,k_n}(x_n), \quad \mathbf{k} \in \mathbb{Z}^n \tag{27}$$

Below we plot $\bar{\phi}_{0,0}(x)$ and $\bar{\phi}_{\mathbf{0},\mathbf{0}}(x_1, x_2)$ integrated from Meyer scaling function.

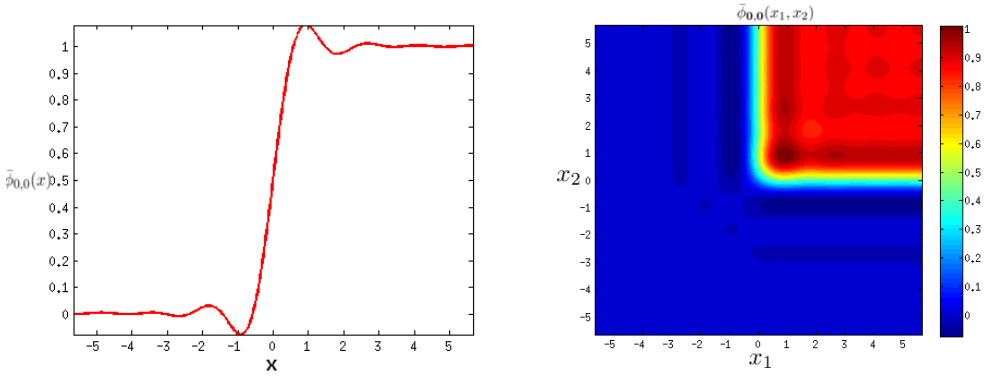

Figure 3: $\bar{\phi}_{0,0}$ in 1D and 2D, integrated from Meyer wavelet scaling function

Clearly, using Eqn(27) as the basis for the flux functions will be problematic for high dimensional $\mathbf{x}$. We will address this problem in section 3.7.

Although function approximation with wavelet scaling function has mathematical rigor, $\bar{\phi}$'s are not ideal bases for our problem. Firstly, the integral of 1D scaling function is rippled, as can be seen in the Meyer wavelet example. This is due to the *localization in frequency* property of scaling functions, known as the Gibbs phenomenon. The rippled flux may cause shock waves in hyperbolic equation simulations. Indeed, the property of localization in frequency is not necessary in our twin model problem. Secondly, generally $\bar{\phi}$ does not have an analytical expression. So $\bar{\phi}$ must be evaluated by numerical integration or interpolation using a lookup table, which can be more computationally intensive than anlytical expressions. Therefore, we prefer a *monotonic* and *analytical* approximation for $\bar{\phi}$.

We choose a sigmoid function to replace $\bar{\phi}_{0,0}$:

$$\bar{\phi}_{0,0}(x) = \frac{1}{2}\left(\tanh(\beta x) + 1\right), \tag{28}$$

where β is a constant determining the gradient of the function at $x = 0$. And

$$\bar{\phi}_{j,k}(x) = \frac{1}{2}\left\{\tanh\left(\beta(2^j x - k)\right) + 1\right\} \tag{29}$$

Or if we don't specify resolution on a dyadic grid 2^j, we can write Eqn(29) as

$$\bar{\phi}_{r,k}(x) = \frac{1}{2}\left\{\tanh\left(\beta(rx - k)\right) + 1\right\}, \tag{30}$$

where $r > 0$. Without further specifications we'll choose $\beta = 1.5$. Below we plot $\bar{\phi}_{0,0}(x)$ and $\bar{\phi}_{\mathbf{0},\mathbf{0}}(x_1, x_2)$ using the sigmoid function.

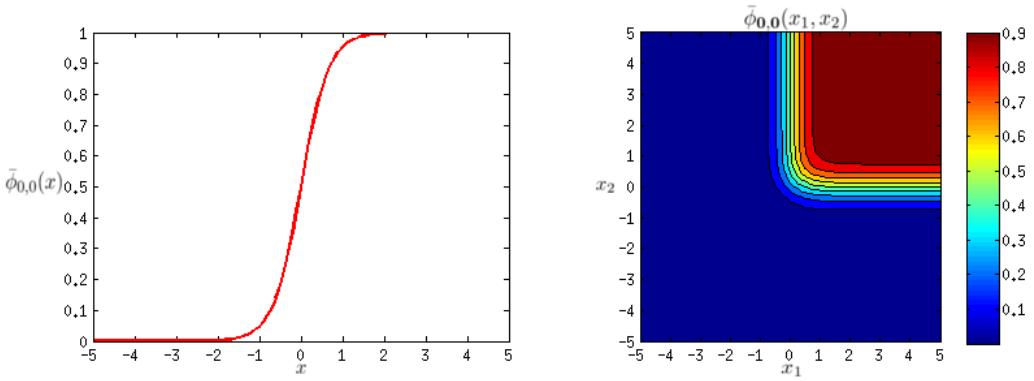

Figure 4: tanh *sigmoid function in 1D and 2D*

3.6 Numerical example: twin model with fixed structure

In this section we demonstrate a numerical example for fitting a twin model with fixed structure. Consider a 1D Buckley-Leverett equation

$$\frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} = 0 \quad x \in [0,1]\,,\ t \in [0,1] \tag{31}$$

with periodic boundary condition

$$u(x=0) = u(x=1) \tag{32}$$

and initial condition

$$u(t=0) = u_0 \tag{33}$$

Buckley-Leverett equation is a simple model for 1D, two-phase, porous media flow driven by capillary pressure and Darcy's law [31]. u indicates one phase's saturation and $0 \le u \le 1$. The flux function $F(u)$ depends on the phases and the porous media. A popular $F(u)$ is

$$F(u) = \frac{u^2}{1 + A(1-u)^2}\,, \tag{34}$$

where A is a constant. In the following we assume the blackbox simulation solves Eqn(31) with the flux given by Eqn(34), and $A = 2$.

Assume $F(u)$ is unknown, we will fit a twin model using the blackbox simulation. As discussed in section 3.4 and 3.5, the twin model can be written as

$$\frac{\partial \tilde{u}}{\partial t} + \frac{\partial}{\partial x}\left(\sum_{k=1}^{m} \xi_k \bar{\phi}_{r,k}(\tilde{u})\right) = 0 \tag{35}$$

with the same initial and boundary conditions. We will use Eqn(30) for $\bar{\phi}$. Both the black-box simulation and the twin model use a second order finite volume discretization and Crank-Nicolson time integration scheme.

We need to determine r and the range for k a prior. A larger r means higher resolution of the flux, but the twin model will be more computationally costly to fit. For this example, we specify $r = 10$. Besides, not all $k \in \mathbb{Z}$ are required in the twin model. As mentioned before, the gradient of each basis indexed by k is non-zero only around $[\frac{k}{r} - \frac{1}{\beta r}, \frac{k}{r} + \frac{1}{\beta r}]$. In our problem, $0 \le u \le 1$, so it suffices to choose corresponding k's. We will choose $k = -1, 0, \cdots, 11$.

The objective of fitting the twin model is given in Eqn(19), and is itself an optimization. We use an automatic differentiation module *numpad* [71] to compute $\frac{dJ}{d\xi_k}, k = 1, \cdots, m$. Since the gradient information is available, we consider using a quasi-Newton method for the optimization. Quasi-Newton

methods build the Hessian approximation iteratively using gradient, and can greatly accelerate convergence [74]. When the degree of freedom of the optimization is high, the memory required to store the Hessian matrix can be large. To reduce the memory requirement, we can use the *low-memory Broyden-Fletcher-Goldfarb-Shannon* (L-BFGS) algorithm [73, 76, 75]. L-BFGS approximates the Hessian using only the gradients at newer previous iterations, and inverse the approximated Hessian efficiently using the Sherman-Morrison formula. For this work, we use the implementation of L-BFGS in Nlopt [73].

Although the objective is to minimize $\sum_{i=1}^{N}\sum_{k=1}^{T}\left(\tilde{u}_{ik}-u_{ik}\right)^2\Delta t_k\left|\Delta\mathbf{x}_i\right|$, we will not use it as the objective function directly and perform optimization in one shot. If $\tilde{F}(u)$ deviates from $F(u)$ a lot, then $\tilde{u}(x,t)$ can deviate from $u(x,t)$ significantly even at a small t. Therefore, solving the twin model and its adjoint in $t=[0,T]$ without an educated $\tilde{F}(u)$ can be a waste of computation resources. To improve efficiency, we propose a progressive optimization procedure:

$\xi^*=\mathbf{0}$;
Set integers $2=i_1<\cdots<i_M=T$;
for $I=i_1,\cdots,i_M$ **do**
  Optimize

$$\xi^*\leftarrow\arg\min_{\xi}\sum_{i=1}^{N}\sum_{k=1}^{I}\left(\tilde{u}_{ik}-u_{ik}\right)^2\Delta t_k\left|\Delta\mathbf{x}_i\right|$$

  with initial guess ξ^*.
end

Algorithm 1: Progressive optimization procedure

Notice $k=1$ corresponds to the initial condition. Since the twin model uses the same initial condition as the black-box model, $\tilde{u}_{i1}-u_{i1}$ is always zero. Choosing the integer sequence $i_1,\cdots,i_M$ can be problem dependent. Our experience shows the sequence should be denser at small i, and sparser at larger i. The tolerance of each sub-optimization problem does not need to be tight, except for the last iteration where $I=T$. In our problem, we use $i_l=\min\left\{1+2^l,T\right\}$, a relative tolerance of 10%, and maximum 10 iterations for the sub-optimization problems.

The initial condition is shown in Fig 5. Notice u_0 does not cover the entire range of $0\le u\le 1$: we have $\max(u_0)=0.89$, $\min(u_0)=0.07$.

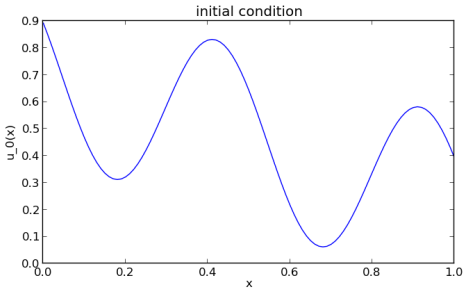

Figure 5: Initial condition $u_0(x)$

In Fig 6 we compare the converged twin model's flux $\tilde{F}(u)$ and its gradient $\frac{d\tilde{F}}{du}$ with $F(u)$ and $\frac{dF}{du}$. The range $u\in[\min(u(t,x)),\max(u(t,x))]$ is colored green.

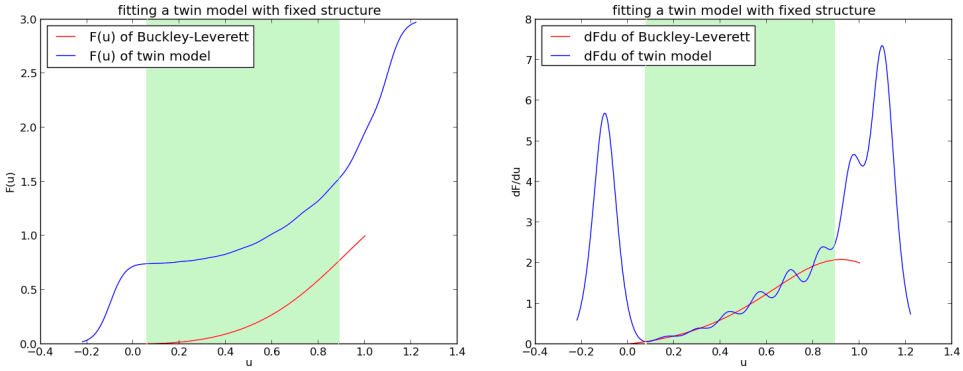

Figure 6: Compare $F(u)$ and $\frac{dF}{du}$ with $\tilde{F}(u)$ and $\frac{d\tilde{F}}{du}$

In Fig 7 we compare the solution $u(t,x)$ with the converged twin model's solution $\tilde{u}(t,x)$.

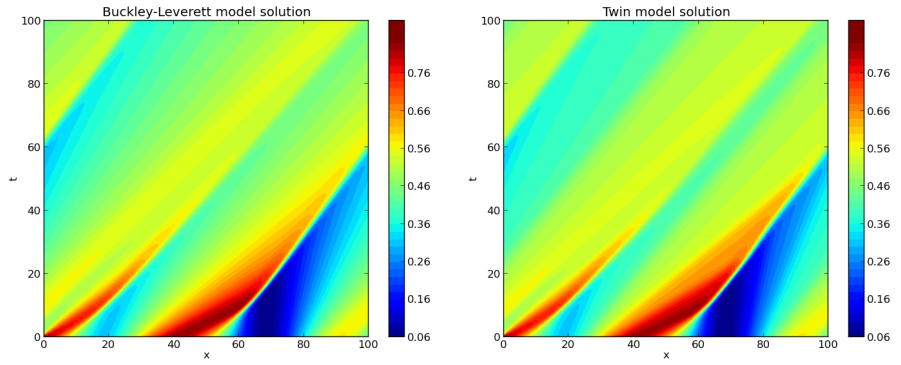

Figure 7: Compare $u(t,x)$ with $\tilde{u}(t,x)$

If we increase the flux resolution, the flux and the solution should have a better fit. For example, we use $r = 30$, $k = -3, \cdots, 33$. The flux comparison is shown in Fig 8

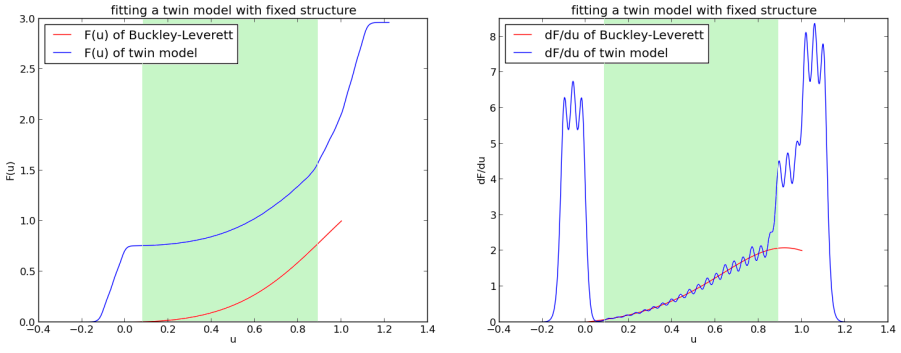

Figure 8: Compare $F(u)$ and $\frac{dF}{du}$ with $\tilde{F}(u)$ and $\frac{d\tilde{F}}{du}$ with higher flux resolution

The solution comparison is shown in Fig 9

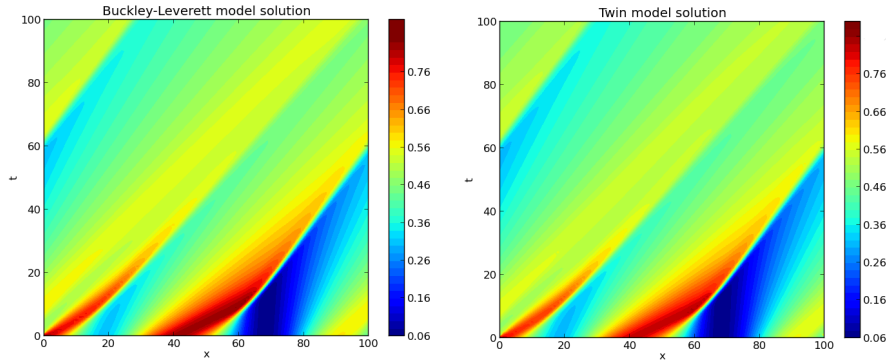

Figure 9: Compare $u(t,x)$ with $\tilde{u}(t,x)$ with higher flux resolution

Comparing Fig 6, 7 with Fig 8, 9, we observe a refined flux basis gives a more accurate twin model. However, we seek for an educated way to adaptively refine the basis and to control the computation cost. Therefore we will investigate *twin model with adaptive structure* in the next section.

3.7 Twin model with adaptive structure

A twin model with adaptive structure should be able to adaptively refine its basis $\bar{\phi}$ on-the-fly during fitting the basis coefficients ξ. For example, in section 3.6's numerical example, the value of $u(t,x)$, $t \in [0,T]$, $x \in [0,1]$ is bounded, i.e. $0 < u_{\min} \leq u(t,x) \leq u_{\max} < 1$; therefore as long as $\nabla\tilde{F}(u) = \nabla F(u)$ for $u \in [u_{min}, u_{max}]$, we will have $\tilde{u}(t,x) = u(t,x)$ for $t \in [0,T]$ and $x \in [0,1]$. In other words, in the numerical example, $\nabla\tilde{F}(u) = \nabla F(u)$ for $u \in [0,1]$ is a sufficient but not necessary condition for $\tilde{u}(t,x) = u(t,x)$. Intuitively, for some domains u, $\nabla\tilde{F}(u)$ has to approximate $\nabla F(u)$ accurately in order to give a good space-time solution match. We will call such domains *excited domain*, written as $\mathcal{E}$. Notice the excited domain is not a domain of space or time. Clearly $\mathcal{E}$ depends on $u(t,x)$. For u not inside $\mathcal{E}$, $\nabla\tilde{F}(u)$ will have little or no effect on the solution match; therefore we may not certify $\nabla\tilde{F}(u)$'s accuracy outside $\mathcal{E}$ no matter how closely the solutions match. As mentioned in section 3.5, we are interested in the problem of basis selection. Basis selections should only be navigated to $\mathcal{E}$.

To start with, consider a primal model solving the Eqn(3), and a twin model solving

$$\frac{\partial\tilde{u}(t,x)}{\partial t} + \nabla\cdot\tilde{F}(\mathcal{D}\tilde{u},\kappa) = q(\tilde{u},c(t,x))\,, \qquad i = 1,\cdots,n\,, \tag{36}$$

with the same initial condition, boundary condition, and controls. We define $\mathcal{E}$ by its complement $\bar{\mathcal{E}}$:

Definition 1. *Given a primal model, its discretized solution $u(t,x)$, and a twin model Eqn(36). The excited domain $\mathcal{E}$ is a domain of $\tilde{F}(\cdot)$. Let the complement of $\mathcal{E}$ be $\bar{\mathcal{E}}$. Consider a perturbed twin model flux $\tilde{F}_{\epsilon\delta}(\cdot) = F(\cdot) + \epsilon\delta(\cdot)$, $\epsilon \in \mathbb{R}, \delta \in L_2(\mathbb{R}^n)$. The perturbed twin model gives the discretized solution $\tilde{u}_{\epsilon\delta}$.*

A set $e \subseteq \bar{\mathcal{E}}$ if and only if

$$\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}\left(\tilde{u}_{\epsilon\delta,ik} - u_{ik}\right)^2 \Delta t_k \left|\Delta\mathbf{x}_i\right|$$

is independent of ϵ for any δ with support$[\delta] = e$.

This definition requires F a prior, therefore it is not directly implementable. In practice, we have to replace $\tilde{F}$'s baseline F with some approximation. The definition also requires enumeration of all possible δ to validate $\mathcal{E}$. In practice, we have to choose a finite set of δ, for example using the basis function library $\bar{\phi}$. However, even with a finite set of δ, we should not validate $\mathcal{E}$ with this definition numerically.

Basis selection may be performed by regularization [82, 83]. For example, it has been shown that basis selection can be performed by having a Lasso regularization term in the solution mismatch. In our problem, the metric of solution mismatch with Lasso regularization would be

$$\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}\left(\tilde{u}_{\epsilon\delta,ik}-u_{ik}\right)^2\Delta t_k\left|\Delta\mathbf{x}_i\right|+\lambda\|\xi\|_1\,, \tag{37}$$

where $\|\cdot\|_1$ is the L_1 norm. Clearly, a straightforward minimization of Eqn(37) requires solving the twin model for $\tilde{u}$ from $[0,T]$. As mentioned in section 3.6, unless $\tilde{u}(\tau)$ is reasonably close to $u(\tau)$, it does not make sense to keep solving the twin model for $t>\tau$. Therefore, we should not let $\tilde{u}$ deviates from u a lot. We will delay the discussion of Lasso regularization to later in the section. Instead, we will propose a basis selection scheme for time-dependent twin model *without solving the twin model from* $[0,T]$.

To restrict $\tilde{u}$ from large deviation to u, we propose a *global-local error* approach for basis selection. Define *global error*:

$$Err_G=\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}\left(\tilde{u}_{ik}-u_{ik}\right)^2\Delta t_k\left|\Delta\mathbf{x}_i\right| \tag{38}$$

and *local error*:

$$Err_L=\frac{1}{T}\sum_{i=1}^{N}\sum_{k=1}^{T}\left(\tilde{u}'_{ik}-u_{ik}\right)^2\Delta t_k\left|\Delta\mathbf{x}_i\right| \tag{39}$$

Here u is the *one-shot* solution of the twin model solving from $t=0$ to $t=T$ with initial condition u_0. $\tilde{u}'_k$ is the solution of the twin model at $t=t_k$, solving from $t=t_{k-1}$ with initial condition u_{k-1}, for just one timestep. In other words, we solve $\tilde{u}'$ with restart for every timestep, whose initial condition is given by u at a previous timestep. We illustrate the idea in Fig 10.

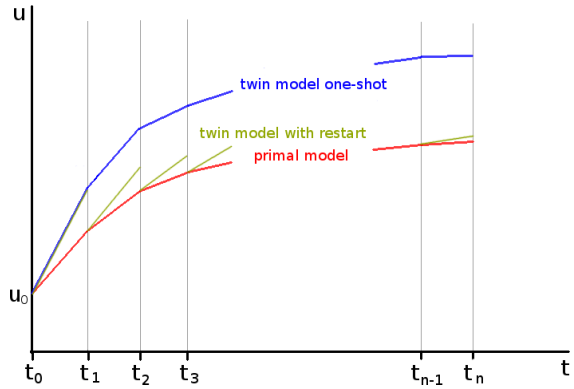

Figure 10: illustration of twin model with restart.

If we are able to bound Err_G with Err_L, we are guaranteed a good solution match even if we do not solve for the one-shot solution $\tilde{u}$. Under what condition can we bound Err_G? We give the following theorem.

Theorem 1. **Global-local error**
Consider the timestepwise mapping of the twin model

$$G : \mathbb{R}^n \mapsto \mathbb{R}^n,\ \tilde{u}^i \to G\tilde{u}^i = \tilde{u}^{i+1}\,, \quad i = 1, \cdots, n\,. \tag{40}$$

If G is a Lipschitz continuous mapping with constant α

$$\|Gx - Gy\|_{L_2} \leq \alpha \|x - y\|_{L_2} \tag{41}$$

then

$$Err_G \leq (1 + \alpha + \cdots + \alpha^{n-1}) Err_L\,. \tag{42}$$

If $\alpha < 1$, then

$$Err_G < \frac{1}{1-\alpha} Err_L \tag{43}$$

In practice, the assumption of $\alpha < 1$ may not be guaranteed. Therefore a small local error may not guarantee a small global error, for example for chaotic dynamical systems. Therefore, we only consider the local error as a tool to select basis, not a tool to finalize the twin model. After selecting the basis using the local error metric, we will still use the global error metric to fit the twin model, for example using algorithm 1.

Using the local error metric, we are able to reduce the twin model into solving a series of one-timestep problems with restart. But in each one-timestep problem, the numerical implementation of $\nabla\cdot$ in Eqn(36) is nonlinear. If we approximate $\nabla\cdot$ linearly, we will be able to use the wealth of knowledge of linear system basis selection. Let $\nabla\cdot$ be approximated by a finite difference operator D. Besides, a stable time integration scheme of twin model's solver is necessary for theorem 1 to hold. Here we let $\frac{\partial}{\partial t}$ be approximated by Backward Euler. Then Eqn(36) becomes

$$-\frac{1}{\Delta t_k}\left(\tilde{u}_k - \tilde{u}_{k-1}\right) + q(\tilde{u}_k, c(t_k)) = \sum_{i=1}^{m} \xi_i D\left(\bar{\phi}_i(\mathcal{D}\tilde{u}_k, \kappa)\right)\,, \quad k = 1, \cdots, T \tag{44}$$

For the kth timestep, let

$$P_{ki} = D\left(\bar{\phi}_i(\mathcal{D}\tilde{u}_k, \kappa)\right) \in \mathbb{R}^N\,, \tag{45}$$

where N is the number of spatial gridblocks, and i is the basis index. Also let

$$Y_k = -\frac{1}{\Delta t_k}\left(\tilde{u}_k - \tilde{u}_{k-1}\right) + q(\tilde{u}_k, c(t_k)) \in \mathbb{R}^N\,. \tag{46}$$

We rewrite Eqn(44) as

$$Y = P\xi\,, \tag{47}$$

where

$$P = \begin{pmatrix} P_{11} & \cdots & P_{1m} \\ \vdots & & \vdots \\ P_{T1} & \cdots & P_{Tm} \end{pmatrix}_{NT\times m}\,, \quad Y = \begin{pmatrix} Y_1 \\ \vdots \\ Y_T \end{pmatrix}_{NT}\,, \quad \xi = \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_m \end{pmatrix}_{m} \tag{48}$$

Unlike linear regression, the objective of Eqn(47) is not to compute ξ, but to select a subset of variables of ξ. This is a *linear* variable dimensional reduction problem. Existing methods can be categorized into two classes [8]: One class is to rank the importance of the variables and to select a subset of important variables directly from candidate variables [82, 79]. Another class is to transform the original variables into a new reduced dimensional space [77]. One of the most popular methods of the second calss is principal component analysis (PCA) [80]. Although PCA can be efficiently computed by singular value decomposition, it has a disadvantage. Because the principal components depend on *all* original variables, they can be difficult to interpret [8]. In the twin model analysis, we want a basis selection scheme that

reflects the excited domain $\mathcal{E}$. Therefore, we will focus on the first class of methods.

Linear variable selection is a rich field. It considers a linear model where each response variable y_i relates with predictors x_{ij} as follows

$$y_i = \xi_0 + \sum_{j=1^m} x_{ij}\xi_j + \epsilon_i\,, \quad i = 1, \cdots, N\,. \tag{49}$$

ϵ's are independent noises. Some of the m predictors may be unhelpful for predicting y, and can be deleted. The task of variable selection is to decide which subset of of variables should be deleted in order to get the *best* equation [83]

$$y_i = \xi_0 + \sum_{j\in\mathcal{S}\in 1,\cdots,m} x_{ij}\xi_j + \epsilon_i\,, \quad i = 1, \cdots, N\,. \tag{50}$$

Different metrics of defining 'best' lead to different variables selection recipes. We can by no means give a complete review of the methods here. Instead, we will only review two most popular classes of methods: *test-based methods*, and *regularization-based methods*.

Test-based methods implements a sequence of hypothesis tests to select variables. They include forward, backward, and stepwise sequential testings [84, 83]. Forward methods start with an empty set of selected variables, then sequentially add the most significant unselected variables, for example the forward orthogonal least squares method for variable selection [79]. Backward methods start with a full set of selected variables, then prunes non-significant predictors one at a time. The stepwise methods alternate between adding and removing variables. However, there has been some critics to test-based methods [83]: Firstly, these methods select variables sequentially using a greedy approach. But the combinations of the best players at each step may not yield the best team. Secondly, the greedy heuristics do not have a firm justification in statistical theory. The testing-based methods attempt to find the best model without first defining what 'best' means.

In contrast, regularization-based methods first define a metric of model goodness, then search for the best model that maximize the metric. The metric is generally the *penalized log likelihood*:

$$\ell(\mathbf{x}, \mathbf{y}, \xi) - c(\xi)\,, \tag{51}$$

where ℓ is the likelihood, c is a measurement of model complexity. Generally the penalty term is

$$c(\xi) = \lambda \sum_{j=1}^{m} |\xi^q|^{1/q}\,, \quad \lambda > 0,\, 0 \le q \le 2 \tag{52}$$

It can be shown subset selection emerges as q is close to 0. Ridge regression chooses $q = 2$, and has no effect of variable selection. Methods based on Akaike information criterion (AIC) [86] and Bayesian information criterion (BIC) [87] chooses $q \to 0$, with different choices of λ. But they yields a non-convex problem which is less attractive for computation. Lasso chooses $q = 1$, which is the smallest value of q that yields a convex problem [82]. The properties of variable selection and computation efficiency motivate us to use Lasso to select basis.

We implement Lasso variable selection using a python module *sklearn* [72]. The result is shown in Fig 11 and Fig 12.

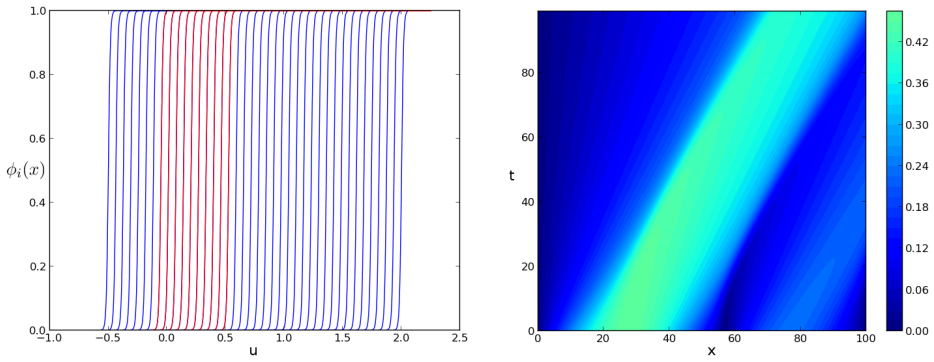

Figure 11: Basis selection using Lasso. The figure on the right side shows the primal model's solution $u(t, x)$. The figure on the left side shows all candidate bases. The selected bases are colored red. Notice how the selected bases correspond to the colorbar of the primal model solution.

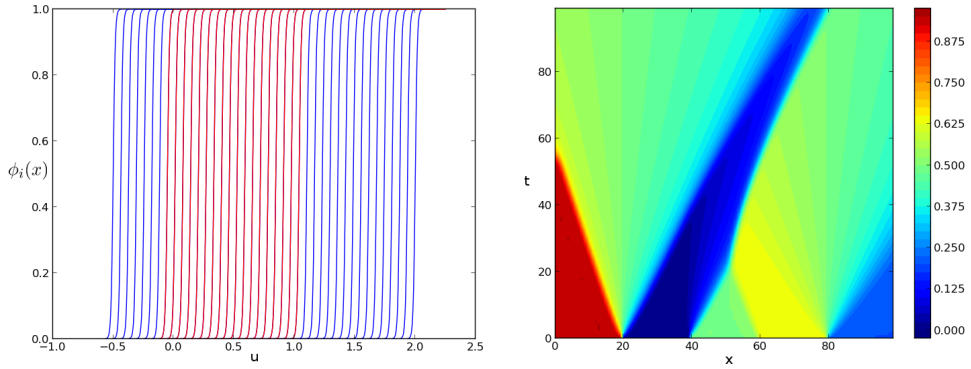

Figure 12: Basis selection with a different $u(t, x)$. Notice how selected bases depend on the primal model solution.

3.8 Future work

The sections above provide a framework to infer the twin model for time-dependent primal models with unknown flux functions. However, there are two issues to investigate.

Firstly, we need to extend the basis selection scheme to time-independent twin model. Right now, the basis selection scheme in section 3.7 is only for time-dependent PDEs. Using a global-local error approach and a finite difference approximation, the basis selection problem was reduced to a linear basis selection problem. However, this approach is not suitable for time-independent twin model. A feasible approach is to use Eqn(37) as the metric of solution mismatch. Unlike time-dependent twin model, we may have to solve time-independent twin model to select basis.

Secondly, we need to explore the weighting scheme used in Eqn(16). Loosely speaking, choosing $w = 1$ is equivalent to setting a uniform sample weight, where each 'sample' corresponds to the solution at a spatial and time grid point. By using a variable w, we should be able to fine tune the twin model accuracy at different locations in the excited domain $\mathcal{E}$. For example, should we assign a larger w for regions with sparser samples? Or should we assign a larger w for regions where more interesting physics happen (e.g. around a shock wave)? The weighting scheme may be ad hoc, but can possibly improve the twin model's accuracy for specific numerical examples.

4 Optimization with twin model

In chapter 3, we developed twin model as a physics-based surrogate method. How should we leverage this surrogate to facilitate optimizations based on gray-box simulations? In this chapter, we answer this question by developing a provably convergent optimization scheme. We also demonstrate our method's superiority for optimization with high-dimensional controls.

4.1 Motivation of the Bayesian approach

We are interested in optimization problems of the following form

$$\begin{aligned} &\min_{c\in\mathcal{C}} J(u,c) \\ \text{where } u \text{ satisfies } &\ \mathcal{R}(u,c)=0 \\ \text{subject to } &\mathcal{C}=\left\{c\in\mathbb{R}^d \middle|\, Ac\le b, V(c)\le 0\right\} \end{aligned} \tag{53}$$

where $u(t,\mathbf{x})\in\mathbb{R}, t\in[0,T]$, $x\in\Omega$ is the space-time solution of a PDE abstracted as $\mathcal{R}$, with given initial and boundary conditions. The controls are $c\in\mathbb{R}^d$. There are l linear constraints defined by $A\in\mathbb{R}^{l\times d}$ and $b\in\mathbb{R}^l$ and s nonlinear constraints defined by $V(c)\in\mathbb{R}^s$. These linear and nonlinear constraints define a feasible region $\mathcal{C}\subseteq\mathbb{R}^d$. The evaluation of J for a given c requires solving the PDE $\mathcal{R}(u,c)=0$ which can be computational costly, such as large eddy simulation (LES) [97, 98] and direct numerical simulation (DNS) [99] in aerodynamic design problems. It is not uncommon for these simulations to take days, weeks, or even longer on a cluster. In addition to expensive evaluations of J, the evaluations of nonlinear constraints $V(c)$ can be similarly expensive. For example, in airfoil design we may aim at reducing drag while keeping lift above a threshold value [100]. Evaluating the lift of a given design requires simulating the flow. Besides, we consider the PDE and its solver as a gray-box. Finally, we consider controls of high dimension, i.e. $d\gg 1$.

We start with optimization problem Eqn(53) with feasible region $\mathcal{C}=\mathbb{R}^d$ for simplicity. There are already many methods that deal with such optimizations. But as mentioned in section 1.3, these methods are not suitable for our problem because they either require adjoint-based gradient of the PDE solver, or suffer from the curse of dimensionality. Can we utilize the twin model together with the primal model, and develop an optimization scheme that addresses these challenges?

The question fits naturally into a *multifidelity optimization (MFO)* framework. MFO performs optimization with multiple models ranging from low-fidelity to high-fidelity. Conventionally, low-fidelity models are cheaper to solve, but yield results with less credibility; high-fidelity models are more credible, but more expensive to solve. MFO balances computation investments between high- and low-fidelity models in order to reach optimum with overall less computation cost. In our problem, the primal model can be considered as a high-fidelity model, and the twin model can be considered as a low-fidelity model.

MFO methods can take various flavors. For example, an MFO method can be as simple as a two-stage optimization [89]. In the first stage it optimizes with the low-fidelity model only. In the second stage it optimizes with the high-fidelity model only, using the first stage optimum as the initial guess. However, we are only interested in *provably convergent* MFO methods. Based on our literature review, there are three types of such methods: *pattern search MFO*, *trust region MFO*, and *Bayesian MFO*. Each type can be viewed as extensions to their *single-fidelity* versions respectively, i.e. *pattern search methods*, *trust region methods*, and *Bayesian optimization methods*. In the following let's review these methods and their corresponding MFO methods.

In *pattern search methods*, the objective function J is evaluated at a set of trial points, called 'mesh', adjacent to the current best c in the design space. This step is called a 'poll step'. If improvement in J is obtained, the current best c is updated; otherwise, the mesh size is shrinked. These steps are iterated until convergence [92]. Pattern search methods can takes advantage of low-fidelity models to reduce computation cost, hereby *pattern search MFO*. It searches for improved design points using low-fidelity models before deciding whether to perform a poll step on the high-fidelity model [93]. However, each

poll step requires order d high-fidelity model evaluations, which can be too expensive to afford in our problem settings.

In *trust region methods*, a functional surrogate is firstly constructed from evaluations of J, then the surrogate is minimized within a trust-region to generate a candidate design point. Depending on the availability of $\frac{\partial J}{\partial c}$, the functional surrogate can either use the gradient information [14], or not use the gradient information [6, 16, 17]. When a low-fidelity model exists, the functional surrogate of J can be replaced by the low-fidelity model, hereby *trust region MFO* [41, 42]. To improve MFO performance and guarantee convergence, researchers have proposed using functional surrogates to account for the discrepancy $\Delta J = J_{high} - J_{low}$ between high- and low- fidelity models within the trust region [5, 94]. However, there are two problems with trust region MFO. Firstly, the convergence proof of such methods requires a condition called *fully-linearity* [16]. The fully-linearity condition bounds the deviation of $\frac{dJ}{dc}$ between the high- and low- fidelity models within the trust region. As mentioned in chapter 1, it can be hard to obtain the gradient from the primal model. Therefore, it can be hard to certify fully-linearity and to support the convergence proof. Secondly, at every iteration, only the model evaluations within or adjacent to a trust region is actually used [94]. In our problem settings, each high-fidelity model evaluation can be so expensive that we should use *all* high-fidelity evaluations available. Therefore, trust region MFO is not suitable for our problem settings too.

In *Bayesian optimization methods*, J as a function of c is assumed to be an unknown realization of a *Gaussian process*. Bayesian optimization starts from a prior distribution of J, and updates its posterior as new samples of $J(c)$ arrive. This procedure is also known as *Kriging* [13, 18]. Then it constructs an *acquisition function* $\rho : \mathcal{C} \mapsto \mathbb{R}\ ,\ c \to \rho(c)$ from the posterior, which measures the expected utility of investing the next sample at c. Finally, we choose the next sample as the maximizer of the acquisition function. Bayesian optimization iterates over the posterior-update step and the ρ-maximization step until convergence [88, 9]. There are two advantages of Bayesian optimization: Firstly, Bayesian optimization uses all available model evaluations to construct the posterior. Secondly, it balances *exploration* with *exploitation* elegantly. Given a set of samples $J(c)$, should we choose the next sample in a sparsely sampled region (exploration), or in a region adjacent to the current best sample (exploitation). Bayesian optimization makes a trade-off between exploration and exploitation via appropriate acquisition functions. This method is provably convergent to the global optimal under mild requirement of the objective function [30, 7].

When a low-fidelity model exists, Bayesian optimization methods can be extended to *Bayesian MFO*. Let the high-fidelity model's objective function be $J(c)$, and the low-fidelity model's objective function be $\tilde{J}(c)$. Bayesian MFO assumes both $J(c)$ and $\tilde{J}(c)$ are unknown realizations of (possibly different) Gaussian processes. Loosely speaking, by connecting the high- and low-fidelity models using a Bayesian framework, we can better predict the high-fidelity model using low-fidelity model evaluations. Bayesian MFO assumes a *Markov property* between $J(c)$ and $\tilde{J}(c)$ [11]:

$$p\left[J(c_0)\middle|\tilde{J}(c_0), \tilde{J}(c_1), \cdots, \tilde{J}(c_N)\right] = p\left[J(c_0)\middle|\tilde{J}(c_0)\right]\ , \quad N \in \mathbb{Z}^+\,, \tag{54}$$

where $p(\cdot|\cdot)$ indicates the posterior. In other words, if we want to predict J at c_0 using $\tilde{J}$, then it suffices to evaluate $\tilde{J}$ only at c_0; we will gain no further information on $J(c_0)$ by observing J at $c \neq c_0$. This assumption is intuitive, and proves to be very useful and convenient [10, 11]. By furthur assuming the Gaussian processes to be stationary, we can obtain

$$\tilde{J}(c) = \rho J(c) + \epsilon(c)\,, \tag{55}$$

where ρ is a constant and $\epsilon(\cdot)$ is a realization of a Gaussian process. Assume we have evaluated J at $\mathbf{c} = \{c_1, \cdots, c_{N_1}\}$, and $\tilde{J}$ at $\tilde{\mathbf{c}} = \{\tilde{c}_1, \cdots, \tilde{c}_{N_2}\}$, then the posterior of $J(c)$ can be constructed from the joint distribution

$$p\left(J(c), J(\mathbf{c}), \tilde{J}(\tilde{\mathbf{c}})\right) = \mathcal{N}\left(\begin{pmatrix} m \\ \mathbf{m} \\ \tilde{\mathbf{m}} \end{pmatrix},\ \begin{pmatrix} \mathrm{var}(J(c)) & \mathrm{cov}(J(c), J(\mathbf{c})) & \mathrm{cov}(J(c), \tilde{J}(\tilde{\mathbf{c}})) \\ \mathrm{cov}(J(\mathbf{c}), J(c)) & \mathrm{var}(J(\mathbf{c})) & \mathrm{cov}(J(\mathbf{c}), \tilde{J}(\tilde{\mathbf{c}})) \\ \mathrm{cov}(\tilde{J}(\tilde{\mathbf{c}}), J(c)) & \mathrm{cov}(J(\mathbf{c}), \tilde{J}(\tilde{\mathbf{c}})) & \mathrm{var}(\tilde{J}(\tilde{\mathbf{c}})) \end{pmatrix}\right) \tag{56}$$

where m is the mean of J, $\tilde{m}$ is the mean of $\tilde{J}$, $\mathbf{m} = m\mathbf{1}_{N_1}$, and $\tilde{\mathbf{m}} = \tilde{m}\mathbf{1}_{N_2}$. The variance-covariances, known as *kernels*, have to be selected a prior. However, these kernels are adaptive in that they have tunable parameters known as *hyperparameters*, so realizations of the Gaussian processes can represent a wide class of functions [95]. The estimation of the hyperparameters and $\rho, m, \tilde{m}$ will be discussed below. The choice of kernels can be various, among the most popular ones are squared exponential kernel and Matern kernel [9]. Because of the additional information of $\tilde{J}(\tilde{\mathbf{c}})$, we can estimate $J(c)$ with less variance than using $J(\mathbf{c})$ alone. This procedure is also known as *coKriging* [19]. With the posterior obtained by the coKriging, Bayesian MFO maximizes the acquisition function to select the next design point c to evaluate J.

How should we apply our twin model to a Bayesian MFO framework? Naturally we would think of a coKriging approach using Eqn(56): We would first simulate the primal model as a high-fidelity model at an initial guess c_0. This gives us the space-time solution u_0 and the objective function $J(c_0)$. We can train the twin model with u_0 using methods proposed in chapter 3. The trained twin model, as a low-fidelity model, can be simulated to obtain $\tilde{J}$ at a set of new design points. Finally, the posterior of $J(c)$ can be constructed using the joint distribution Eqn(56). We may repeat these steps while re-training the twin model at updated design points.

However, this approach is not suitable for our problem settings. The twin model may not be very cheap to evaluate. Contrary to the conventional ideology: "low-fidelity model is cheap", the twin model can be expensive to simulate. Twin model is itself a PDE simulator. Unlike many commericial or open-source gray-box PDE simulators whose code is highly optimized for efficiency, the twin model code can be less efficient and more costly to solve. When the design space dimension d is high, explorations of the design space using evaluations of $\tilde{J}$ suffers from the curse of dimensionality too.

To unleash the power of twin model, we propose using the twin model's gradient $\frac{d\tilde{J}}{dc}$ instead of $\tilde{J}(c)$ for coKriging. The twin model is an open-box and has the adjoint capability to compute $\frac{d\tilde{J}}{dc}$. It has been shown that using the adjoint-based gradient information can reduce the number of function evaluations for Bayesian optimization [47, 43], especially when d is large. In our problem settings, we do not have access to the adjoint-based gradient $\frac{dJ}{dc}$, but it is possible to approximate $\frac{dJ}{dc}$ by $\frac{d\tilde{J}}{dc}$. We can't make general arguments about the approximation accuracy. Even if the twin model can reproduce the space-time solution of the primal model at a training design point, we can't ascertain that their adjoints are close, for example when the twin model solves a chaotic system. However, the theoretical flaw should not prohibit us from investigating practical applications of twin model. Besides, later we will show that our optimization scheme converges to the global optimal for the primal model, no matter how accurate or inaccurate the gradient approximation is.

Here is an example demonstrating why it can be promising to approximate $\frac{\partial J}{\partial u}$ with $\frac{\partial \tilde{J}}{\partial u}$. Consider the numerical example in section 3.6, where the primal model solves:

$$\frac{\partial \tilde{u}}{\partial t} + \frac{\partial}{\partial x}\left(F(u)\right) = c\,, \tag{57}$$

for $c = 0$, with $F(u)$ given by Eqn(34). We trained a twin model Eqn(35) using the space time solution. We are interest in the approximation quality of the twin model's gradient at c adjacent to $c = 0$. The result is shown in

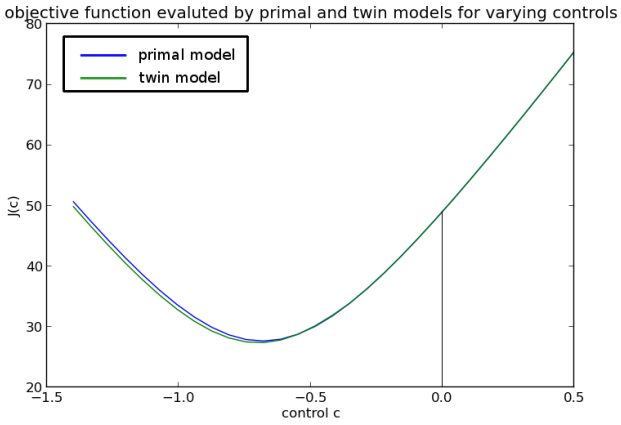

Figure 13: Train the twin model at $c = 0$, the resultant twin model gives good approximation for the $\frac{dJ}{dc}$.

The result is encouraging as the gradient computed by the twin model gives a good approximation of the gradient of the primal model. We reiterate that the good approximation quality benefits from the matching of space-time solution.

4.2 Bayesian modeling of the primal and twin models

We model the relation of $\nabla J(c)$ and $\nabla \tilde{J}(c)$ as

$$\nabla \tilde{J}(c) = \nabla J(c) + \epsilon(c)\,, \tag{58}$$

Assume $\nabla J(c)$, $\nabla \tilde{J}(c)$, and $\epsilon(c)$ are realizations of stationary Gaussian processes. We first model the priors of $\nabla J(c)$, $\nabla \tilde{J}(c)$, and $\epsilon(c)$, when no evalutions of the high- and low-fidelity models are made.

For any c_1 and c_2, assume $\epsilon(x)$ is an unknown realization of Gaussian process $\mathcal{N}(\mathbf{0}, K_\epsilon(\cdot,\cdot))$. $K_\epsilon : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$

$$K_\epsilon(c_1, c_2) = \boldsymbol{I}_{n\times n} k_\epsilon(c_1, c_2; \theta_\epsilon)\,, \tag{59}$$

where θ_ϵ are the hyperparameters of the kernel.
$J(c)$ is treated as a realization of the Gaussian process $\mathcal{N}(\mu, K(\cdot,\cdot))$, $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ where

$$K(c_1, c_2) = k(c_1, c_2; \theta)\,. \tag{60}$$

θ are the hyperparameters of the kernel. Assume

$$\begin{aligned} cov(\nabla J(c_1), \epsilon(c_2)) &= 0 \\ cov(J(c_1), \epsilon(c_2)) &= 0\,, \end{aligned} \tag{61}$$

for any c_1, c_2. Then

$$cov(J(c_1), \nabla \tilde{J}(c_2)) = cov(J(c_1), \nabla J(c_2)) \tag{62}$$

for any c_1, c_2. There are multiple popular choices for the kernels. Generally the squared exponential kernel is a default choice. However, sample functions from this kernel are unrealistically smooth and may not represent the objective function well. Another popular choice is the Matern kernel, especially $5/2$ kernel:

$$K_0(c_1, c_2; \theta) \equiv \sigma^2 \left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right) \exp\left(-\frac{\sqrt{5}d}{\rho}\right)\,, \tag{63}$$

where $d = \|c_1 - c_2\|_{L_2}$. θ are the hyperparameters ρ and σ. The corresponding sample functions are twice differentiable, but not as smooth as the square exponential kernel. It has been shown the Matern

5/2 kernel outperforms other kernels in several Bayesian optimization problems [9]. Thus we choose the Matern 5/2 kernel as our choice. It is worth noting that the kernel function was chosen to be isotropic. In our problem, all the design parameters are the boundary points that controls the return bend geometry, so it makes sense to use a isotropic kernel for simplicity. In general, the metric d does not need to be isotropic, but this comes at the price of having more hyperparameters. Based on the assumptions above, we have

$$K_1(c_1, c_2; \theta) \equiv \frac{\partial}{\partial c_2} cov(J(c_1), J(c_2)) = \sigma^2 \exp\left\{-\frac{\sqrt{5}d}{\rho}\right\}\left(\frac{5}{3\rho^2} + \frac{5\sqrt{5}d}{3\rho^3}\right)(c_1 - c_2) \tag{64}$$

and

$$K_2(c_1, c_2; \theta) \equiv \frac{\partial^2}{\partial c_1 \partial c_2} cov(J(c_1), J(c_2)) = \sigma^2 \exp\left\{-\frac{\sqrt{5}d}{\rho}\right\}\left\{\left(\frac{5}{3\rho^2} + \frac{5\sqrt{5}d}{3\rho^3}\right)\boldsymbol{I} - \frac{25}{3\rho^4}(c_1 - c_2)(c_1 - c_2)^T\right\} \tag{65}$$

We will model $J(c)$ as sampled function with the kernel Eqn(63) with hyperparameters $\theta = (\rho, \sigma)$. We will model $\epsilon(c)$ as sampled function with the kernel

$$\boldsymbol{I}_{n\times n} K_0(c_1, c_2; \theta_\epsilon) \tag{66}$$

with hyperparameters $\theta_\epsilon = (\rho_\epsilon, \sigma_\epsilon)$. Suppose we sample J on $\mathbf{c}$, and sample $\nabla\tilde{J}$ on $\tilde{\mathbf{c}}$, then the joint probability of $J(c)$ at an unsampled c and the sampled data can be constructed. Specifically, the joint probability is a Guassian process with mean

$$(\mu, \boldsymbol{\mu}, \mathbf{0})^T \tag{67}$$

and covariance

$$\begin{pmatrix} K_0(c, c; \theta) & K_0(c, \mathbf{c}; \theta) & K_1(c, \tilde{\mathbf{c}}; \theta) \\ K_0(c, \mathbf{c}; \theta)^T & K_0(\mathbf{c}, \mathbf{c}; \theta) & K_1(\mathbf{c}, \tilde{\mathbf{c}}; \theta) \\ K_1(c, \tilde{\mathbf{c}}; \theta)^T & K_1(\mathbf{c}, \tilde{\mathbf{c}}; \theta)^T & K_2(\tilde{\mathbf{c}}, \tilde{\mathbf{c}}; \theta) + K_\epsilon(\tilde{\mathbf{c}}, \tilde{\mathbf{c}}; \theta_\epsilon) \end{pmatrix} \tag{68}$$

In practice, because the high-fidelity model is never simulated for infinite time to compute the time average, the sampling of J can be noisy. We assume the noise in sampling J due to finite high-fidelity simulation to be a white noise with variance δ_T^2, where T is the physical time running the high-fidelity simulation. Therefore, the covariance of the joint probability may be modified to

$$\begin{pmatrix} K_0(c, c; \theta) + \delta_T^2 & K_0(c, \mathbf{c}; \theta) & K_1(c, \tilde{\mathbf{c}}; \theta) \\ K_0(c, \mathbf{c}; \theta)^T & K_0(\mathbf{c}, \mathbf{c}; \theta) + \delta_T^2 \boldsymbol{I} & K_1(\mathbf{c}, \tilde{\mathbf{c}}; \theta) \\ K_1(c, \tilde{\mathbf{c}}; \theta)^T & K_1(\mathbf{c}, \tilde{\mathbf{c}}; \theta)^T & K_2(\tilde{\mathbf{c}}, \tilde{\mathbf{c}}; \theta) + K_\epsilon(\tilde{\mathbf{c}}, \tilde{\mathbf{c}}; \theta_\epsilon) \end{pmatrix} \tag{69}$$

The relationship of δ_T with T can be determined empirically.

4.3 Optimization using the posterior of the objective function

Using the assumptions in section 4.2, we can derive the posterior of $J(c)$ given samples of J and $\nabla\tilde{J}$. Then Bayesian optimization can be performed with the posterior. However, there are at least three questions to answer before applying Bayesian optimization. The first question is the choice of the acquisition function. The second question is how to optimize the acquisition function. The third question is how to determine the hyperparameters $\theta = (\sigma, \rho)$ and $\theta_\epsilon = (\sigma_\epsilon, \rho_\epsilon)$.

For the first question, generally people use two popular acquisition functions [9]. The first one is the *expected improvement* acquisition function. Let c_s^* be the best design after s previous samples $c_1, \cdots, c_s$, then the next sample c_{s+1} will be drawn with

$$c_{s+1} = \arg\max_{c\in\mathcal{C}} \mathbb{E}\left[\max\left(J(c) - J(c_s^*), 0\right) | \mathcal{S}\right], \tag{70}$$

where S indicates all available samplings on J and $\nabla\tilde{J}$. This formulation chooses the next sample point to maximize the expectation of the improvement: $\max\left(J(c) - J(c_s^*), 0\right)$ conditioned on existing samples,

hereby its name. Another popular method is the *upper confidence bound* acquisition function. The next sample is drawn with

$$c_{s+1} = \arg\max_{c\in\mathcal{C}} \left[\mu(J(c)\big|\mathcal{S}) + \kappa\sigma(J(c)\big|\mathcal{S})\right] , \tag{71}$$

with a tunable κ to balance exploration against exploitation. We will use the expected improvement formulation because it does not have the user tunable parameter κ. The formulation Eqn(70) has a close form under the Gaussian process assumption [9], which makes its evaluation straightforward.

For the second question, we consider using a gradient-driven optimizer with multiple initial guesses, for the following reasons. Firstly, the gradient of the acquisition function with respect to the candidate design point, $\frac{d\rho}{dc}$, can be obtained easily using automatic differentiation. Secondly, we want to reduce the risk of the optimization converging to a non-global-optimal candidate design, so it can be reasonable to optimize with multiple initial guesses. The specific optimizer is yet to be chosen, though we expect little impact on optimization performance due to different choices.

For the third question, there are two popular approaches to treat the hyperparameters. One approach is to use the maximum likelihood estimate of the hyperparameters, in which a point estimate of the hyperparameters are made by maximizing the likelihood function. Another approach is a Bayesian treatment of the hyperparameters. Instead of optimizing the acquisition function with a point estimate of the hyperparameters, it optimize an acquisition function marginalized over the hyperparameters, a.k.a integrated acquisition function. The marginalization is often implemented with an Monte Carlo approach [12]. Generally these two approach have comparible performance [9]. Without further specification our study is conducted using the maximum likelihood estimate approach.

To sum up, the proposed optimization flowchart is shown in Fig14

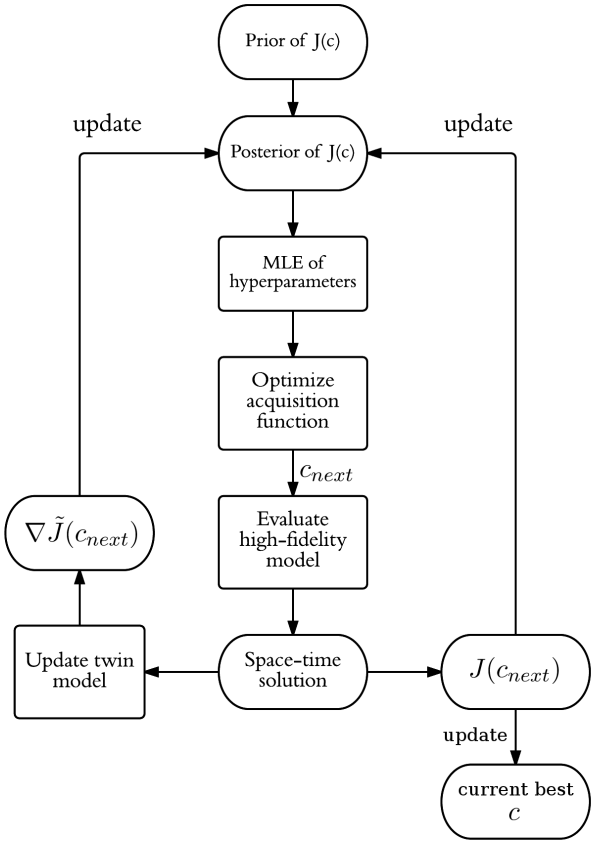

Figure 14: The flowchart of Bayesian optimization with twin model.

Intuitively, as the design space dimension increases, the ability to estimate $\frac{dJ}{dc}$ may become more important for faster J reduction. For illustration, we consider minimizing

$$J(x) = \sum_{i=1}^{\dim} x_i^2 \,. \tag{72}$$

The true optimal is $x = \mathbf{0}$. Suppose the evaluation of J is so expensive that we can only evaluate it for 10 times. When dim increases, we may expect the smallest J in the 10 evaluations to increase. When an estimation of $\frac{dJ}{dc}$ is available, the objective function may be reduced more quickly during the 10 evaluations, even if the gradient estimation is noisy. We demonstrate the superiority of using estimated gradient in Fig15.

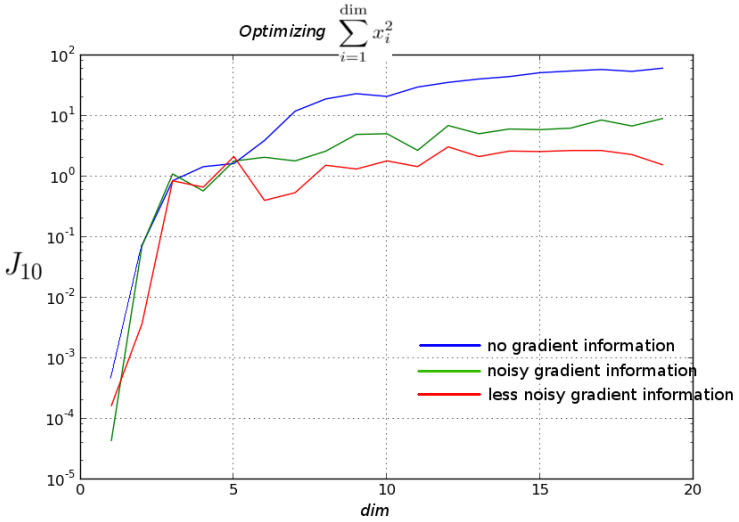

Figure 15: Minimizing $J(x) = \sum_{i=1}^{dim} x_i^2$ with variable dim. The y-axis is the best / smallest J after 10 function evaluations, in log 10 scale. The blue line is obtained from Bayesian optimization with no gradient information. The green line is obtained from Bayesian optimization with a noisy gradient, by setting $\sigma^2 = \sigma_\epsilon^2$, see Eqn(63) and Eqn(66). The red line is obtained from Bayesian optimization with a less noisy gradient, by setting $0.1\sigma^2 = \sigma_\epsilon^2$ We use Monte Carlo to randomly select some initial guesses for x. Each point on these lines represents the averaged J_{10} using a random set of initial guesses.

4.4 Discussion of the convergence property

Bayesian optimization is a deterministic global optimization strategy. Given previous evaluations, it locates deterministically the next design point for evaluation. In the end, the optimization generates an infinite sequence of design points. Clearly, given an optimization strategy, the sequence depends on the underlying function to evaluate, and depends on the initial design. The convergence property of Bayesian optimization can be discussed by the property of the design sequence, using the following theorem [106]:

Theorem 2. Let $W^{k,p}(\mathcal{X})$ be a Sobolev space of functions defined on $\mathcal{X}$. For any initial design in $\mathcal{X}$, a global optimization algorithm converges for all functions in $W^{k,p}(\mathcal{X})$ if and only if the design sequence produced by the algorithm is dense in $\mathcal{X}$ for all functions in $W^{k,p}(\mathcal{X})$.

Therefore, we will investigate whether the sequence produced by the proposed optimization scheme is dense. If we can sample the exact function value, and if we use the expected improvement acquisition function, it has been shown [7, 30] the sequence is indeed dense, under some mild conditions on the objective function. In this section, we extend the theoretical result to sample both the exact function value and the noisy function gradient.

Theorem 3. **The design sequence is dense.**
Assume

1. $\mathcal{X} \in \mathbb{R}^n$, $\|\cdot\|$ be the L_2 norm defined on $\mathcal{X}$.
2. $\mathcal{H}$ and $\mathcal{H}'$ be two reproducing kernel Hilbert spaces of functions on $\mathcal{X}$, with kernels $K(\cdot,\cdot) : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ and $K'(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ respectively.
3. There exist $k : \mathbb{R}^+ \bigcup\{0\} \mapsto \mathbb{R}$ and $k' : \mathbb{R}^+ \bigcup\{0\} \mapsto \mathbb{R}$, such that K and K' satisfies $K(x,y) = k(\|x-y\|)$ and $K'(x,y) = k'(\|x-y\|)$ respectively, for $\forall x, y \in \mathcal{X}$.
4. k and k' has the Fourier transforms $\hat{k}$ and $\hat{k}'$ respectively. They satisfy the asymptotic properties $\hat{k}(u) = \Theta(|u|^{-n-2\nu})$ and $\hat{k}'(u) = \Theta(|u|^{-n-2\nu'})$, as $|u| \to \infty$, with $\frac{1}{2} < \nu < \infty$ and $\nu' = \nu - 1$. (Θ is the asymptotic big Θ notation.)

Let

1. $f \in \mathcal{H}$ be the underlying objective function to maximize.
2. $\epsilon = (\epsilon_1, \cdots, \epsilon_n)^T$, $\epsilon_i \in \mathcal{H}'$, $i = 1, \cdots, n$ be the gradient sample noise. $\epsilon_i \perp \epsilon_j$ for $i \neq j$.
3. g be the noisy gradient of f,
$$g = \epsilon + \nabla f\,.$$

Let the optimization strategy generates the design sequence according to

$$\begin{aligned} x_{s+1} &= \arg\max_{x\in\mathcal{X}} \mathbb{E}\left[\max\left(f(x) - f(x_s^*)\right)\Big|\mathcal{S}\right] \\ x_s^* &= \arg\max_{x\in\{x_1,\cdots,x_s\}} f(x) \\ \mathcal{S} &= \Big\{ \{x_1,\cdots,x_s\}, \{f(x_1),\cdots,f(x_s)\}, \{g(x_1),\cdots,g(x_s)\} \Big\} \end{aligned}$$

Then the sequence $\{x_1, x_2, \cdots\}$ is dense in $\mathcal{X}$ for $\forall x_1 \in \mathcal{X}$.

As a future work, we will prove the theorem.

4.5 Optimization constraints

In the previous sections, we considered only non-constraint optimization for simplicity. In order to deal with the more general formulation Eqn(53), we need to consider inequality constraint optimization. Inequality constraint optimization is of great interest in the engineering community. For example, in the design of internal cooling ducts in turbomachineries, one possible objective is to minimize the pressure loss in a 180-degree return bend. However, reducing pressure loss may also reduce heat transfer, thus reduce the cooling ability [101, 102]. Therefore, a viable approach is to set a minimum heat transfer constraint while minimizing pressure loss. Another possible constraint in the internal cooling duct problem is the geometric bound on the duct's wall. This is because the duct must be contained inside the airfoil, and must be geometrically compatible with other components. We illustrate such a constraint in Fig 16.

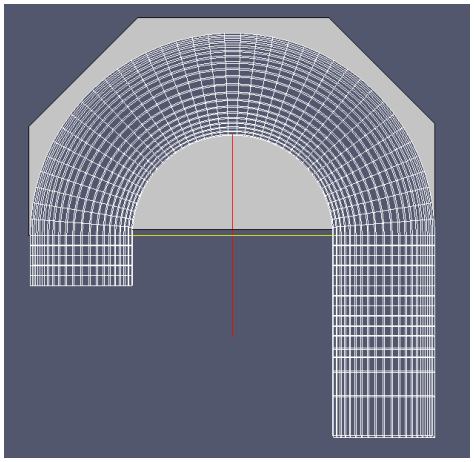

Figure 16: Example of a constraint of the feasible geometry, The brighter region indicates feasible region of the return bend geometry. The wireframe indicates a feasible design.

Constraints can be categorized into two types based on how they are evaluated: the first type are constraints that require a PDE simulations's space-time solution; the second type are constraints that do *not* require the space-time solution. In the example above, the constraint on the heat transfer belongs to the first type, while the constraint on the duct geometry belongs to the second type. Generally, evaluation of the first type constraint is much more expensive than the second type constraint, and can be similarly expensive as the evaluation of the objective function. Let the feasible region specified by the first type constraints be $\mathcal{C}_1$, and let the feasible region specified by the second type constraints be $\mathcal{C}_2$. The feasible region in Eqn(53) is $\mathcal{C} = \mathcal{C}_1 \bigcap \mathcal{C}_2$.

The second type constraints are not the topic of my research, because generally they can be evaluated cheaply. Optimization modules such as NLopt[73] have already implemented many established methods for such constraints. In Bayesian optimization, this type of constraints can be implemented in the optimization of the acquisition function, so the next evaluation point is always inside $\mathcal{C}_2$. The first type constraints, however, require special attention. If we choose the expected improvement formulation in our Bayesian optimization, Eqn(70) should be modified to:

$$c_{s+1} = \arg\max_{c \in \mathcal{C}_2} \left\{ \mathbb{E}\left[\max\left(J(c) - J(c_s^*), 0\right) \Big| \mathcal{S} \right] \cdot \mathbb{P}\big[V(c) \leq 0 \big| \mathcal{S}\big] \right\}, \tag{73}$$

The modified formulation is called *constraint expected improvement* [81]. Here c_s^* is the best *feasible* design in the previous s iterations. Notice the optimization domain is $c \in \mathcal{C}_2$. Intuitively, improvement on J is not counted valid unless the design is feasible.

Similar to $J(c)$, we can construct a posterior distribution on $V(c)$ and evaluate $\mathbb{P}\big[V(c) \leq 0 \big| \mathcal{S}\big]$ in Eqn(73). Straightforward evaluations of $V(c)$ using the primal model can be costly. However, similar to $J(c)$, we can use the twin model to estimate the gradient of $V(c)$ cheaply. Specifically, we construct the posterior of $V(c)$ using evaluations of $V(c)$ and $\frac{d\tilde{V}(c)}{dc}$, and use Eqn(73) as our acquisition function to perform optimization. The framework is shown in Fig 17.

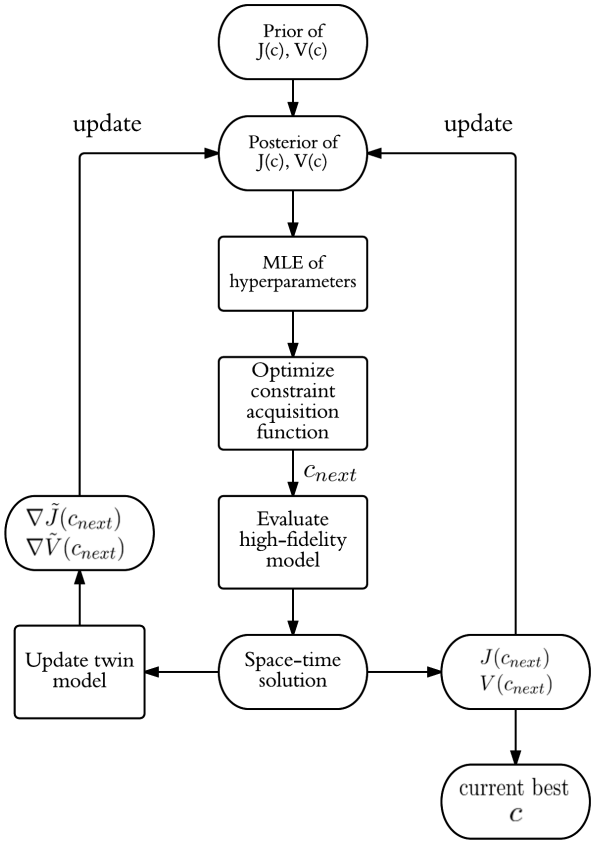

Figure 17: The flowchart of constraint Bayesian optimization with twin model.

5 Application to a turbulent flow optimization

5.1 Problem description

Gas turbine blades are designed to operate at high temperatures. Because materials employed for turbine components may not withstand such high termperatures, cooling should be applied to the blades for safety. One cooling method is to drill tubes inside the blades to circulate the coolant air, shown in Fig18. The pressure to drive the coolant flow is diverted from the compressor, thus resulting in a penalty to the turbine efficiency. An effective design should have minimal coolant flow rate and/or pressure drop penalty [101, 102]. Generally, the cooling tube layout looks like a stack of several *return bends*. Therefore, some researchers consider optimizing the design of a single return bend for simplicity. Optimization based on flow simulation have been employed to optimize the return bend geometry. Generally such flow is turbulent, since the Reynolds number can be tens of thousands (∼40,000 [102]). Also, the flow may contain several complex features, such as separation and reattachment, which are not captured well in the low-fidelity simulations. Although a high-fidelity PDE model can be used to simulate the flow, the computational cost can be prohibitively expensive when the design space is high.

Figure 18: Example of turbine blade cooling tubes. Image from: http://www.amateras-tyo.biz/eng/technologies.html

Currently, adjoint method has not been implemented for high-fidelity PDE simulations like DNS and LES. So efficient gradient evaluation of the high-fidelity PDE models are not available. However, there are a set of low-fidelity models like RANS. RANS can be evaluated more cheaply; and adjoint can be implemented on RANS easily. Because the high-fidelity simulations can provide space-time solutions of all relevant flow quantities (velocity, pressure, etc), we consider training RANS models using the time averged space-time solution of the high-fidelity simulation. Therefore, we will consider the gray-box high-fidelity simulations as the primal model, and consider the trained RANS model the twin model. Generally, the Mach number in such flows are low (∼0.05 [101]), so we may focus on incompressible models. Besides, we will model the flow as two dimensional for simplicity.

5.2 Review of incompressible RANS models

RANS are time averaged equations of motion for fluid flow. The idea behind the equation is Reynolds decomposition, where the flow instantaneous quantities are decomposed into its time-averaged and fluctuating quantities. For a stationary incompressible Newtonian fluid, the equations are

$$\begin{aligned} \nabla \cdot \bar{\mathbf{u}} &= 0 \\ \rho \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} &= -\nabla \bar{p} + \mu \nabla^2 \bar{\mathbf{u}} - \rho \nabla \cdot \left(\overline{\mathbf{u}' \mathbf{u}'} \right) . \end{aligned} \tag{74}$$

Here the instantaneous velocity $\mathbf{u} = (u, v)^T$ is decomposed as $\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}'$. $\bar{\mathbf{u}}$ is the time-averaged velocity and $\mathbf{u}'$ is the fluctuating velocity. Similarly $p = \bar{p} + p'$. ρ is the constant density, μ is the dynamic

viscosity. The term

$$\mathbf{R} = \rho\overline{\mathbf{u}'\mathbf{u}'} \tag{75}$$

is called *Reynolds stress*.

Modelling the Reynolds stress is recognized as a closure problem, and is a subject of intense interest for almost a century. There are many models for Reynolds stress, in which a popular class of models is based on the *Boussinesq hypothesis* [103]. Boussinesq hypothesis models the Reynolds stress by a linear constitutive relationship with the mean flow straining field:

$$\mathbf{R} = -\mu_t \left(\nabla\bar{\mathbf{u}} + \nabla\bar{\mathbf{u}}^T\right) + \frac{1}{3}\rho\bar{\mathbf{u}}\cdot\bar{\mathbf{u}}\boldsymbol{I} \tag{76}$$

μ_t is called *eddy viscosity*. $\nu_t = \mu_t/\rho$ is called *kinematic eddy viscosity*. The only term that needs to be modelled for closure is μ_t (or ν_t).

Depending on how μ_t is modelled, most RANS models based on Boussinesq hypothesis can be categorized in to three classes: *zero equation models*, *one equation models*, and *two equation models*. Zero equation models do not require the solution of additional equations. μ_t is calculated directly from *local* flow variables. Therefore, zero equation models may not be able to capture the phenomenons like convections and diffusions of turbulent energy. For example, a classical zero equation model is the Prandtl's mixing length model [103]:

$$\mu_t = \rho\left|\frac{\partial u}{\partial y}\right| l_m^2\,, \tag{77}$$

where $\frac{\partial u}{\partial y}$ is the partial derivative of the streamwise velocity with respect to the wall normal direction. One equation models require the solution of one additional equation. Generally the additional equation is a transport equation of a quantity related to the eddy viscosity. For example, the Baldwin-Barth model [103] decomposes μ_t as

$$\mu_t = C_\mu \mu\tilde{R}_T D_1 D_2\,, \tag{78}$$

where C_μ is a constant, D_1, D_2 are spatial-dependent variables, $\mu\tilde{R}_T$ is solved by a transport equation. Two equation models require the solution of two additional equations. An example is the *k-epsilon models* [103]. k-epsilon models assume the eddy viscosity to be

$$\mu_t = C_\mu \frac{k^2}{\epsilon}\,, \tag{79}$$

where

$$k = \frac{\bar{u'}^2 + \bar{v'}^2}{2} \tag{80}$$

is the *turbulent kinetic energy*,

$$\epsilon = \nu\overline{(\nabla\mathbf{u}') : (\nabla\mathbf{u}')^T} \tag{81}$$

is the *turbulent dissipation rate*, and C_μ is a constant. k-epsilon models solve two transport equations for k and ϵ. Generally, two-equation models are better at capturing some turbulent flow features such as convection and diffusion of turbulent energy, detachment, and reattachment [103].

5.3 Future work

We will optimize the return bend geometry to minimize the pressure loss using the proposed optimization framework. We may parameterize the return bend geometry by a set of *design nodes* on the inner and outter boundaries, and use a spline interpolation to generate the boundaries. The dimension of the design space will be the number of design nodes. We will use tens of design nodes, which yields a challenging high-dimensional optimization problem. There has been researches on this topic both numerically and experimentally [101, 102]. These researches can benchmark our results. However, instead of optimizing for the low-fidelity model's optimal which was the practice of most previous researches, we

aim at optimizing for the high-fidelity model's optimal.

We will implement an LES model in OpenFoam to serve as the high-fidelity gray-box simulation; and implement a one-equation or two-equation RANS model in python to serve as the low-fidelity twin model. We will implement adjoint in the RANS model, using an automatic differentiation toolbox. In the RANS model, the equation(s) of the eddy viscosity will be adaptive. Further investigation is required to find appropriate forms of the adaptive eddy viscosity equation, and to adaptively select the basis for the eddy viscosity equation(s). After that, we will train the adaptive RANS model with the high-fidelity simulation's time-averaged space-time solutions. Then we can use the proposed Bayesian optimization framework to perform optimization. In the end, we will survey existing designs in literatures, and compare them with our optimal design, using the LES simulation.

6 Appendices

6.1 Proof of theorem 1: global-local error

Proof: Consider a primal model and a twin model solving for the space-time solution u and $\tilde{u}$ on the same time grid $\{t_0 = 0, t_1, \cdots, t_n = T\}$. The solutions at timestep t_i are u_i and $\tilde{u}_i$. Both simulations starts from the same initial condition u_0. Define the mapping of the primal model

$$H : \mathbb{R}^n \mapsto \mathbb{R}^n,\ u^i \to Hu^i = u^{i+1}\,, \quad i = 1, \cdots, n \tag{82}$$

and the mapping of the twin model

$$G : \mathbb{R}^n \mapsto \mathbb{R}^n,\ \tilde{u}^i \to G\tilde{u}^i = \tilde{u}^{i+1}\,, \quad i = 1, \cdots, n \tag{83}$$

G and H are linear or nonlinear mappings. We illustrate these mappings in Fig 19.

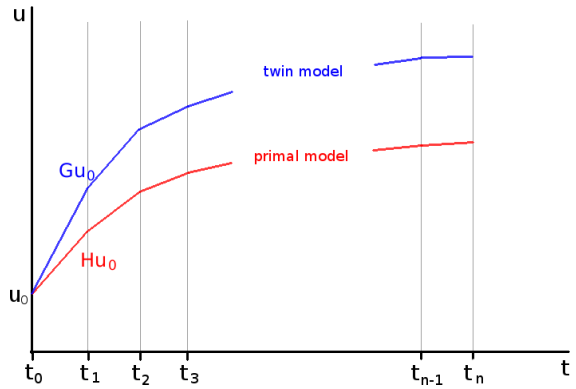

Figure 19: Trajectories of the primal model and the twin model. H and G are their timestep-wise mapping functions.

The global error is

$$Err_G = \frac{1}{n\Delta t}\left\{\|(G^n - H^n)u_0\| + \|(G^{n-1} - H^{n-1})u_0\| + \cdots + \|(G^1 - H^1)u_0\|\right\} \tag{84}$$

The local error is

$$Err_L = \frac{1}{n\Delta t}\left\{\|\left(GH^{n-1} - H^n\right)u_0\| + \|\left(GH^{n-2} - H^{n-1}\right)u_0\| + \cdots + \|\left(G^1 - H^1\right)u_0\|\right\} \tag{85}$$

Using the equality

$$G^i - H^i = (G^i - G^{i-1}H) + (G^{i-1}H - G^{i-2}H^2) + \cdots + (GH^{i-1} - H^i)\,, \quad i \in \mathbb{Z}^+\,, \tag{86}$$

we derive from Eqn(84)

$$Err_G \le \frac{1}{n\Delta t}\left\{\begin{array}{lll} \|(G^{n-1}G - G^{n-1}H)u_0\| + \|(G^{n-2}GH - G^{n-2}H^2)u_0\| & +\cdots+ \|(GH^{n-1} - H^n)u_0\| \\ + \|(G^{n-2}G - G^{n-2}H)u_0\| & +\cdots+ \|(GH^{n-2} - H^{n-1})u_0\| \\ \ddots & \vdots \\ & + \|(G - H)u_0\| \end{array}\right\} \tag{87}$$

Using Eqn(87) and Eqn(85), we get

$$Err_G - Err_L \le \frac{1}{n\Delta t}\left\{\begin{array}{lll} \|(G^{n-1}G - G^{n-1}H)u_0\| + \|(G^{n-2}GH - G^{n-2}H^2)u_0\| & +\cdots+ \|(GGH^{n-2} - GH^{n-1})u_0\| \\ + \|(G^{n-2}G - G^{n-2}H)u_0\| & +\cdots+ \|(GGH^{n-3} - GH^{n-2})u_0\| \\ \ddots & \vdots \\ & + \|(GG - GH)u_0\| \end{array}\right\} \tag{88}$$

Apply our assumption

$$\|Gx - Gy\| \leq \alpha \|x - y\| \tag{89}$$

and its implication

$$\left\|G^i x - G^i y\right\| \leq \alpha^i \|x - y\|\,, \quad i \in \mathbb{Z}^+ \tag{90}$$

to Eqn(91), we get

$$Err_G - Err_L \leq \frac{1}{n\Delta t} \left\{ \begin{array}{lll} \alpha^{n-1}\|(G-H)u_0\| + \alpha^{n-2}\|(GH-H^2)u_0\| & +\cdots+\alpha\|(GH^{n-2}-H^{n-1})u_0\| \\ \qquad\qquad + \alpha^{n-2}\|(G-H)u_0\| & +\cdots+\alpha\|(GH^{n-3}-H^{n-2})u_0\| \\ \qquad\qquad \ddots & \vdots \\ & \qquad + \alpha\|(G-H)u_0\| \end{array} \right\} \tag{91}$$

Reorder the summation, we get

$$Err_G - Err_L \leq \frac{1}{n\Delta t} \left\{ \begin{array}{lll} \alpha^{n-1}\|(G-H)u_0\| + \alpha^{n-2}\|(G-H)u_0\| & +\cdots+\alpha\|(G-H)u_0\| \\ \qquad\qquad + \alpha^{n-2}\|(GH-H^2)u_0\| & +\cdots+\alpha\|(GH-H^2)u_0\| \\ \qquad\qquad \ddots & \vdots \\ & \qquad + \alpha\|(GH^{n-2}-H^{n-1})u_0\| \end{array} \right\} \tag{92}$$

Therefore, from Eqn(92) and Eqn(85), we get

$$Err_G \leq (1 + \alpha + \cdots + \alpha^{n-1}) Err_L\,. \tag{93}$$

If $|\alpha|$ is strictly less than 1,

$$Err_G \leq \frac{1}{1-\alpha} Err_L\,, \tag{94}$$

which completes the proof. □

7 References

[1] Han Chen. "Blackbox stencil interpolation method for model reduction" Master thesis, 2012

[2] Chen, Han, et al. "Conditional sampling and experiment design for quantifying manufacturing error of transonic airfoil." Proceedings of the 49th Aerospace Sciences Meeting. 2011.

[3] Constantine, Paul G., Eric Dow, and Qiqi Wang. "Active subspace methods in theory and practice: Applications to kriging surfaces." SIAM Journal on Scientific Computing 36.4 (2014): A1500-A1524.

[4] Willcox, Karen, and Jaime Peraire. "Balanced model reduction via the proper orthogonal decomposition." AIAA journal 40.11 (2002): 2323-2330.

[5] March, Andrew, Karen Willcox, and Qiqi Wang. "Gradient-based multifidelity optimisation for aircraft design using Bayesian model calibration." Aeronautical Journal 115.1174 (2011): 729.

[6] Jones, Donald R., Matthias Schonlau, and William J. Welch. "Efficient global optimization of expensive black-box functions." Journal of Global optimization 13.4 (1998): 455-492.

[7] Bull, Adam D. "Convergence rates of efficient global optimization algorithms." The Journal of Machine Learning Research 12 (2011): 2879-2904.

[8] Stephen A Billings. "Nonlinear System Identification, NARMAX methods in time, frequency and spatial-temporal domains" ISBN:978-1-119-94359-4, 2013

[9] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian optimization of machine learning algorithms." Advances in Neural Information Processing Systems. 2012.

[10] Kennedy, Marc C., and Anthony O'Hagan. "Predicting the output from a complex computer code when fast approximations are available." Biometrika 87.1 (2000): 1-13.

[11] A. O'Hagan "A Markov property for covariance structures." Nottingham University Statistics Research Report 13 (1998).

[12] Murray, Iain, and Ryan P. Adams. "Slice sampling covariance hyperparameters of latent Gaussian models." Advances in Neural Information Processing Systems. 2010.

[13] Oliver, Margaret A., and R. Webster. "Kriging: a method of interpolation for geographical information systems." International Journal of Geographical Information System 4.3 (1990): 313-332.

[14] Carter, Richard G. "Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information." SIAM Journal on Scientific Computing 14.2 (1993): 368-388.

[15] Dembo, Ron S., Stanley C. Eisenstat, and Trond Steihaug. "Inexact newton methods." SIAM Journal on Numerical analysis 19.2 (1982): 400-408.

[16] Conn, Andrew R., Katya Scheinberg, and Lus N. Vicente. "Global convergence of general derivative-free trust-region algorithms to first-and second-order critical points." SIAM Journal on Optimization 20.1 (2009): 387-415.

[17] Wild, Stefan M., and Christine Shoemaker. "Global convergence of radial basis function trust-region algorithms for derivative-free optimization." SIAM Review 55.2 (2013): 349-371.

[18] G.M. Matheron, "Principles of geostatistics". Economic Geology 58.8 (1963): 1246-1266

[19] Goovaerts, Pierre. "Ordinary cokriging revisited." Mathematical Geology 30.1 (1998): 21-42.

[20] Christopher M. Bishop Pattern recognition and machine learning ISBN: 978-0387310732. 2007

[21] Sarma, Pallav, LJ Durlofsky, K Aziz, WH Chen. "Efficient real-time reservoir management using adjoint-based optimal control and model updating." Computational Geosciences 10.1 (2006): 3-36.

[22] Rios, Luis Miguel, and Nikolaos V. Sahinidis. "Derivative-free optimization: A review of algorithms and comparison of software implementations." Journal of Global Optimization 56.3 (2013): 1247-1293.

[23] Powell, Warren B. "Approximate Dynamic Programming: Solving the curses of dimensionality". Vol. 703. John Wiley & Sons, 2007.

[24] Dennis, Jr, John E., and Jorge J. Mor. "Quasi-Newton methods, motivation and theory." SIAM review 19.1 (1977): 46-89.

[25] Plessix, R-E. "A review of the adjoint-state method for computing the gradient of a functional with geophysical applications." Geophysical Journal International 167.2 (2006): 495-503.

[26] Nadarajah, Siva, and Antony Jameson. "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization." AIAA paper 667 (2000): 2000.

[27] A Griewank, GF Corliss Automatic differentiation of algorithms: theory, implementation, and application. Defense Technical Information Center, 1992.

[28] Zhangxin Chen "Reservoir Simulation: Mathematical Techniques in Oil Recovery" Society for Industrial and Applied Mathematics, ISBN 0898716403, 2007

[29] Mallat, Stephane G. "A theory for multiresolution signal decomposition: the wavelet representation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 11.7 (1989): 674-693.

[30] Vazquez, Emmanuel, and Julien Bect. "Convergence properties of the expected improvement algorithm with fixed mean and covariance functions." Journal of Statistical Planning and inference 140.11 (2010): 3088-3095.

[31] S.E. Buckley and M.C. Leverett "Mechanism of fluid displacement in sands." Transactions of the AIME 146 (1942): 107-116

[32] Chen, Zhangxin, Guanren Huan, and Yuanle Ma. "Computational methods for multiphase flows in porous media." Vol. 2. Siam, 2006.

[33] Stephen Boyd and Lieven Vandenberghe "Convex Optimization" Cambridge University Press, 2004

[34] G. Cybenko "Approximation by superpositions of a sigmoid function" Mathematics of control, signals and systems 2.4 (1989): 303-314

[35] Alfred Haar "On the theory of orthogonal function systems" Mathematische Annalen 69 (1910): 331-371

[36] V.VV. Vermehren, H.M. de Oliveira "Close expressions for Meyer wavelet and scale function" arXiv:1502.00161 [stat.ME]

[37] Slawomir Koziel, Xin-She Yang "Computational optimization, methods and algorithms" Springer Berlin Heidelberg, 2011

[38] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidynathan, P.K Tucker "Surrogate-based analysis and optimization" Progress in Aerospace Sciences 41 (2005): 1-28

[39] T.D. Robinson, M.S. Eldred, K.E. Willcox, R. Haimes "Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping" AIAA Journal 46 (2008): 2814-2822

[40] Mohamed H. Bakr, John W. Bandler "An introduction to the space mapping technique" Optimization and Engineering 2 (2011): 369-384

[41] N.M. Alexandrov, E.J. Nielsen, R.M. Lewis, W.K. Anderson "First-Order Model Management with Variable-Fidelity Physics Applied to Multi-Element Airfoil Optimization" 8th AIAA Symposium on Multidisciplinary Design and Optimization (2000)

[42] N.M. Alexandrov, R.M. Lewis, C.R. Gumbert, L.L. Green, P.A. Newmann "Optimization with Variable-Fidelity Models Applied to Wing Design" 38th Aerospace Sciences Meeting (2000)

[43] Han Zhong-Hua, Stefan Grtz, Ralf Zimmermann "Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function." Aerospace Science and Technology 25.1 (2013): 177-189.

[44] Gary G. Wang, S. Shan "Review of metamodeling techniques in support of engineering design optimization." Journal of Mechanical Design 129.4 (2007): 370-380.

[45] Shinkyu Jeong, Mitsuhiro Murayama, Kazuomi Yamamoto "Efficient optimization design method using kriging model" Journal of aircraft 42.2 (2005): 413-420.

[46] Nestor V. Queipo, Javier V. Goicochea, Salvador Pintos. "Surrogate modeling-based optimization of SAGD processes." Journal of Petroleum Science and Engineering 35.1 (2002): 83-93.

[47] Hyoung-Seog Chung, Juan J. Alonso. "Using gradients to construct cokriging approximation models for high-dimensional design optimization problems." AIAA paper 317 (2002): 14-17.

[48] Songqing Shan, G. Gary Wang. "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions." Structural and Multidisciplinary Optimization 41.2 (2010): 219-241.

[49] Jonas Sjberg et al. "Nonlinear black-box modeling in system identification: a unified overview." Automatica 31.12 (1995): 1691-1724.

[50] Laurens JP van der Maaten, Eric O. Postma, H. Jaap van den Herik "Dimensionality reduction: A comparative review." Journal of Machine Learning Research 10.1-41 (2009): 66-71.

[51] T.R. Browning "Applying the design structure matrix to system decomposition and integration problems: a review and new directions" IEEE Trans Eng Manage 48.3 (2001): 292-306

[52] Raymond H. Myers, Douglas C. Montgomery, Christine M. Anderson-Cook "Response surface methodology: process and product optimization using designed experiments" Vol. 705. John Wiley and Sons, 2009

[53] Ira H. Abbott, E. Albert Von Doenhoff "Theory of wing sections" Dover Publications Inc., Section 4.2 (1959)

[54] Tsan-Hsing Shih, et al. "A new k- eddy viscosity model for high reynolds number turbulent flows" Computers and Fluids 24.3 (1995): 227-238.

[55] Virendra C. Patel, Wolfgang Rodi, and Georg Scheuerer "Turbulence models for near-wall and low Reynolds number flows-a review" AIAA journal 23.9 (1985): 1308-1319

[56] Toby S. Cubitt, Jens Eisert, Michael M. Wolf "Extracting dynamical equations from experimental data is NP hard" Physical review letters 108.12 (2012): 120503

[57] Rick Salmon "Hamiltonian fluid mechanics" Annual review of fluid mechanics 20.1 (1988): 225-256

[58] Randall J. LaVeque "Finite volume methods for hyperbolic problems" Vol. 31. Cambridge university press, 2002

[59] Pieter Eykhoff "System identification, parameter and system estimation" John Wiley and Sons, 1974

[60] S.A. Billings, W.S.F Voon "Piecewise linear identification of non-linear system" International Journal of Control, 46.1 (1987): 215-235

[61] Georgios B. Giannakis, Erchin Serpedin "A bibliography on nonlinear system identification" Signal Processing 81.3 (2001): 533-580

[62] M.J. Korenberg, I.W. Hunter "The Identification of Nonlinear Biological Systems: Volterra Kernel Approaches" Annals Biomedical Engineering 24.2 (1996): 250-268

[63] Julian Jakob Bussgang "Crosscorrelation functions of amplitude-distorted Gaussian signals" (1952)

[64] C.P. Kwong, C. F. Chen "Linear feedback system identification via block-pulse functions" International Journal of Systems Science 12.5 (1981): 635-642

[65] S.A. Billings, I.J. Leontaritis "Identification of nonlinear systems using parametric estimation techniques" Proceedings of the IEE Conference on Control and its Application, Warwick, UK, pp.183-187

[66] Sheng Chen, S. A. Billings, P. M. Grant "Non-linear system identification using neural networks" International journal of control 51.6 (1990): 1191-1214

[67] Stephen A. Billings, Hua-Liang Wei "A new class of wavelet networks for nonlinear system identification." Neural Networks, IEEE Transactions on 16.4 (2005): 862-874.

[68] S. A. Billings, W. S. F. Voon "Correlation based model validity tests for non-linear models" International Journal of Control 44.1 (1986): 235-244.

[69] Tammo Jan. Dijkema "Adaptive tensor product wavelet methods for solving PDEs" PhD thesis, Utrecht University (2009)

[70] Ismail Kucuk, Ibrahim Sadek "An efficient computational method for the optimal control problem for the Burgers equation." Mathematical and computer modelling 44.11 (2006): 973-982.

[71] Qiqi Wang, Numpad package, https://github.com/qiqi/numpad.git

[72] Scikit-learn package, https://github.com/scikit-learn/scikit-learn.git

[73] Steven G. Johnson, The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt

[74] John E. Dennis, Jorge J. Mor. "Quasi-Newton methods, motivation and theory." SIAM review 19.1 (1977): 46-89.

[75] Eric Alexander. Dow, "Quantification of structural uncertainties in RANS turbulence models." Dissertation, Massachusetts Institute of Technology, 2011.

[76] J. Nocedal. "Updating quasi-Newton matrices with limited storage" Mathematics of Computation, 35 (1980): 773-782

[77] Van der Maaten, Laurens JP, Eric O. Postma, and H. Jaap van den Herik. "Dimensionality reduction: A comparative review." Journal of Machine Learning Research 10.1-41 (2009): 66-71.

[78] Havi, Ron, and George H. John. "Wrappers for feature subset selection." Artificial intelligence 97.1 (1997): 273-324.

[79] Wei, Hua-Liang, and Stephen A. Billings. "Feature subset selection and ranking for data dimensionality reduction." Pattern Analysis and Machine Intelligence, IEEE Transactions on 29.1 (2007): 162-166.

[80] Jolliffe, Ian. Principal component analysis. John Wiley and Sons, Ltd, 2002.

[81] Gardner, Jacob, et al. "Bayesian optimization with inequality constraints." Proceedings of The 31st International Conference on Machine Learning. 2014.

[82] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society. Series B (Methodological) (1996): 267-288.

[83] Dziak, John, Runze Li, and Linda Collins. "Critical review and comparison of variable selection procedures for linear regression (Technical report)." (2005).

[84] Derksen, Shelley, and H. J. Keselman. "Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables." British Journal of Mathematical and Statistical Psychology 45.2 (1992): 265-282.

[85] Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67.2 (2005): 301-320.

[86] Stone, Mervyn. "An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion." Journal of the Royal Statistical Society. Series B (Methodological) (1977): 44-47.

[87] Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics 6.2 (1978): 461-464.

[88] J Mockus, V Tiesis, and A Zilinskas "The application of Bayesian methods for seeking the extreme." Towards Global Optimization, 2 (1978): 117-129

[89] Choi, Seongim, Juan J. Alonso, and Ilan M. Kroo. "Two-level multifidelity design optimization studies for supersonic jets." Journal of Aircraft 46.3 (2009): 776-790.

[90]

[91] Robinson, T. D., et al. "Multifidelity optimization for variablecomplexity design." Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA. 2006.

[92] Torczon, Virginia. "On the convergence of pattern search algorithms." SIAM Journal on optimization 7.1 (1997): 1-25.

[93] Booker, Andrew J., et al. "A rigorous framework for optimization of expensive functions by surrogates." Structural optimization 17.1 (1999): 1-13.

[94] Andrew I. March "Multifidelity methods for multidisciplinary system design" Dissertation, Massachusetts Institute of Technology (2012)

[95] Aronszajn, Nachman. "Theory of reproducing kernels." Transactions of the American mathematical society (1950): 337-404.

[96] Vogel, Curtis R. "Computational methods for inverse problems." Vol. 23. Siam, 2002.

[97] Meneveau, Charles, and P. Sagaut. Large eddy simulation for incompressible flows: an introduction. Springer Science and Business Media, 2006.

[98] Smagorinsky, Joseph. "General circulation experiments with the primitive equations: I. the basic experiment." Monthly weather review 91.3 (1963): 99-164.

[99] Moin, Parviz, and Krishnan Mahesh. "Direct numerical simulation: a tool in turbulence research." Annual review of fluid mechanics 30.1 (1998): 539-578.

[100] Li, Wu, Luc Hyuse, and Sharon Padula. "Robust airfoil optimization to achieve consistent drag reduction over a Mach range." No. ICASE-TR-2001-22. INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING HAMPTON VA, 2001.

[101] Verstraete, Tom, et al. "Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling ChannelsPart I: Numerical Method." Journal of Turbomachinery 135.5 (2013): 051015.

[102] Coletti, Filippo, et al. "Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling ChannelsPart II: Experimental Validation." Journal of Turbomachinery 135.5 (2013): 051016.

[103] Wilcox, David C. "Turbulence modeling for CFD." Vol. 2. La Canada, CA: DCW industries, (1998)

[104] Wang, Qiqi. "Forward and adjoint sensitivity computation of chaotic dynamical systems." Journal of Computational Physics 235 (2013): 1-13.

[105] Talnikar, C., et al. "Parallel Optimization for LES." Proceedings of the Summer Program. 2014.

[106] Aimo Torn, Antanas Zilinskas "Global Optimization" Springer-Verlag New York, Inc. New York, NY, (1989)