THESIS PROPOSAL

JORDAN R. HALL

Abstract. Obtaining the solution to stochastic inverse problems is an important challenge in computational science. By solving a stochastic inverse problem, one may find an updated characterization of parameter space that combines prior beliefs and observable data. Using a new approach detailed in [Butler], we may solve for a probability distribution in parameter space, Λ, such that its push-forward (*i.e.,* image) through the parameter-to-data map f matches the observed probability density on the data, $\mathcal{D}$. Under certain conditions [**?**, **?**], an approximate solution of the stochastic inverse problem can be obtained as the solution of a regularized, convex minimization problem.

In this proposal, we consider a noisy map f from a high-dimensional Λ to $\mathcal{D}$ of specified dimension, where the gradient of f exists, but may be inaccessible. If the gradient of f is accessible, the minimization problem can be solved using gradient-based descent methods. If the gradient of f is inaccessible, one may apply Derivative-Free Optimization (DFO) schemes as in [**C/W**]. In either case, the effectiveness of the optimization algorithm may be enhanced by performing dimension reduction in Λ as in [**?**, **?**]. We consider different strategies for computing the active subspace of $f(\Lambda)$ in the context of both approaches, and apply them to the data-consistent inversion of two model problems. Both of these model problems are related to stochastic inverse problems arising in the simulation of plasmas in fusion reactors.

Contents

Literature Review and Framework

In the proceeding section we provide a literature review of inverse problem theory, derivative-free (DF) optimization, and dimension reduction. In the process, we will build a theoretical framework upon which to pose research questions, state initial results, and form a research plan.

We begin our discussion by defining a parameter space Λ of dimension N, a map or "model" f, and a data space $\mathcal{D}$. We assume that $\mathcal{D}$ has dimension M, typically with $M \leq N$. Points in $\mathcal{D}$ may be known values of $f(\lambda), \lambda \in \Lambda$; as such, we may write $d = f(\lambda)$ to denote the particular datum corresponding to the evaluation of a point $\lambda \in \Lambda$ under f. Points in $\mathcal{D}$ may also be *observed* data, denoted d_{obs}, where the corresponding $\lambda \in \Lambda$ may be unknown. We allow for realizations of f to be noisy. Hence, we may model draws of f with $f(\lambda) = f(\lambda) + \epsilon$, which is an additive noise structure, or $f(\lambda) = f(\lambda)(1 + \epsilon)$, which is a multiplicative noise structure.

Data-Consistent Inversion We follow [**?**, **?**, **?**] to formulate an inverse problem in our setting. Then, we briefly present the data-consistent solution; we refer to [**?**] for details.

First, we assume we have prior knowledge of the parameter space, given by the *prior distribution*, $\pi_\Lambda^{\text{prior}}(\lambda)$. In practical applications, a prior distribution may be given by application experts, but may also reflect the state of knowledge on Λ obtained by other mathematical or statistical processes. Second, we assume that we have an *observed density*, $\pi_{\mathcal{D}}(d)$, which represents our uncertain state of knowledge of observed data and is uncertain due to noise in f and potential measurement error. Informally, the inverse problem is the task of finding an *updated* probability distribution, $\pi_\Lambda^{\text{update}}$, in Λ space that combines the given prior information and data.

It has been shown [**?**] that the so-called classical Bayesian or statistical Bayesian solution to the inverse problem is not generally a "pull-back probability measure," in the sense that the image of the updated distribution $\pi_\Lambda^{\text{update}}$ under the map f, called the push-forward, is not equal to (and often not "close" to) the "observed" probability distribution, $\pi_{\mathcal{D}}$.

Given a distribution on $\lambda \in \Lambda$, the *forward problem* is finding the probability distribution of $f(\lambda)$. The forward problem is, in its own right, a nontrivial and important problem in uncertainty quantification. As it turns out, performing DCI will require solving a forward problem.

Data-consistent inversion (DCI) seeks an updated solution $\pi_\Lambda^{\text{update}}$ for which the push-forward exactly equals $\pi_{\mathcal{D}}$. To obtain such a solution, it is necessary to compute the push-forward of $\pi_\Lambda^{\text{prior}}$. We may form the push-forward of the prior by solving the corresponding forward problem; with certain assumptions on f, Λ, and $\mathcal{D}$ the forward problem may be straightforward, with other assumptions, this problem is non-trivial. We denote the solution to the forward problem with $\pi_{\mathcal{D}}^{f(\Lambda)}$. Then, the *data-consistent* solution to the inverse problem is

$$\pi_\Lambda^{\text{update}} = \pi_\Lambda^{\text{prior}}(\lambda) \frac{\pi_{\mathcal{D}}(f(\lambda))}{\pi_{\mathcal{D}}^{f(\Lambda)}(f(\lambda))}. \tag{1}$$

Optimization Methods for Solving Inverse Problems With our assumptions of an expensive f and a high-dimensional Λ, approximately solving the forward problem to obtain the push-forward density generally requires some type of kernel density estimation (KDE),

which converges slowly, typically at Monte Carlo rates of $\frac{1}{\sqrt{N}}$. As in [**?**], depending on f, the solution to the inverse problem can be obtained exactly or approximately by solving an equivalent deterministic convex optimization problem. We may alter the classical formulation of the deterministic optimization problem to ensure we obtain, depending on f, the *data-consistent* (exact or approximate) solution to the inverse problem.

We begin by examining the classical formulation of a deterministic optimization problem which corresponds to either exactly or approximately solving the stochastic inverse problem (SIP) under consideration. In particular, for a linear map f, the exact solution is obtained; for nonlinear maps, we obtain an approximate solution. As in [**?**], given observed data $d_{\text{obs}} \sim \pi_{\mathcal{D}}$ and a draw $\lambda_{\text{prior}} \sim \pi_{\Lambda}^{\text{prior}}$, we define a *misfit function*

$$S(\lambda) = \frac{1}{2}\left(||f(\lambda) - d_{\text{obs}}||_{\mathcal{D}}^2 + ||\lambda - \lambda_{\text{prior}}||_{\Lambda}^2\right). \tag{2}$$

The misfit function appears in the classical (i.e., statistical Bayesian) solution to the inverse problem, as $\pi_{\Lambda}^{\text{update}}(\lambda) \sim \exp(-S(\lambda))$. We observe that finding the MAP point is equivalent to finding the λ for which S is minimized.

As in the preceding section, we assume $\pi_{\Lambda}^{\text{prior}}$ is a known probability density with mean $\bar{\lambda}$ and covariance matrix C_{Λ}; likewise, $\pi_{\mathcal{D}}$ is a known probability density with mean $\bar{d}$ and covariance matrix $C_{\mathcal{D}}$. As in [**?**], we can use the assumed covariance structures on Λ and $\mathcal{D}$ to rewrite (2) as

$$S(\lambda) = \frac{1}{2}\left(\left|\left|C_{\mathcal{D}}^{-1/2}(f(\lambda) - d_{\text{obs}})\right|\right|_2^2 + \left|\left|C_{\Lambda}^{-1/2}(\lambda - \lambda_{\text{prior}})\right|\right|_2^2\right). \tag{3}$$

Classically [**?**], it is assumed that $\pi_{\Lambda}^{\text{prior}}$ and $\pi_{\mathcal{D}}$ are Gaussians, and the misfit function is specified by $\lambda_{\text{prior}} = \bar{\lambda}$ and $d_{\text{obs}} = \bar{d}$ in our notations. In (2) and (3), we choose to define a more general misfit function, so that we may consider the "misfit" given any draws $\lambda_{\text{prior}} \sim \pi_{\Lambda}^{\text{prior}}$ and $d_{\text{obs}} \sim \pi_{\mathcal{D}}$. The misfit function defined above can be understood term-by-term. Given $\lambda \in \Lambda$, the term $||f(\lambda) - d_{\text{obs}}||_{\mathcal{D}}^2$ corresponds to finding the mismatch between the observed d_{obs} and $f(\lambda)$ in data space; the term $||\lambda - \lambda_{\text{prior}}||_{\Lambda}^2$ corresponds to performing Tikhonov regularization, which improves conditioning in minimizing S over Λ. The minimum of S is typically referred to as the maximum a posteriori point or *MAP point*.

In [**?**], S is rewritten to obtain a data-consistent MAP point. An additional "deregularization" term is appended so that if a unique solution exists, the regularization will be "turned off." For now, we assume that f is linear (or can be linearized locally [**?**]), and write the matrix A to define that action of f on Λ. We also assume that π_{Λ} and $\pi_{\mathcal{D}}$ are Gaussian. In this case, we write the *data-consistent misfit function,*

$$T(\lambda) = \frac{1}{2}\left(\left|\left|C_{\mathcal{D}}^{-1/2}(f(\lambda) - d_{\text{obs}})\right|\right|_2^2 + \left|\left|C_{\Lambda}^{-1/2}(\lambda - \lambda_{\text{prior}})\right|\right|_2^2 - \left|\left|C_{A}^{-1/2}(f(\lambda) - f(\lambda_{\text{prior}}))\right|\right|_2^2\right), \tag{4}$$

where $C_A = AC_{\Lambda}A^T$. We note that in the case that π_{Λ} and $\pi_{\mathcal{D}}$ are Gaussian and f is linear, the data-consistent solution to our inverse problem is given exactly by $\pi_{\Lambda}^{\text{update}}(\lambda) \sim \exp(-T(\lambda))$. Generally, the deregularization term will ensure a solution that updates the distribution on Λ only in the directions in which the data is informative. The linear case illustrates the action of the un-regularization term. In the case that a unique solution to the

inverse problem exists for a linear f (i.e., $N = M$ and A is full rank), the un-regularization term will equal the Tikhonov regularization term, so that the MAP point solves the data misfit exactly. When the problem is under-determined ($N > M$), then with a linear f, the data-consistent solution to inverse problem will lie in a hyperplane in Λ containing all the possible points that could have produced the M observed data. Regardless of whether $N = M$ or $N > M$, the MAP point obtained by minimizing the "classical" misfit, in general, will not be a point that could have produced the observed data. We highlight the difference in solutions in the example below.

Example 1. *Let $f(\lambda) = 2\lambda$, $\lambda_{prior} = 0.1$, $C_\Lambda = [0.5]$, $d_{obs} = 0.25$, and $C_\mathcal{D} = [0.25]$; note, $N = M = 1$. Then we find that $S(\lambda) = 2((2\lambda - 0.25)^2 + (\lambda - 0.1)^2)$, which is minimized by $\lambda_S^* = 3/25$. Assuming that λ_{prior} and d_{obs} are the means of Gaussians with standard deviations corresponding to the above covariance matrices, we have $\pi_\Lambda^{post} \sim \exp(-S(\lambda_S^*))$. Notice $f(\lambda_S^*) = 6/25 \neq d_{obs}$. We write $T(\lambda) = 2(2\lambda - 0.25)^2$, which is minimized by $\lambda_T^* = 1/8$. Notice $f(\lambda_T^*) = d_{obs}$. With Gaussian assumptions on the prior and data, we have $\pi_\Lambda^{post} \sim \exp(-T(\lambda_T^*))$.*

In the preceding example, we observe for linear f and $N = M$, the classical MAP point, given by $\lambda_S^* = 0.12$ and obtained by minimizing S, strikes a balance between the prior belief that $\lambda = 0.1$ and the fact that the value $\lambda = 0.125$ is the pre-image of 0.25 under f, the observed data point. The data-consistent MAP point, given by $\lambda_T^* = 0.125$, is exactly the pre-image of d_{obs}. Indeed, since the linear map A is full rank in Example 1, the third term in (3) fully deregularizes (cancels out) the second term, so that the minimum of T is actually just the minimum of the first term, the data mismatch. We see here that classical Bayesian inverse problem theory and DCI pose and answer different questions. For a similar, concrete example where the number of parameters N is greater than the number of data, M (so that the problem is under-determined), we present the following example adapted directly from [**?**].

Example 2. *Let $f(\lambda) = 2\lambda_1 - \lambda_2$, $\lambda_{prior} = (0.1 \quad 0.2)^\top$, $C_\Lambda = diag[0.5, 0.25]$, $d_{obs} = 0.1$, and $C_\mathcal{D} = [0.25]$. Note that with $N > M$, we have an under-determined problem; in this case with $N = 2$ and $M = 1$, for any $\lambda \in \Lambda$, $f(\lambda) = d = 2\lambda_1 - \lambda_2$. Since there is just a single d_{obs}, we have $0.1 = 2\lambda_1 - \lambda_2$. We find*

$$S(\lambda) = 2(2\lambda_1 - \lambda_2 - 0.1)^2 + (\lambda_1 - 0.1)^2 + 2(\lambda_2 - 0.2)^2,$$

which is minimized by $\lambda_S^* = (7/50, 19/100)$. Assuming that λ_{prior} and d_{obs} are the means of Gaussians with standard deviations corresponding to the given covariance matrices, we have $\pi_\Lambda^{post} \sim \exp(-S(\lambda_S^*))$. Notice $f(\lambda_S^*) = 9/100 \neq d_{obs}$. In the data-consistent formulation, we have

$$T(\lambda) = 2(2\lambda_1 - \lambda_2 - 0.1)^2 + \frac{1}{9}\left((\lambda_1 - 0.1)^2 + 8(\lambda_1 - 0.1)(\lambda_2 - 0.2) + 16(\lambda_2 - 0.2)^2\right),$$

which is minimized by $\lambda_T^* = (13/90, 17/90)$. Notice $f(\lambda_T^*) = d_{obs}$. With Gaussian assumptions on the prior and data, we have $\pi_\Lambda^{post} \sim \exp(-T(\lambda_T^*))$. Figure 1 below shows plots of λ_S^* and λ_T^* along with the relevant visualizations of the terms in S and T.

In the example above, we see that the single datum $d_{\text{obs}} = 0.1$ could be observed by any point λ on the line $\lambda_2 = 2\lambda_1 - 0.1$, represented by the solid black line seen in several plots in Figure 1. The left-hand plots in Figure 1 are identical, since the data mismatch term in S and T are equal. The Tikhonov regularization term is depicted in the top-middle plot,

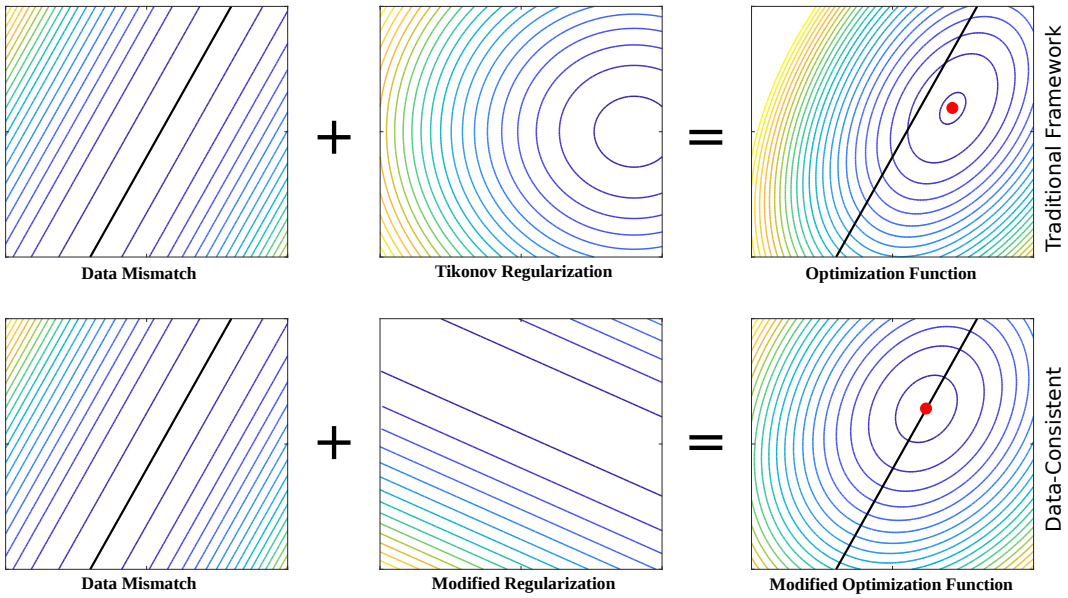

Figure 1: This figure from [CITATION NEEDED] shows the process of obtaining the statistical/classical Bayesian solution and data-consistent solution from solving the equivalent deterministic optimization problem corresponding to Example 2.

and the top-right panel shows the corresponding classical Bayesian solution with the MAP point λ_S^* shown as a red point and contours corresponding to the updated (Gaussian) prior distribution $\exp(-S(\lambda_S^*))$. Just as we saw analytically in Example 2, we see that λ_S^* does not lie on the black line representing data-consistent solutions. The action of the modified regularization – ie, the last two terms in (4) – is shown in the bottom-middle plot. Notice that the modified regularization is constant along directions perpendicular to the data, hence, the updated prior will be formed with prior information used only in directions that data is not informative. Thus, it is ensured that λ_T^* will lie on the line of data-consistent solutions, as seen in the bottom-right plot in Figure 1, where λ_T^* is represented as a red point and the data-consistent solution is given by contours of $\exp(-T(\lambda_T^*))$.

So far, we have presented a deregularization term in (3) which will hold only for linear (or linearizable) f and Gaussian assumptions on the parameter and data spaces. For general nonlinear functions f, the deregularization must be rewritten. Here, we arrived at an open research question, which we discuss further in the proceeding section.

Dimension Reduction We consider functions from a high-dimensional space Λ to a data space of much less dimension. Many of the processes of interest in this paper require function evaluations, which are generally expensive due to the many parameters. It is not often the case that every parameter has an equal impact on function values; usually some parameters matter more than others. If it is possible to mimic the response of f by processing fewer parameters, we can expect computations savings in many of the problems (i.e., minimization and forward problems) considered here.

In our setting, it may be advantageous to perform dimension reduction on f. In particular, we shall consider Active Subspace methods described by Paul Constantine in [**?**] and an

equivalent method by T.M. Russi in [**?**]. These techniques seek to explain outputs $f(\Lambda)$ in a subspace $A \subset \Lambda$ for which the $\dim(A) < N$. Here we discuss the theoretical formulation of A, but also how it may be found in practice.

We begin by noting $\nabla f(\lambda) \in \Lambda$ is a column vector of the same dimension as inputs λ with rows containing the N partial derivatives of f, which for this discussion we assume exist, and are square integrable in Λ equipped with some probability density that is positive everywhere in Λ and 0 otherwise; for instance, we could consider $\pi_\Lambda^{\text{prior}}(\lambda)$, the density describing our prior state of knowledge, which we choose to abbreviate as π_Λ for this discussion. In classical active subspace analysis, one transforms inputs λ to the origin with some fixed variance, typically so that $\lambda \in [-1, 1]$. Then, as in [**Constantine MC**], we may write

$$W = \int_\Lambda \nabla f(\lambda) \nabla f(\lambda)^\top \pi_\Lambda(\lambda) d\lambda, \tag{5}$$

which is an $N \times N$ symmetric positive semi-definite matrix which defines a certain covariance structure of ∇f over Λ. This interpretation of (5) leads one to the idea of computing the Singular Value Decomposition of W,

$$W = U \Sigma V^*, \tag{6}$$

where U is $N \times N$ unitary, Σ is $N \times N$ diagonal with the singular values of W along its diagonal, and V^* is $N \times N$ unitary. With the singular values of W in hand, we search for a drop-off in the spectrum of W. In detail, we plot the singular values, $\{\sigma_i\}_{i=1}^n$ and seek a drop-off in magnitude between some pair of singular values, σ_j and σ_{j+1}. The active subspace is the span of $u_1, \ldots, u_j$, which are the first j columns of W, the so-called "left singular vectors" of W.

The fact that $u_1, \ldots, u_j$ correspond to the nontrivial singular values is exactly why they account for the most amount of variance in function values. In fact, one can view active subspace analysis as an artful choice of principal components after a *full* Principal Component Analysis (PCA) is performed in the gradient space W; for more details on this viewpoint, we refer the interested reader to Section 6.4 in [**?**].

For a point $\lambda \in \Lambda$, we define

$$\mathcal{P}_A(\lambda) = \sum_{i=1}^{j} \left(u_i^T \lambda\right) u_i \in A, \tag{7}$$

which is a projection of the point λ in the "active directions" of f. We call this projection an "active variable," which is a point in the active subspace A. We have arrived at the property that

$$f\left(\mathcal{P}_A(\lambda)\right) \approx f(\lambda). \tag{8}$$

In practice, finding an active subspace of f will require forming an approximation to W in a Monte Carlo fashion; see [**Constantine MC**]. We choose to present a Monte Carlo approach that is simple to implement, as in [**?**]. In short, for a random draw $\lambda_i \in \Lambda$, we will find its evaluation under the approximate or analytic gradient (depending on whether we have ∇f analytically), and store the row vector $f(\lambda_i)^\top$ in a matrix with an SVD corresponding to (6), up to scaling.

In the following, we assume that we lack an analytic form of ∇f; if the analytic gradient is available, the proceeding Monte Carlo method remains valid, and all notations corresponding to approximating ∇f may be dropped.

One initializes the method by performing S random draws of $\lambda_i \in \Lambda$. We then compute $f(\lambda_i)$ for all $i = 1, \ldots, S$ samples, which we note will require, at the very least, S evaluations of f; in a realistic setting, this would require S solves of a model such as a PDE-constrained system. We define $D_S = \{(\lambda_i, f(\lambda_i))\}_{i=1}^S \in \mathcal{D}$, a set of S pairs of samples λ_i and their function values. Next, we need $\nabla_\Lambda f$ evaluated at λ_i for all $i = 1, \ldots, S$, which we assume that we do not have in closed analytic form. Hence, we generally need some gradient approximation method, and typically a locally linear approximation to the gradient is a fair balance between reasonably estimating the gradient and not pushing computational expenses to an unreasonable regime. With this approximation formed, we denote each estimation to $\nabla f(\lambda_i)$ with $\widehat{\nabla f}(\lambda_i)$ and we define the $N \times S$ matrix $\tilde{W}$ (which is presented below as $\tilde{W}^\top$)

$$\tilde{W}^\top := \left[\widehat{\nabla f}(\lambda_1)^\top \cdots \widehat{\nabla f}(\lambda_S)^\top \right]. \tag{9}$$

Forming the SVD of $\tilde{W}$, $\tilde{W} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$, we search for a drop off in the magnitude of the singular values $\{\tilde{\sigma}_i\}_{i=1}^S$. Assuming such a drop off occurs for an index $j : 1 < j < S$, we have the j corresponding left singular vectors,$\tilde{u}, \ldots, \tilde{u}_j$. We let $A(f; D_S) := \text{span}\{\tilde{u}, \ldots, \tilde{u}_j\}$ to denote the active subspace of f with respect to the samples D_S. We choose to use a notation with D_S included to emphasize the dependence of the active subspace on the random draws made in Λ, which led to our particular D_S set of samples.

In practice, we can check the extent to which the active subspace accounts for functions values $f(\lambda)$ by checking for resolution in a *sufficient summary plot* [**?**], where we plot active variables against function values. In these plots, we hope to see a pattern between the active variables versus their function values. For example, if f is quadratic in its active variables, then we expect to see quadratic-resolved sufficient summary plots.

Derivative-Free Optimization Many important physical systems possess turbulent or chaotic behavior. The physical state of the system $u(x, \lambda)$ and the corresponding parameter to observable map $f(u(x, \lambda))$ may be modeled as a stochastic process, or as a deterministic function with additive or multiplicative noise. In this setting, the efficient extraction of accurate gradients of f in parameter space is a challenging undertaking, as popular techniques based on linearization, including adjoint methods, are inaccurate [**?**, **?**]. The finite-difference approximations to ∇f_Λ involve $N = \dim\Lambda$ additional, usually nonlinear model solves for the physical system state $u(x, \lambda_i + \delta\lambda_i)$, and is greatly polluted by the noise in f.

As a consequence of these difficulties, the approximate data-consistent solution of a SIP in this setting by optimization techniques will need algorithms that do not require gradient information from f. We consider derivative-free optimization (DFO) algorithms suited for additive and multiplicative noise as in [**C/W**]. This technique requires nothing more than evaluations of the noisy model and random draws from a normal distribution. Briefly, this method finds a new iterate by randomly perturbing the previous iterate in Λ; iterates are

not allowed to stray much, though, due to relatively small smoothing factors and step sizes. The smoothing factor and step size in the DF algorithms are of great importance to their convergence and termination. As in [**CW**], both the smoothing factor and step size will depend on a scale factor of the L_1 Lipschitz constant of f. As such, it will be of interest to obtain estimates of L_1, which is not straightforward in a gradient-free setting. We refer to [**KS**] for Lipschitz constant learning in this setting, and discuss this problem more in the proceeding sections.

In detail, as in [**C/W**], we consider the problem

$$\min_{x\in\mathbb{R}^N} \quad \mathbb{E}\left[f(x)+\nu(x;\xi))\right], \tag{10}$$

where the authors assume that:

(i.) $f:\mathbb{R}^N\to\mathbb{R}$ is convex;
(ii.) $\xi\in\Xi$ is a random variable with probability density $P(\xi)$;
(iii.) for all $x\in\mathbb{R}^N$ the additive noise model ν is independent and identically distributed, has bounded variance σ_a^2, and is unbiased; i.e., $\mathbb{E}_\xi(\nu(x;\xi))=0$.

The use of the $\mathbb{E}(\cdot)$ expected value notation is a mathematically precise way of expressing the fact that the authors view function evaluations as a random draw, in the sense that true function values $f(x)$ are perturbed by a random vector in an additive or multiplicative fashion. Indeed, others [] make similar assumptions, viewing evaluations of f as a black-box process with noise, which obscures the true value of draws $f(\lambda)$.

Chen and Wild propose the "STARS (STep-size Approximation in Randomized Search)" algorithm, which: uses small perturbations in the domain Λ given by the addition of a random vector with components drawn from a normal distribution; computes the function value at the randomly perturbed point with additive or multiplicative noise drawn from a specified distribution; and updates iterates using a Gaussian-smoothed finite-difference scheme for approximate gradient information in a gradient descent scheme. We will examine the STARS algorithm in closer detail in the following sections. For now, we point out that the algorithm requires the ability to evaluate the model, and needs access to random draws from a normal distribution and the distribution from which noise is drawn. All in all, the algorithm can be implemented in about 10 lines of code in `python`.

Research Questions

In the following section, we pose research questions to be investigated within the framework provided in the preceding section.

Forming a Data-Consistent Deregularization for Nonlinear f For linear and linearizable f and Gaussian assumptions on Λ and $\mathcal{D}$, we have presented a data-consistent solution to the SIP by posing a deterministic optimization problem that is properly deregularlized so that prior beliefs are updated only in directions that the data is informative. For a general nonlinear map f, we must reformulate the deregularization which we recall may be written

$$\left|\left|C_A^{-1/2}(f(\lambda)-f(\lambda_{\text{prior}}))\right|\right|_2^2, \tag{11}$$

where $C_A = AC_\Lambda A^\top$. Expanding f linearly around λ_{prior}, we have $f(\lambda) - f(\lambda_{\text{prior}}) = \nabla f(\lambda_{\text{prior}})^\top(\lambda - \lambda_{\text{prior}})$ so that in a neighborhood centered at λ_{prior} we may write $f(\lambda) - f(\lambda_{\text{prior}}) = A(\lambda - \lambda_{\text{prior}})$, where $A = \nabla f(\lambda_{\text{prior}})^\top$, which is an $N \times M$ matrix. We may rewrite (11) as

$$(A(\lambda - \lambda_{\text{prior}}))^\top (AC_\Lambda A^\top)^{-1} A(\lambda - \lambda_{\text{prior}}). \tag{12}$$

In order to fully determine the form of (12), we must determine $(AC_\Lambda A^\top)^{-1}$. In fact, if we can solve $(AC_\Lambda A^\top)X = A(\lambda - \lambda_{\text{prior}})$ for X, then (12) becomes

$$(A(\lambda - \lambda_{\text{prior}}))^\top X. \tag{13}$$

The conditioning of the linear algebra problem of finding such an X is not characterized here, and will be a focus of research. We refer to [**?**] for a similar Newton-based approach. We note that since we assume ∇f is inaccessible, forming the data-consistent deregularization term in this regime may require gradient approximation methods for f.

Impact of Noise on MAP Point and Updated Prior The effects of noise on individual methods considered in this paper – particularly: DCI, the deterministic optimization solution for SIPs, active subspaces, and DFO) – have been characterized to varying extents in corresponding literature. We are particularly interested in investigating the impact of additive and multiplicative noise models in our framework, which essentially involves embedding the aforementioned methods within each other, performing them in series, or both.

Formally, we investigate both sensitivity and stability of any proposed methods. Theoretically, this may involve using the individual sensitivity/stability analyses of certain methods while deriving new results when the methods are embedded or performed in series. In practice, this may involve comparing the results of individual methods to nested ones, and observing errors and deviations in results.

Learning and Sampling With the viewpoint that evaluations of the model are costly, we want to compute $f(\lambda)$ as little as possible with learning as much as possible about the function. We are interested in several problems that involve sampling Λ and evaluating f including solving the forward problem, finding an active subspace via Monte Carlo, and performing DFO. We also investigate the possibility of learning the L_1 Lipschitz constant of f from random sampling.

Learning the Active Subspace We are interested in several research questions regrading the active subspace learned from different sets of samples in Λ. Generally, for fixed f, we are interested in comparing active subspaces obtained from sampling f with few samples, or with samples generated from some other process, such as a DFO algorithm.

A challenge in learning the active subspace is exploring Λ thoroughly enough to resolve f in fewer variables. The samples (i.e., iterates) formed in a DFO algorithm are only allowed to stray from previous samples by a value proportional to the L_1 Lipschitz constant of f; hence, iterates may explore the space in Λ near the initial iterate in a DFO scheme very thoroughly while missing other global phenomenon. If we only wish to find an active subspace of f in some neighborhood of a point $\lambda \in \Lambda$, such samples may suffice; if we hope to achieve a global understanding of A, we may need to consider supplementing the samples with other, more explorative draws in Λ.

Lipschitz Constant Learning We explore ways in which one may approximate the L_1 Lipschitz constant of f using sampling. Since many DFO algorithms use parameters for step sizes and smoothing that depend on L_1, the literature [**KS**] contains attempts to learn Lipschitz constants from sampling alone. We are most interested in methods that require few evaluations of f.

Using the Active Subspace Generally, if an active subspace A exists for a model f, dimension reduction may be performed by, for instance, projecting inputs λ into A then evaluating f for the projection; i.e., form $f\left(\mathcal{P}_A(\lambda)\right)$. Computational expense may be saved by projecting points λ into their so-called active variables, since such projections are of lower dimension than λ. This property gives the ability to save computational expense for a number of problems in Uncertainty Quantification, including optimization, representation and solving inverse problems. In the following, we consider ways in which we may use the information given by an active subspace $A(f; D_S)$.

DFO in the Active Variables Only Given some f and its corresponding active subspace $A(f; \mathcal{D})$ found by the Monte Carlo method discussed in the preceding section, we are interested in investigating the effectiveness of only optimizing f in its active variables. There are several approaches one may consider, and handful of those approaches and their corresponding results are discussed in the proceeding section. The most compelling approach we have observed is to modify the DFO algorithm discussed above to only take random walks in directions lying in A. That is, at iteration k, standard DFO algorithms [**CW**] use random walks given by drawing a random vector $v^{(k)}$ of dimension N in which every component $v_i^{(k)}, i = 1, \ldots, N$ of v is drawn from a specified normal distribution. Instead, given the first j singular unit vectors $u_1, \ldots, u_j$ corresponding to the SVD of $\hat{W}$, one may take j draws from a specified normal distribution, which we denote with $s_i \sim N(\mu, \sigma^2)$, and form the random vector v for the k-th step in a DFO algorithm as $v^{(k)} = \sum_{i=1}^{j} s_i u_i$, which is just a linear combination of the active variables of f with coefficients given by random draws; we see that such a vector could be interpreted as a random walk in A.

Relevant research directions include: considering other ways to embed the information from A within a DFO scheme; analyzing the difference between minimizing f in all its variables versus only in its active subspace; considering ways in which the problem may be solved with the mentioned techniques in tandem, such as using a "burn-in" phase in which f is minimized over all of Λ followed by an active variables minimization.

Representations and Surrogate Modeling We are generally interested in the representation and surrogates one may form with an active subspace. If f is truly a black box, then it lacks an analytic form we can access. By performing dimension reduction, it is possible to test the resolution of function values f versus active variables by forming a sufficient summary plot or *response surface* [**?**]. Depending on the quality and form of the resolution, one may (or may not) be able to discern the relation between f and its active variables visually, by fitting a surrogate through the response surface and testing the fit, or both. Representing f with a lower-dimensional surrogate may help with solving other problems of interest. In particular, we may be able to solve a forward problem at lower cost, or pose the convex optimization for solving an inverse problem with such a representation used to form the misfit functions considered in the preceding section.

Solving the Forward Problem If the action of the map f can be captured by a lower-dimensional representation, we may be able to more cheaply solve for the push-forward needed for DCI. Research directions include: generally comparing the cost of solving the forward problem with KDE for an f defined in its full variables versus some surrogate for f defined over A; analyzing differences in MAP points and updated densities in Λ when the push-forward is obtained from its full variables or only from active variables.

Application: Callibrated Anomolous Diffusion in Tokamak Plasmas Transport in tokamak plasmas can be strongly driven by a combination of *neoclassical* effects and plasma microturbulence. Neoclassical theory modifies classical diffusion by adding effects due to the geometry of the fusion reactor. Complex particle motion, such as banana orbits, may be described by the solution of an axisymmetric version of the Vlasov-Boltzman equation [**?**]. Simulating the microturbulence in the plasma involves more expensive kinetic simulations, especially when simulating the plasma edge [**?**]. An alternative, analogous approach to turbulence modeling in hydrodynamic turbulence is to fit an anomalous diffusion model to experimental or high-fidelity simulation data. Deterministic efforts in this setting include [**?**]. We propose to investigate the data-consistent calibration of an anomalous diffusion model in an axisymmetric setting(XGCa) to high-fidelity experimental or simulation data(XGC1) [**?**].

Application: Magnetic Equilibria in Tokamaks

Varis: I'll write this, covering Grad-Shafonov Equations, and why these MHD equilibria are important to kinetic simulations.

Varis: I'll write a brief section transitioning from current toy problems to pde models to plasma physics models

Preliminary Results and Research Plan

In this section, we re-visit the questions from the previous section to discuss preliminary results, if any, and present a plan to begin investigating questions that are unanswered. We present a guiding example which is discussed in below sections in the context of relevant research questions.

Example 3. *Let* $\Lambda = [-1, 1]^{10}$ *and define*

$$f(\lambda) = \sum_{i=0}^{10} 2^{(-1)^i i} \lambda_i^2 + \epsilon(\lambda),$$

where $\epsilon(\lambda)$ *is a draw of additive noise corresponding to the input* λ*; here, we take draws of* ϵ *of order* 10^{-4}*. We see that* $\mathcal{D} = [0, 2^{10}]$ *and* $N = 10$*,* $M = 1$*. Note that the minimimum of* f *is given by* $0 \in \Lambda$*. Here, as* i *increases, terms in* f *become either more important or less important, depending on whether* i *is even or odd.*

Forming a Data-Consistent Deregularization for Nonlinear f Forming a data-consistent mismatch function for a nonlinear f will require manipulating only the deregularization term, which in the linear case, uses a matrix A to perform the action of f on inputs. For an f that is not linear (or easily linearizable), we must consider ways in which f may expressed for the deregularization. This is a theoretical task which will require a closer literature review and heavy collaboration with advisers.

While the attempts at formulating the nonlinear deregularization in the preceding section – particularly (13) – may be correct mathematically, it may not be a practical or tractable approach for solving for such a form. Ideas to explore may include ways in which the active subspace of f could lead to more easily obtained linearizations around particular points in

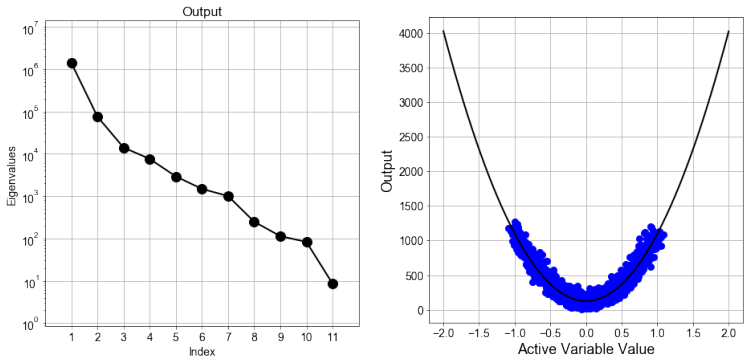

Figure 2: Left: A plot of the eigenvalues of the matrix $\hat{W}$ formed from 1000 Monte Carlo samples in Λ. We see one dominant eigenvalue on the order of 10^6. Right: A sufficient summary plot where all 1000 samples are projected into A and plotted against their function values.

Λ. As we will observe, the f as defined in Example 3 is globally quadratic with respect to its active subspace, and won't permit a global linearization. We may consider, however, a local linearization at various points on the response surface $f(\mathcal{P}_A(\lambda))$ may be acceptable, particularly near 0.

Impact of Noise on MAP Point and Updated Prior Thus far, the impact of noise in our framework has only been observed, not rigorously characterized. The impact of noise on each of the processes of interest (i.e., the minimization solution to DCI; DFO; solving for active subsapces) has been characterized in the literature to different extents.

We have performed several numerical experiments in which f with additive noise is minimized with DFO at the same time that the active subspace is formed. For comparison, we have also performed numerical experiments in which f with additive noise is minimized via DFO with no active subspace analysis, and likewise we have searched for the active subspace of f using random points in Λ (not given by a DFO algorithm).

In Example 3, we consider a noisy, convex form f to be minimized. We find that with small additive noise (on the order of 10^{-4}), the active subspace and minimizers obtained from various processes match our intuition, and the impact of the additive noise in this regime is negligible. The first concrete research step to obtain experimental evidence of noise effects will be increasing the noise level until results no longer match expectations. Of course, experimental evidence is useless with no corresponding theory; we plan to develop a rigorous characterization of the impact of noise on many of the algorithms (or series of algorithms) studied.

Learning and Sampling We are particularly interested in the quantitative and qualitative difference between active subspaces learned from random sampling versus active subspaces learned from deterministic draws that come from another algorithm such as DFO.

We revisit Example 3, and present two different active subspaces of f – one active subspace was formed in a Monte Carlo fashion with 1000 random draws in Λ; the other subspace was formed with only 100 random sample of f where sample points are in fact iterates from a DFO scheme. To consider a realistic scenario, we form the active subspace in both cases

by making global quadratic approximations to ∇f, which will work well in our regime since f is quadratic in Λ.

In Figure 2, we show the results from the first active subspace of f, which was obtained from 1000 iid uniform random samples in $[-1,1]^{10}$; i.e., for any sample λ_i, we have $\lambda_i \sim U[-1,1]^{10}$. The plot on the left-hand side of Figure 2 shows the resulting eigenvalues (the squares of the singular values) from taking the SVD of $\hat{W}$, which we recall is simply a matrix (in this case 1000×10) where row i is the transpose of approximated gradient of f evaluated at a sample λ_i, $i = 1, \ldots, 1000$. Notice that there is one dominating eigenvalue, which is on the order of 10^6; this matches intuition since the final term in $f(\lambda)$, given by $2^{10}\lambda_{10}^2$, will be, at most, on the order of 10^3 (and again, eigenvalues are the squares of singular values). This eigenvalue corresponds to λ_{10}, meaning that the value of λ_{10} has the most impact out of all other parameters on the function values $f(\lambda)$. Notice that the eigenvalue near 10^6 does not dominate the other eigenvalues by much; indeed, another eigenvalue appears around the order 10^5. Since the second-to-last term in f is small (with coefficient 2^{-9}), this order 10^5 eigenvalue must correspond to $2^8\lambda_8^2$.

The active subspace implementation used here [**?**] detects the 10^6 order eigenvalue as dominating enough to determine that A is one-dimensional, spanned by the singular vector corresponding to the direction $(0, \ldots, 0, 1)$, which is a unit vector in the axis of λ_{10}, orthogonal to all other axes in Λ.

The plot on the right-hand side of Figure 2 shows the sufficient summary plot, where the 1000 samples are projected into A using (7) and their values are plotted along the horizontal axis against their original function values on the vertical axis. The sufficient summary plot can be thought of as a visualization of f along the axes for which f is active; thus, we are seeing a quadratic response in f versus points projected onto the λ_{10} axis. We use the mentioned software tools to fit the response with a polynomial surrogate, which is a fit where $R^2 \approx 0.9$. Indeed, points do not fall perfectly along the surrogate since there are other parameters which contribute to the value of f.

Now we investigate the effectiveness of learning A with different samples. Since we want to avoid evaluating f heavily and we want to perform several analyses of f, it will be advantageous to "recycle" function values whenever possible.

To experiment, we consider trying to learn A from 100 iterates and their function values generated from performing the DFO algorithm STARS [**C/W**] on f. The iterates generated from a DFO algorithm may make large jumps (limited by a step size and smoothing parameter) through regions of Λ space early in the routine, but will begin to take smaller and smaller steps as the routine hones in on a local minimum. We note that typically for a fair comparison between sampling methods, one would use the same number of samples; however, we choose to use much less samples in the DFO case for the exact reason that the samples will cease to explore parameter space in an informative manner once the iterates begin to converge to a local minimum. In Figure 3, we present the sufficient plot obtained from learning A from 100 DFO samples. The eigenvalue plot obtained from the analysis was virtually identical to that in Figure 2, but the sufficient summary plot here in Figure 3 looks very different than the plot in Figure 2 since the sampled points were not random, but deterministic. Notice that the large function value occurs in the top-left corner of the sufficient summary plot in Figure 3; this value corresponds to the initial iterate $\lambda^{(0)}$, which had a function value of approximately 460. Iterates drive down the value of f, and we see

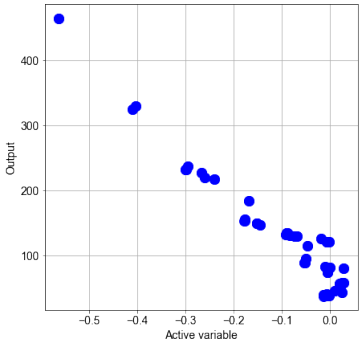

Figure 3: A sufficient summary plot where all 100 samples are iterates from a DFO algorithm.

that the large majority of the iterates (samples) occur in the bottom-right corner of the plot, where the DFO algorithm is converging to a local minimum.

We find that in our case the samples generated from the DFO algorithm explored Λ enough to give an active subspace with negligible differences to that obtained in the case of 1000 Monte Carlo draws.

We are interested in exploring the extent to which A can be learned for more general functions f, particularly those arising in certain applications. We are also interested in learning the L_1 Lipschitz constant of f in this setting. For the example analyzed here, the step size and smoothing parameters, which are functions of the value of L_1, were formed heuristically. A more careful analysis is left to be desired, especially in the case that ∇f is inaccessible so that L_1 must be estimated. Techniques in the literature need to be considered and tested.

Using the Active Subspace We are generally interested in using the active subspace of a function whenever it may cheapen computational expense while maintaining a representations that does not deviate too far from the true action of f. Here we consider the effectiveness of DFO performed on f from Example 3 in a standard way [**C/W**] using 500 iterations versus a scheme that blends STARS and active subspace analysis using only 150 iterations.

We find that after 500 iterations of the standard DFO implementation the routine closes in on the minimum value of f, finding that $f\left(\lambda^{(500)}\right) \approx 0.84$. In Figure 4, the left-hand plot shows a *log-log* plot of the logarithm of the iteration on the horizontal axis against the logarithm of the function evaluated at that iterate. We begin to see convergence to a minimum in 500 iterations. We modified the DFO algorithm to find a similar result with only 150 iterations.

To make a modified DFO routine, we begin with a "burn in" of 50 iterations of standard DFO, which actually gives 100 samples of the function at different points in Λ, as one step of the DFO routine has two calls of f. From the 100 samples denoted with D, we form $A(f, D)$. The results of this analysis were discussed above; recall, we obtain a one-dimensional A spanned in the direction of λ_{10}. Though A is one-dimensional, there are still other important parameters in the model, including λ_8 and λ_6, which make order 10^2 and 10^1 contributions to f respectively. After the standard DFO burn-in, which was used to both minimize f and explore Λ to obtain an approximate A, we begin to use DFO only in

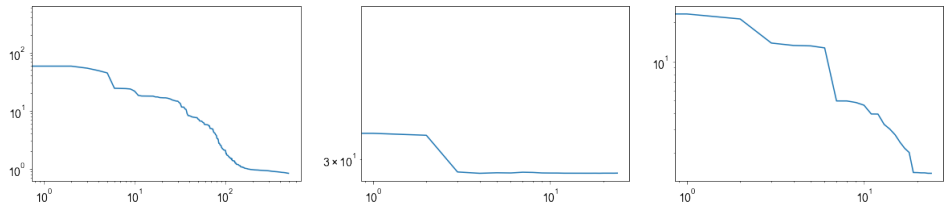

Figure 4: 3 log-log plots from (left to right): 500 iterations of standard DFO; 25 iterations of minimizing f with DFO along the λ_8 axis; 25 iterations of minimizing f with DFO along the λ_6 axis.

the directions of the active and some inactive variables. Since we have only a single direction to minimize along in the active subspace, we modify the DFO algorithm to take random steps in the direction of the λ_{10} axis only. We do so by taking draws from a $d \sim N(0,1)$ and forming the randomly scaled vector du_1, where u_1 is the singular vector corresponding to the active direction in λ_{10}. Now, each iterate will only be perturbed in its last component in the DFO scheme. We find that in this example, the STARS burn-in phase reduces the 10th component of iterates so that $\lambda_{10}^{(50)} \approx -0.01$ and $f(\lambda^{(50)}) \approx 38.02$. Performing DFO in the λ_{10} axis with 25 iterations gives $\lambda_{10}^{(75)}$ on the order of 10^{-4}, but $f(\lambda^{(75)}) \approx 37.74$, which is only a small drop in the function value. Next, 25 iterations of DFO was performed along the λ_8 axis – this time, we see a significant drop in the value of f, where $f^{(100)} \approx 29.34$. Despite the fact that the λ_8 direction is not considered active, we do see the value of f decrease as f is minimized along its axis. Another 25 iterations of DFO was performed along the direction corresponding to the third singular vector, dropping the value of f by about 6. Finally, 25 iterations of DFO was performed along the axis corresponding to the fourth singular vector, which minimizes f to an order similar to the results from 500 iterations of standard DFO. Indeed, $f(\lambda^{(150)}) \approx 1.38$.

In the middle plot of Figure 4, we show the log-log plot obtained from using 25 steps of DFO in the λ_8 direction; the right-hand log-log plot shows the results of 25 steps of DFO in the λ_6 direction.

We note that we have presented just one of several algorithms that have in some way blended the active subspace of f with a DFO scheme. We are interested in investigating the best way to use the information of A in DFO. Likewise, we are interested in representing f with surrogates based on A and investigating whether A can lessen the computational expense in the forward problem.

Software Dissemination Currently, only some of the software used to produce results in this paper are publicly available on GitHub.com. Some of the algorithms used here and other algorithms that are of interest are within open-source packages available online including those of Constantine and Butler et al. Other schemes considered in this paper make modifications to given algorithms and remain under development. A major goal of this thesis proposal will be developing well-documented, open-source software complete with python Jupyter Notebooks and illustrative, replicable examples.

Timeline

We outline a rough timeline for the remaining 2 years in a 5.5 year plan.

- Clean existing algorithms and examples, generate richer research results related to DFO and active subspaces, and build a model inverse problem for investigation. (Dec 2018/Jan 2019)
- On RA for Spring 2019.
- Write and present MS-level results. (Feb/Mar)
- Work on theoretical formulation of deregularization for nonlinear f. Work on generating notebooks and examples. (Ongoing/Spring and Summer 2019)
- Summer research? Begin writing thesis? Summer school/internship/conference? (Jun/Jul/Aug)
- Fall 2019 - writing phase, revisions
- Spring 2020 - final revisions, software
- Summer 2020 internship/collaborations?, publishing, formatting
- Defense summer 2020 or early fall