Reliable Real-Time Optimization of Nonconvex Systems Described by Parametrized Partial Differential Equations

I. B. Oliveira and A. T. Patera
Massachusetts Institute of Technology
Department of Mechanical Engineering, Room 3-266
Cambridge, MA 02139-4307 USA

Abstract— **The solution of a single optimization problem often requires computationally-demanding evaluations; this is especially true in optimal design of engineering components and systems described by partial differential equations. We present a technique for the *rapid* and *reliable* optimization of systems characterized by linear-functional outputs of partial differential equations with affine parameter dependence. The critical ingredients of the method are: (*i*) reduced-basis techniques for dimension reduction in computational requirements; (*ii*) an "off-line/on-line" computational decomposition for the rapid calculation of outputs of interest and respective sensitivities in the limit of many queries; (*iii*) *a posteriori* error bounds for rigorous uncertainty and feasibility control; (*iv*) Interior Point Methods (IPMs) for efficient solution of the optimization problem; and (*v*) a trust-region Sequential Quadratic Programming (SQP) interpretation of IPMs for treatment of possibly non-convex costs and constraints.**

I. Introduction

Optimization of engineering components often requires computationally demanding evaluations, especially when considering systems described by partial differential equations (PDEs). Standard application of typical modeling methodologies — Finite Element Methods, for example — in the context of optimization often results in computational storage and time demands that exceed current resources. Typical problems, however, rarely require the entire solution field for optimal design. Instead, a comparatively small collection of *design* parameters and *output* functionals are often found as costs or constraints that govern the behavior of the system in question. That is, only a relatively small *design space* is required to affect a relatively small *output space*. We consider here a subset of such parametrized problems that exhibit a relatively low order "affine" parameter dependence which, we later show, allows for a particularly attractive computational decomposition. These problems are presented and described in Section II.

To exploit this low-order parametric dependence we introduce the first component of the proposed method: *reduced-basis approximations* for the calculation of outputs of interest and their first- and second-order parameter sensitivities (i.e. gradients and Hessians). Though field variables lie in high dimensional "state" spaces, they vary parametrically in much lower-dimensional manifolds. Reduced-basis techniques guarantee rapid convergence by optimally approximating the state-space solution on this much lower-dimensional set. In order to exploit the problem structure and maximize the gains afforded by the reduced-basis framework, we introduce an *off-line/on-line computational decomposition*, which allows for fast, "real-time" calculation of variables and sensitivities in the limit of many queries — precisely the case of interest in the optimization context.

Without accounting for errors introduced by approximation it is possible that a feasible reduced-basis solution parameter set renders the problem infeasible in the "true" solution. It thus becomes critical to accurately quantify errors introduced by the reduced-order approximation so as to rigorously guarantee feasibility — often synonymous with either "reliability" or safety in the engineering context. The third component of the proposed method, *a posteriori error estimation*, provides real-time error bounds and has been developed (see [4], [5], [6], [7], [8]) to satisfy our feasibility demand without being overly conservative: the bounds are rigorous and sharp. We present an overview of the reduced-basis method, off-line/on-line computational procedures, and our error estimation approach in Section III.

The fourth ingredient of our approach, *Interior-Point Methods* (IPMs), addresses the actual optimization of the system. The systems under consideration are governed by elliptic PDEs, which tend to have continuous, differentiable solutions: IPMs adapt well to these types of problems, and much recent research activity has made these methods very attractive. In developing our IPMs formulation, however, care must be taken to preserve the significant computational savings and "uncertainty management" provided by our reduced-basis approach. We describe our IPMs approach in Section IV.

The final ingredient of the proposed method is *a trust-region Sequential Quadratic Programming interpretation of*

the IPMs formulation: this is essential in ensuring that a (local) optimum, rather than simply a stationary point, is obtained. Though the functional dependence of the outputs of interest is linear with respect to field variables, the outputs (cost objectives, constraints) can be *nonlinear* with respect to the parameters (design variables). Furthermore, engineering problems are typically nonconvex; in this case it can be shown that direct application of IPMs (as originally developed for linear problems) to nonlinear programs will often result in slow convergence and, more importantly, solutions that are stationary but not optimal. In Section IV-C we present a method that avoids these pitfalls.

Finally, in Section V, we illustrate our method: we consider the optimal design of a three-dimensional thermal fin characterized by an (elliptic) affinely-parametrized partial differential equation that yields nonconvex input-output dependencies.

II. Problem Statement – Abstract Formulation

We denote as the "forward problem" the calculation of input-output relationships governed by partial differential equations. That is, given a set of design parameters, we wish to find the value of outputs of interest.

A. Forward Problem

Consider a suitably regular domain $\Omega \subset \mathbb{R}^d$, $d = 1,\ 2,$ or 3, with boundary $\partial\Omega$, and associated function spaces $H^1_0(\Omega) \subset Y \subset H^1(\Omega)$, where $H^1(\Omega) = \{v \in L^2(\Omega) \mid \nabla v \in (L^2(\Omega))^d\}$, $L^2(\Omega)$ is the space of square integrable functions over Ω, and $H^1_0 = \{v \in H^1(\Omega) \mid v|_{\partial\Omega} = 0\}$. The inner product and norm associated with Y are given by $(\,\cdot\,,\,\cdot\,)_Y$ and $\|\cdot\|_Y = (\,\cdot\,,\,\cdot\,)^{1/2}$, respectively. Also define a parameter set $\mathcal{D} \in \mathbb{R}^P$, a typical point of which will be denoted x.

We assume that we require the evaluation of $(K+M)$ outputs associated with a so-called "forward" problem; we will later see that K such outputs will be assigned to cost objectives and M to constraints, and acknowledge that this definition may be redundant (outputs may simultaneously be included as costs and constraints) for convenience. The abstract formulation of the forward problem is as follows: for any $x \in \mathcal{D}$, find the j^{th} output $s_j(x) \in \mathbb{R}$ given by

$$s_j(x) = c_j(x) + \ell_j(u(x)), \tag{1}$$

where c_j is a nonlinear function of x, $\ell_j : Y \to \mathbb{R}$ is a linear functional, and $u(x) \in Y$ satisfies the partial differential equation (in weak form)

$$a(u(x), v; x) \;=\; f(v), \quad \forall\, v \in Y. \tag{2}$$

Note that s_j is affine in u but nonlinear in x. We assume here that the bilinear form $a(\,\cdot\,,\,\cdot\,;x)$ is symmetric, $a(w,v;x) = a(v,w;x), \forall\, w,v \in Y$; continuous, $a(w,v;x) \le \gamma(x)\|w\|_Y\|v\|_Y \le \gamma_0\|w\|_Y\|v\|_Y, \forall\, x \in \mathcal{D}$; and coercive, $0 < \alpha_0 \le\ \alpha(x) = \inf_{w\in Y} \frac{a(w,w;x)}{\|w\|^2_Y}, \forall\, x \in \mathcal{D}$; and that the linear functionals $f(v)$ and $\ell_j(v)$ are bounded for all $v \in Y$. We consider here only the *compliant* output $\ell_j(v) = f(v)$ (other outputs that are not related to u may still exist), and note that $f(v)$ has been defined independently of x. We have required symmetry of a, compliance, and the independence of f and ℓ on x to simplify the exposition, and note that our framework easily allows these assumptions to be relaxed (see [6], [7], [8]).

We also make certain assumptions on the nature of the parametric dependence of a. We require that, for some finite (preferably small) integer Q, $a(w,v;x)$ be expressible as

$$a(w,v;x) \;=\; \sum_{q=1}^{Q} \sigma^q(x)\; a^q(w,v), \tag{3}$$

$\forall\, w,v \in Y,\ \forall\, x \in \mathcal{D}$, for appropriately chosen functions $\sigma^q : \mathcal{D} \to \mathbb{R}$, and associated x-independent bilinear forms $a^q : Y \times Y \to \mathbb{R}$, $q = 1,\ldots,Q$. We note that we pose our problem on a *fixed* reference domain Ω (i.e., Ω does not depend on x) to ensure that the parametric dependence on geometry enters through $a(\,\cdot\,,\,\cdot\,;x)$ and ultimately through $\sigma^q(x)$.

B. Optimization Problem

Having presented the mathematical model of the forward problem, one can solve the required equations to arrive at the outputs of interest given a certain configuration x. We are interested here in solving an optimization problem; that is: find the optimal configuration x^* such that a given cost/performance measure is minimized. Our goal is to solve such problems that involve arbitrary, nonlinear (in x) cost and constraint functions governed by partial differential equations of the type described in the previous section.

We define the cost functional as

$$J(x) = \sum_{j=1}^{K} \omega_j s_j(x) \tag{4}$$

where ω_j represent prescribed positive weights and $s_j(x)$ represent functionals of the type defined in (1). Cost functionals may in fact take more general forms, though we will consider the above for simplicity. Note that $J(x)$ can be nonlinear in x. We allow the following constraint types: parameter constraints, $x_{i\min} \le x_i \le x_{i\max}$ for $i = 1,\ldots,P$; and outputs constraints, $s_{j\min} \le s_j(x) \le s_{j\max}$ for $j = 1,\ldots,M$. In summary, we address the following optimization problem:

$$\text{find} \qquad x^* \;=\; \arg\min_{x\in\mathcal{D}} J(x)$$

$$\text{subject to} \quad \begin{cases} s_{\min} \le s(x) \le s_{\max}, \\ x_{\min} \le x \le x_{\max}, \end{cases}$$

where $x \in \mathcal{D}$ and $s(x) \in \mathbb{R}^M$ represent vectors of size P and M, respectively.

III. Reduced-Basis Methods for Forward Problem

The reduced-basis method is critical in allowing for the real-time nature of the proposed approach. We present here a brief exposition, and direct the reader to [8] and [9] for a more complete survey.

A. General Formulation

Given the parameter set $\mathcal{D}$ we randomly choose N points $x^n \in \mathcal{D}$, $n = 1, \ldots, N$, the collection of which we denote as $S^N = \{x^1, \ldots, x^N\}$. We then introduce a Lagrangian reduced-basis approximation space $W^N = \text{span}\{u(x^n),\ n = 1, \ldots, N\}$, where $u(x^n)$ is the solution of the forward problem: $a(u(x^n), v; x^n) = f(v),\ \forall\, v \in Y$. The reduced-basis approximation of the j^{th} output is determined as follows: for any $x \in \mathcal{D}$, first find $u_N(x) \in W^N$ such that

$$a(u_N(x), v; x) = f(v), \quad \forall\, v \in W^N, \tag{5}$$

then approximate $s_j(x)$ by $s_{Nj}(x) = \ell_j(u_N(x))$.

Problem (5) is of much lower dimension than (2). In particular, while the former problem is infinite-dimensional (or very high-dimensional for accurate finite element approximations) the latter is of dimension N. Furthermore, it has been shown in [8] that for some problems the reduced-basis approximation converges exponentially fast in N, which suggests that the method can be highly accurate for relatively small N. Section III-B will present a method for estimating the error introduced by a given basis selection; detailed computational requirements and implications will be presented in Section III-C.

Before proceeding we note that the optimization procedure will require, in addition to approximations of $s_j(x)$, higher order information such as $\nabla_x s_j(x)$ and $\nabla^2_{xx} s_j(x)$ — we also use the notation $s'_j(x)$ and $s''_j(x)$ here to indicate, respectively, gradients and Hessians of $s_j(x)$ *with respect to* x. It can be shown that these quantities can be approximated within the reduced-basis context as

$$\begin{aligned} s'_{Nj}(x) &= c'_j(x) + \ell(u'_N(x)), \\ s''_{Nj}(x) &= c''_j(x) + \ell(u''_N(x)), \end{aligned}$$

where $u_N(x)$ is determined as above. The calculation of $u'_N(x) \in (W^N)^P$, $u''_N(x) \in (W^N)^{P\times P}$, and remaining quantities is addressed in Section III-C, where we will show how the special structure of the problem can be exploited.

B. A Posteriori *Error Bounds*

The reduced-basis method introduces errors which can render the solution infeasible. Before we address the optimization framework, we provide here a method for determining rigorous *a posteriori* bounds for the errors in the predicted output — which, in essence, will guarantee that the reduced feasible set is contained in the true feasible set.

More specifically, we develop output estimates $\bar{s}_{Nj}(x)$ and related bound gaps $\Delta_{Nj}(x)$ which provide upper and lower *bounds* for the j^{th} output, $s^{\pm}_{Nj}(x) = \bar{s}_{Nj}(x) \pm \Delta_{Nj}(x)$, such that $s^{-}_{Nj}(x) \le s_{Nj}(x) \le s^{+}_{Nj}(x)$. We require that the bounds be sharp, so as not to mislead the design process (and yield suboptimal solutions), and that they not be significantly more computationally demanding than predictions of s_{Nj}. We only present the method here and refer to [8] for proofs and details.

We begin by assuming that we have a function $\hat{g}(x) : \mathcal{D} \to \mathbb{R}_+$ and a continuous, coercive, symmetric (and x-independent) bilinear form $\hat{a} : Y \times Y \to \mathbb{R}$ such that $\hat{\alpha}_0 \|v\|^2_Y \le \hat{g}(x)\, \hat{a}(v, v) \le a(v, v; x),\ \forall\, v \in Y,\ \forall\, x \in \mathcal{D}$, for some $\hat{\alpha}_0 > 0$. References [8] provide easily computable suggestions for $\hat{g}$ and $\hat{a}$. We then find $\hat{e}(x) \in Y$ such that

$$\hat{g}(x)\, \hat{a}(\hat{e}(x), v) = R(v; u_N(x); x),\ \forall\, v \in Y,$$

where $R(v; w; x) \equiv f(v) - a(w, v; x),\ \forall\, v \in Y$, is the residual. Finally,

$$\begin{aligned} \bar{s}_{Nj}(x) &= s_{Nj}(x) + \frac{\hat{g}(x)}{2}\, \hat{a}(\hat{e}(x), \hat{e}(x)), \\ \Delta_{Nj}(x) &= \frac{\hat{g}(x)}{2}\, \hat{a}(\hat{e}(x), \hat{e}(x)). \end{aligned}$$

We may then evaluate our bounds as $s^{\pm}_{Nj}(x) = \bar{s}_{Nj}(x) \pm \Delta_{Nj}(x)$.

C. Computational Procedure

In practice, the $u(x^n)$, $n = 1, \ldots, N$, is replaced by finite element approximations on a sufficiently fine "truth mesh". For simplicity, assume the finite element approximation requires $\mathcal{N} \gg N$ degrees-of-freedom. Since equation (1) is defined in terms of $u(x)$, we are required to solve problem (2) in order to evaluate $s_j(x)$: this calculation scales as $\mathcal{O}(\mathcal{N}^\star)$, for $\star$ some exponent grater than unity. Gradient and Hessian formations require $\mathcal{O}(P\mathcal{N}^\star)$ and $\mathcal{O}(P^2\mathcal{N}^\star)$ operations, respectively. This "exact" strategy is inefficient since IPMs only require approximate evaluations for most iterates.

We can exploit reduced-basis and the structure of the problems considered here to build what we term "off-line" and "on-line" stages that will allow for significant computational savings. The central motivation is that we plan to perform N "expensive" ($\mathcal{O}(\mathcal{N}^\star)$-type) calculations before beginning the optimization procedure. Then, using these results as an off-line database, we perform the entire optimization procedure in W^N, where the problem scales with a power of N rather than a power of $\mathcal{N}$. Assuming we are able to incorporate rigorous error estimation (without significant increase in computational cost), we can expect to arrive at rigorously feasible solutions that can be arbitrarily accurate in the sense of proximity to true optimality.

Since the problems in question require the solution of a PDE, we expect the calculation of outputs associated with $u(x)$ to be far more expensive than those that have only an explicit dependence on x. Therefore, let us consider in this section outputs s_{Nj} associated with u: in our case, the single compliant output (as we have mentioned, problems that

include a number of affine noncompliant outputs can be considered by simple extensions to the presented framework). We begin by expressing the field variable approximation as $u_N(x) = \sum_{i=1}^N u_{Ni}(x)\zeta_i = \underline{u}_N(x)^T\underline{\zeta}$, where $\underline{u}_N(x) \in \mathbb{R}^N$. Then, a Galerkin projection of (5) in W^N yields

$$\underline{A}_N(x)\,\underline{u}_N(x) \;=\; \underline{F}_N \tag{6}$$

and compliant output evaluation $s_{Nj}(x) = F_N^T\,\underline{u}_N(x)$. Here $\underline{A}_N(x) \in \mathbb{R}^{N\times N}$ is the SPD matrix with entries $A_{Ni,j}(x) \equiv a(\zeta_j,\zeta_i;x)$, $1 \le i,j \le N$, and $\underline{F}_N(x) \in \mathbb{R}^N$ is the "load" (and "output") vector with entries $F_{Ni} \equiv f(\zeta_i)$, $i = 1,\ldots,N$. Invoking (3) we have $A_{Ni,j}(x) = a(\zeta_j,\zeta_i;x) = \sum_{q=1}^Q \sigma^q(x)\,a^q(\zeta_j,\zeta_i)$, or

$$\underline{A}_N(x) = \sum_{q=1}^Q \sigma^q(x)\underline{A}_N^q,$$

where $A_{Ni,j}^q = a^q(\zeta_j,\zeta_i)$, $i \le i,j \le N$, $1 \le q \le Q$. The off-line/on-line decomposition is now clear. In the *off-line* stage, we compute $u(x^n)$ and form $\underline{A}_N^q$ and $\underline{F}_N^q$: this requires N (expensive) "a" finite element solutions and $\mathcal{O}(QN^2+N)$ finite-element vector inner products. In the *on–line* stage, for any given new x, we form $\underline{A}_N$ from (III-C), solve (6) for $\underline{u}_N(x)$, and finally evaluate $s_N(x) = \underline{F}_N^T\,\underline{u}_N(x)$, which requires $\mathcal{O}(QN^2+N)+\mathcal{O}(\frac{2}{3}N^3)$ operations and $\mathcal{O}(QN^2+N)$ storage. We point out that it is possible to make significant gains on these requirements at the expense of additional bookkeeping.

We evaluate $s_j'(x)$ and $s_j''(x)$ by

$$s_{Nj}'(x) \;=\; \underline{F}_N(x)^T\underline{u}_N'(x) \quad\text{and}\quad s_{Nj}''(x) \;=\; \underline{F}_N(x)^T\underline{u}_N''(x),$$

where $\underline{u}_N(x)$ is determined as above, $\underline{u}_N'(x) \in (\mathbb{R}^N)^P$ and $\underline{u}_N''(x) \in (\mathbb{R}^N)^{P\times P}$ satisfy, respectively, $\underline{A}_N(x)\underline{u}_N'(x) = -\underline{A}_N'(x)\underline{u}_N(x)$, and $\underline{A}_N(x)\underline{u}_N''(x) = -\underline{A}_N''(x)\underline{u}_N(x) - 2\underline{A}_N'(x)\underline{u}_N'(x)$, with $\underline{A}_N'(x) \in (\mathbb{R}^{N\times N})^P$ and $\underline{A}_N''(x) \in (\mathbb{R}^{N\times N})^{P\times P}$ defined by $\underline{A}_N'(x) = \sum_{q=1}^Q \sigma^{q\prime}(x)\,\underline{A}_N^q$ and $\underline{A}_N''(x) = \sum_{q=1}^Q \sigma^{q\prime\prime}(x)\underline{A}_N^q$. Note that we are only required to invert $\underline{A}_N(x)$ as before. It is also evident that the above calculations require no additional data for the off-line database. Computational costs for output gradients and Hessians will therefore increase by factors of P and P^2, respectively, with no significant additional storage requirements. We note that the quantities above have been fully specified: this is due to the fact that we have expressions for $\sigma^q(x)$ and that it is implicitly assumed that $\sigma^{q\prime}(x)$ and $\sigma^{q\prime\prime}(x)$ are easily calculable. In our implementation we have derived these quantities by applying automatic differentiation, though less attractive alternatives exist (analytical expressions, finite-differences).

The crucial point we make is that the marginal cost of evaluating $s_N(x)$, $s_N'(x)$, and $s_N''(x)$ for any given (iterate) x during the optimization procedure is very small: because N is very small, typically $\mathcal{O}(10)$ thanks to the good convergence properties of W^N, and because (6) can be very rapidly assembled and inverted due to the off-line/on-line decomposition. For the problems discussed in this paper, the resulting computational savings relative to standard (well-designed) finite-element approaches are significant — at least $\mathcal{O}(10)$ but often $\mathcal{O}(1000)$ or more.

Finally, we address to the computational approach by which we can efficiently compute $\Delta_N(x)$, $\Delta_N'(x)$, and $\Delta_N''(x)$ in the on-line stage of our procedure. We again exploit the affine parameter dependence, and refer the reader to [10] for motivation of the method. We can express $\hat{e}(x)$ from the previous section as

$$\hat{e}(x) = \frac{1}{\hat{g}(x)}\left(\hat{z}_0 + \sum_{q=1}^Q\sum_{j=1}^N \sigma^q(x)u_{N\,j}(x)\hat{z}_j^q\right), \tag{7}$$

where $\hat{z}_0 \in Y$ satisfies $\hat{a}(\hat{z}_0,v) = \ell(v)$, $\forall\ v \in Y$, and $\hat{z}_j^q \in Y, j = 1,\ldots,N$, $q = 1,\ldots,Q$, satisfies $\hat{a}(\hat{z}_j^q,v) = -a^q(\zeta_j,v)$, $\forall\ v \in Y$. Inserting (7) into our expression for the bound gap, $\Delta_N(x) = 1/2\ \hat{g}(x)\hat{a}(\hat{e}(x),\hat{e}(x))$, we obtain

$$\Delta_N(x) = \frac{1}{2\hat{g}(x)}\left(c_0 + 2\sum_{q=1}^Q\sum_{j=1}^N \sigma^q(x)u_{N\,j}(x)\Lambda_j^q \;+\; \sum_{q=1}^Q\sum_{q'=1}^Q\sum_{j=1}^N\sum_{j'=1}^N \sigma^q(x)\sigma^{q'}(x)u_{N\,j}(x)u_{N\,j'}(x)\Gamma_{jj'}^{qq'}\right)$$

where $c_0 = \hat{a}(\hat{z}_0,\hat{z}_0)$, $\Lambda_j^q = \hat{a}(\hat{z}_0,\hat{z}_j^q)$, and $\Gamma_{jj'}^{qq'} = \hat{a}(\hat{z}_j^q,\hat{z}_{j'}^{q'})$. In addition, we have

$$\Delta_N'(x) = \frac{1}{\hat{g}(x)}\left(\sum_{q=1}^Q\sum_{j=1}^N\left[\sigma^{q\prime}(x)u_{Nj}(x) + \sigma^q(x)u_{Nj}'\right]\Lambda_j^q \;+\; \frac{1}{2}\sum_{q=1}^Q\sum_{q'=1}^Q\sum_{j=1}^N\sum_{j'=1}^N\Big[\sigma^q(x)'\sigma^{q'}(x)u_{Nj}(x)u_{Nj'}(x) + \sigma^q(x)\sigma^{q''}(x)u_{Nj}(x)u_{Nj'}(x) + \sigma^q(x)\sigma^{q'}(x)u_{Nj}'(x)u_{Nj'}(x) + \sigma^q(x)\sigma^{q'}(x)u_{Nj}(x)u_{Nj'}'(x)\Big]\Gamma_{jj'}^{qq'}\right) - \frac{\hat{g}'(x)}{\hat{g}(x)}\Delta_N,$$

and a similar expression for $\Delta''(x)$ which we do not include here for brevity. Quantities for s_N^+, $s_N^{+\prime}$, $s_N^{+\prime\prime}$, s_N^-, $s_N^{-\prime}$, and $s_N^{-\prime\prime}$, follow immediately from the previous definitions.

We again exploit an off-line/on-line decomposition in calculating the error bounds. In the *off–line* stage we compute $\hat{z}_0$ and $\hat{z}_j^q$, $j = 1,\ldots,N$, $q = 1,\ldots,Q$, and then form c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$: this requires $QN+1$ (expensive) "$\hat{a}$" finite element solutions, and $\mathcal{O}(Q^2N^2)$ finite-element-vector inner products. In the *on–line* stage, for any given new x, we evaluate Δ_N: this requires $\mathcal{O}(Q^2N^2)$ operations and $\mathcal{O}(Q^2N^2)$ storage (for c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$). As for the computation of $s_N(x)$, the marginal cost for the computation of $s_N^\pm(x)$ for any given new x is quite

small — in particular, it is *independent* of the dimension of the truth finite element approximation space. We note that once the database has been formed, all of the above information is available as by-products of the on-line computation (u_N, σ^q, and their derivatives) for the zero$^{\text{th}}$ order quantities. As a result, we again observe an increase in computational cost of a factor of P and P^2 for the gradients and Hessians, and no additional storage requirements.

We summarize the two principal strengths of our reduced-basis approach to optimization problems: off-line/on-line decomposition that allows for fast iterate computation that is independent of the degrees of freedom of the original problem; and error-bound calculations that allow for fast, quantifiable estimates that can be used to adjust for required levels of accuracy.

IV. Interior Point Methods and Optimization

A. Optimization with Uncertainty Control

The reduced-basis method allows us to exploit a rigorous and accurate approximation framework and, most importantly, provides a way to sharply estimate introduced errors. For meaningful optimization, error estimates and associated quantities must also be included in the optimization statement. Therefore, before describing the methods employed in optimizing the problem, we restate the problem so that errors due to approximation can be appropriately and rigorously accounted for. In essence, we no longer define the problem in terms of outputs of interest s_j but rather substitute these quantities by their respective lower and upper bounds s_{Nj}^- and s_{Nj}^+: $s_{j_{\min}} \le s_{Nj}^-(x) \le s_j(x) \le s_{Nj}^+(x) \le s_{j_{\max}}$ for $j = 1, \ldots, M$. Or, in terms of $\bar{s}_{Nj}$ and Δ_{Nj}: $\bar{s}_{Nj}(x) - \Delta_{Nj}(x) \ge s_{j_{\min}}$, and $\bar{s}_{Nj}(x) + \Delta_{Nj}(x) \le s_{j_{\max}}$. For the cost, we simply define

$$J_N(x) = \sum_{j=1}^{K} \omega_j \bar{s}_{Nj}(x)$$

(another alternative is to use s_{Nj}^+ for conservatively high cost estimates). Finally we restate the problem with slack variables $p \in \mathbb{R}^M$, $w \in \mathbb{R}^M$, $t \in \mathbb{R}^P$, $g \in \mathbb{R}^P$ as:

$$\text{find} \qquad x^* \;=\; \arg\min_{x \in \mathcal{D}} \; J_N(x)$$

$$\text{subject to} \quad \begin{cases} s_{Nj}^+(x) - s_{j_{\max}} + p = 0, \\ s_{j_{\min}} - s_{Nj}^-(x) + w = 0, \\ x - x_{\max} + t = 0, \\ x_{\min} - x + g = 0, \\ p,\ w,\ t,\ g \ge 0. \end{cases}$$

The advantage of the above formulation is that it is easy to define a point that is feasible in the inequality constraints. Such a point is unlikely to satisfy the equality constraints but, we will show, this is not necessary for solving the system by Interior Point Methods (IPMs). We note that the above is not in the industry-standard MPS format because we have found that difficulties result in the positive definiteness of the SQP approximation for the so-called primal-dual formulation.

B. IPM Subproblem

We first define the strictly-feasible inequality constraint region $\mathcal{F} = \{(x, p, w, t, g) \in (\mathbb{R}^P \times \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^P \times \mathbb{R}^P) \mid p, w, t, g > 0\}$ and denote any $\gamma \in \mathcal{F}$ as an "interior point". Next we define a barrier function $J_N^\mu : \mathcal{F} \to \mathbb{R}$,

$$J_N^\mu(\gamma) = J_N(x) - \mu \left[\sum_{j=1}^{M} (\log p_j + \log w_j) + \sum_{i=1}^{P} (\log t_i + \log g_i) \right],$$

and state the "barrier subproblem"

$$\text{find } (\gamma^\mu) \;=\; \arg\min_{\gamma \in \mathcal{F}} \; J_N^\mu(\gamma)$$

$$\text{subject to} \quad \begin{cases} s_{Nj}^+(x) - s_{j_{\max}} + p = 0, \\ s_{j_{\min}} - s_{Nj}^-(x) + w = 0, \\ x - x_{\max} + t = 0, \\ x_{\min} - x + g = 0, \end{cases}$$

with related Lagrangian $\mathcal{L}_N^\mu(\gamma; q, y, s, z) = J_N^\mu(\gamma) + \lambda^{p^T}(s_{Nj}^+(x) - s_{j_{\max}} + p) + \lambda^{w^T}(s_{j_{\min}} - s_{Nj}^-(x) + w) + \lambda^{t^T}(x - x_{\max} + t) + \lambda^{g^T}(x_{\min} - x + g)$, where $(\lambda^p, \lambda^w, \lambda^t, \lambda^g) \in (\mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^P \times \mathbb{R}^P)$ are Lagrange multipliers.

It can be shown that if $\mathcal{F}$ is a compact set, then $\lim_{\mu \to 0} x^\mu = \hat{x}$, where $\hat{x}$ is some point in the set of stationary points of $J_N(x)$; that is, points that satisfy the related KKT conditions. This is the central argument that motivates IPMs, which produce a decreasing sequence $\{\mu\}^k$ of barrier parameters, and solve (at least approximately) the barrier sub-problem for each μ_k. We call the process of generating solutions for a barrier iterate k the "homotopy" level.

The typical implementation of IPMs takes the above subproblem as a point of departure. These methods work well for many linear and some convex problems because stationarity is necessary and sufficient for optimality in these cases. For nonconvex problems, however, the above is not sufficient: in addition we require that $\nabla_{xx}^2 J_N(x^*)$ be symmetric positive definite in the null-space of gradients of active constraints — a property which is not guaranteed by the above IPMs formulation. Since we are concerned with engineering problems that are typically nonconvex, the above formulation will often produce stationary points which are not necessarily optimal.

Below we propose a procedure to find solutions that satisfy all requirements for optimality for the nonconvex class of problems we are addressing. It is worthwhile to note that even these points are only guaranteed to be *local* optima. Presently, we will assume that local optima are satisfactory for our purposes, and note that improvements in the method can help produce solutions that approach the *global* optimum (a multi-start procedure, for example).

C. IPMs and SQP with Trust Regions

A class of approaches, mostly based on line search techniques, has been proposed for nonconvex problems (see [2], [3], [11], [12] for examples). Many of these produce descent directions for specific merit functions that reflect a modified statement of the KKT system based on matrix factorization. Arbitrary choice must be made in determining appropriate merit functions and factorization schemes, and to date there has been only limited experience in applying these methods.

We have chosen instead to adopt a Sequential Quadratic Programming (SQP) approach, based on [1], to address the issue of nonconvexity. The geometric interpretation of this approach is more intuitive and the implementation of trust regions allows us to intelligently define the step sizes to be taken throughout the algorithm. We are also able to relax the problem further by separately allowing convergence of feasibility ("normal" to constraints) and optimality ("tangential" to constraints) components.

First we define

$$\min_{(d_x,d_p,d_w,d_t,d_g)} \; Q_{\text{SQP}}(d_x,d_p,d_w,d_t,d_g)$$

$$\text{s. t.} \quad \begin{cases} A_q(x_k)^T d_x + d_p + s^+_{Nj}(x_k) - s_{j_{\max}} + p_k = r_q, \\ -A_y(x_k)^T d_x + d_w + s_{j_{\min}} - s^-_{Nj}(x_k) + w_k = r_y, \\ d_x + d_t + x_k - x_{\max} + t_k = r_s, \\ -d_x + d_g + x_{\min} - x_k + g_k = r_z, \\ (d_x,d_p,d_w,d_t,d_g) \in \mathcal{T}_k, \end{cases}$$

as the SQP approximation to the barrier subproblem, where $Q_{\text{SQP}}(d_x,d_p,d_w,d_t,d_g) = \nabla_x J_N(x_k)^T d_x + \frac{1}{2}\, d_x^T \nabla^2_{xx}\mathcal{L}^\mu_N(\xi_k) d_x - \mu e^T(P_k^{-1} + W_k^{-1} + T_k^{-1} + G_k^{-1}) + \frac{1}{2}\,(d_p,d_w,d_t,d_g)^T \Sigma_k (d_p,d_w,d_t,d_g)$, $\xi_k = (\gamma_k;\lambda_k)$, $\gamma_k = (x_k,p_k,w_k,t_k,g_k)$, $\lambda_k = (\lambda^p,\lambda^w,\lambda^t,\lambda^g)$,

$$A_q(x) = \bar{s}'_{Nj}(x) + \Delta'_{Nj}(x),$$
$$A_y(x) = \bar{s}'_{Nj}(x) - \Delta'_{Nj}(x),$$

and $\mathcal{T}_k$ is a trust region to be defined below, and, for example, P_k represents a diagonal matrix whose diagonal is p_k. We have chosen to use $\nabla^2_{xx}\mathcal{L}^\mu_N$ in the definition of Q_{SQP} since this choice is most compatible with the iterative interpretation of the barrier KKT conditions. We then define $\Sigma_k \in \mathbb{R}^{2(M+P)\times 2(M+P)}$ as a diagonal positive definite matrix that approximates the remainder (non-x elements) of the Hessian of the Lagrangian. We implement here a "primal-dual" form of the method:

$$\Sigma_k = \begin{bmatrix} P_k^{-1}\Lambda_k^p & & & \\ & W_k^{-1}\Lambda_k^w & & \\ & & T_k^{-1}\Lambda_k^t & \\ & & & G_k^{-1}\Lambda_k^g \end{bmatrix}. \tag{8}$$

This differs from the typical primal-dual interpretation of IPMs, which owe their name to iterations that include dual variables (i.e. Lagrange multipliers). Here, the method's name stems from the fact that dual variables λ_k are stated as part of the problem in the matrix Σ_k, though we have not yet addressed how to approximate them. Since there are different alternatives to approximate λ_k, we categorize this issue at the implementation level, and address it in Section IV-D.2. We simply note here that it will be advantageous to define λ_k in a way that guarantees positive definiteness of Σ_k.

The SQP constraints best represent a linear approximation of the original problem constraints when $(r_g,r_y,r_s,r_z) = 0$. This may not be possible, however, if the SQP solution is to lie in $\mathcal{T}_k$. The question of how to determine these quantities must therefore be addressed. We decompose the step $d = (d_x,d_p,d_w,d_t,d_g)$ into "normal" and "tangential" components $d^{\text{n}} = (d^{\text{n}}_x,d^{\text{n}}_p,d^{\text{n}}_w,d^{\text{n}}_t,d^{\text{n}}_g)$ and $d^{\text{t}} = (d^{\text{t}}_x,d^{\text{t}}_p,d^{\text{t}}_w,d^{\text{t}}_t,d^{\text{t}}_g)$, where $d = d^{\text{n}} + d^{\text{t}}$. The terminology originates from the fact that d^{n} is a step direction that minimizes the norm of the residuals given the trust region constraints. Finding d^{n} can be stated as the optimization problem:

$$\min_{d^{\text{n}}} \begin{cases} \|A_q(x_k)^T d^{\text{n}}_x + d^{\text{n}}_p + s^+_{Nj}(x_k) - s_{j_{\max}} + p_k\|_2^2 \\ +\| - A_y(x_k)^T d^{\text{n}}_x + d^{\text{n}}_w + s_{j_{\min}} - s^-_{Nj}(x_k) + w_k\|_2^2 \\ +\|d^{\text{n}}_x + d^{\text{n}}_t + x_k - x_{\max} + t_k\|_2^2 \\ +\| - d^{\text{n}}_x + d^{\text{n}}_g + x_{\min} - x_k + g_k\|_2^2 \end{cases}$$
$$\text{subject to} \quad (d^{\text{n}}_x,d^{\text{n}}_p,d^{\text{n}}_w,d^{\text{n}}_t,d^{\text{n}}_g) \in \mathcal{T}^{\text{n}}_k,$$

where $\mathcal{T}^{\text{n}}_k \subset \mathcal{T}_k$. Once d^{n} has been found, we can define

$$\begin{aligned} r_g &= A_q(x_k)^T d^{\text{n}}_x + d^{\text{n}}_p + s^+_{Nj}(x_k) - s_{j_{\max}} + p_k, \\ r_y &= -A_y(x_k)^T d^{\text{n}}_x + d^{\text{n}}_w + s_{j_{\min}} - s^-_{Nj}(x_k) + w_k, \\ r_s &= d^{\text{n}}_x + d^{\text{n}}_t + x_k - x_{\max} + t_k, \\ r_z &= -d^{\text{n}}_x + d^{\text{n}}_g + x_{\min} - x_k + g_k. \end{aligned}$$

which guarantees the existence of total steps in the trust region. We can therefore find d that optimizes the remainder of the now fully-defined SQP subproblem.

For future reference we define the SQP constraint gradient matrix

$$\tilde{A} = \begin{bmatrix} A_q & P & 0 & 0 & 0 \\ -A_y & 0 & W & 0 & 0 \\ I & 0 & 0 & T & 0 \\ -I & 0 & 0 & 0 & G \end{bmatrix}, \tag{9}$$

and note that the "normal" step d^{n} must lie in the range space of $\tilde{A}$ whereas the "tangential" step d^{t} lies in its right null space when using the above residuals.

D. Computational Procedure

D.1 Initial Guess

In order to start any algorithm, it is necessary to determine an *inequality-feasible* initial guess $\xi_0 \in \mathcal{F}$. Our restatement allows us to easily determine this as follows: $x_0 = (x_{\min} + x_{\max})/2$, $p_0 = \max\{\ \theta_0, |s^+_{Nj}(x_0) - s_{j_{\max}}|\ \}$, $w_0 =$

$\max\{\ \theta_0, |s_{j\min} - \bar{s_{Nj}}(x_0)|\ \}$, $t_0 = \max\{\ \theta_0, |x_0 - x_{\max}|\ \}$, $g_0 = \max\{\ \theta_0, |x_{\min} - x_0|\ \}$, $\lambda_0^p = \mu P_0^{-1} e$, $\lambda_0^w = \mu W_0^{-1} e$, $\lambda_0^t = \mu G_0^{-1} e$, $\lambda_0^g = \mu T_0^{-1} e$, where θ_0 is some positive number (we take $\theta_0 = 1$). The above is motivated by the KKT conditions while the chosen value of θ_0 is representative of those found in the literature (see [11]).

D.2 Lagrange Multiplier Approximation

The first unknowns we must address are the Lagrange multipliers λ_k of the barrier subproblem. It can be shown that as $\mu_k \to 0$, λ_k approaches the Lagrange multipliers of the original problem, which are nonnegative. Due to the structure of Σ_k, defined in terms of λ_k and slack variables (which are always positive for feasibility), negative approximations of λ_k will result in potentially non-SPD Hessians of the SQP subproblem. Though this is not necessarily forbidden due to the presence of trust regions, it can nonetheless be counterproductive and degrade convergence.

Following typical SQP procedure, the authors in [1] determine values of λ_k by least squares approximation to the KKT system of the barrier subproblem. Though this will yield a reasonable approximation to these variables, it is often the case that negative values will result. In that paper, the authors revert to the primal formulation for negative Lagrange multipliers to guarantee the SPD property of Σ_k. Here we take a different approach, and revert instead to a traditional primal-dual iterative interpretation for negative values of λ_k. We feel this is a better compromise since it allows us to stay within the primal-dual framework. In summary: we approximate λ_k by least squares:

$$\lambda_k^{\text{LS}} = (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \begin{bmatrix} -\nabla_x J_N \\ \mu e_\lambda \end{bmatrix}, \tag{10}$$

where $e_\lambda = (1, \ldots, 1) \in \mathbb{R}^{2(M+P)}$, then we define λ_k:

$$\lambda_k^i = \begin{cases} \lambda_k^{\text{LS}^i}, & \text{if } \lambda_k^{\text{LS}^i} > 0, \\ \lambda_{k-1}^i + \alpha_\lambda \Delta\lambda_k^i, & \text{if } \lambda_k^{\text{LS}^i} \le 0, \end{cases} \tag{11}$$

for $i = 1, \ldots, 2(M+P)$, where $\Delta\lambda_k$ is determined by a predictor/corrector estimation of traditional primal-dual step directions, and

$$\frac{0.995}{\alpha_\lambda} = \max_i \left\{ 1,\ -\frac{\Delta\lambda_k{}^i}{\lambda_k{}^i} \right\}. \tag{12}$$

Equations (11) and (12) ensure the positivity of iterates λ_k, which in turn ensure the positive definetness of Σ_k. Note that since we are inverting diagonal matrices in the above, we add a negligible computational cost, but guarantees SPD Σ_k and maintain the primal-dual interpretation.

D.3 Definitions of Trust Regions

Trust region constraints, $(d_x, d_p, d_w, d_t, d_g) \in \mathcal{T}_k$ and $(d_x^{\text{n}}, d_p^{\text{n}}, d_w^{\text{n}}, d_t^{\text{n}}, d_g^{\text{n}}) \in \mathcal{T}_k^{\text{n}}$, play a critical role in the algorithm since they provide a mechanism that guarantees that the SQP subproblem is a good approximation to the barrier subproblem. There is flexibility in choosing these regions: one may choose, for example, a ball of radius Δ_T around the current iterate.

It is important, however, to choose an appropriate norm in which to measure these trust constraints. Here we scale the problem with respect to slack variables and arrive at, respectively,

$$\| (d_x, P^{-1} d_p, W^{-1} d_w, T^{-1} d_t, G^{-1} d_g) \|_2 \le \Delta_T,$$
$$P^{-1} d_p, W^{-1} d_w, T^{-1} d_t, G^{-1} d_g \ge -\tau,$$

and

$$\| (d_x^{\text{n}}, P^{-1} d_p^{\text{n}}, W^{-1} d_w^{\text{n}}, T^{-1} d_t^{\text{n}}, G^{-1} d_g^{\text{n}}) \|_2 \le \zeta \Delta_T,$$
$$P^{-1} d_p^{\text{n}}, W^{-1} d_w^{\text{n}}, T^{-1} d_t^{\text{n}}, G^{-1} d_g^{\text{n}} \ge -\tau/2,$$

where $\zeta \in (0,1)$ and $\tau \in (0,1)$ guarantee that $\mathcal{T}_k^{\text{n}}$ is a strict subset of $\mathcal{T}_k$ (we choose $\zeta = 0.8$ and $\tau = 0.995$). Justification for scaling is as follows. Active inequality constraints will necessarily produce slack variables that approach zero near the solution. If the trust region is not scaled appropriately, these variables will impede progress by severely shrinking the set of allowable trust region candidates. The scaling above adequately prevents this situation.

D.4 Calculation of Normal and Tangential Components

Step 1: Calculate a Newton step d^{N} and a steepest descent direction d^{D} for the normal subproblem.

Step 2: Calculate d^{n} by linearly combining d^{N} and d^{D} (for instance, by the dogleg method, see [1]) such that we adhere to all trust region constraints and achieve a reduction in the normal cost.

Step 3: Calculate d by solving the remaining problem in terms of d^{t} and defining residuals (r_g, r_y, r_s, r_z) as described above. Note that for zero d^{t} we immediately adhere to the constraints of the SQP subproblem. Therefore, we choose to use Projected Conjugate Gradients (PCG) with initial guess $d_0^{\text{t}} = 0$ and orthogonally projected iterates to the null space of $\tilde{A}$ by way of operator

$$P = I - \tilde{A}(\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tag{13}$$

(note that $P\tilde{A} = 0$). We note that since we are by definition operating in the null space of $\tilde{A}$, a negative Hessian-norm of a tangential iterate necessarily indicates a direction of negative curvature that will improve the SQP cost and lead to the interior of $\mathcal{F}$. If this is detected we assign this direction to d^{t} and quit the PCG procedure.

D.5 Merit Functions, Predicted and Actual Reductions, and Trust Region Adjustments

Different merit functions can be used to gauge the progress of most IPMs formulations. Here we choose

$$\Psi(\gamma;\nu) = J_N^{\mu}(\gamma) + \nu \left\| \begin{bmatrix} s_{Nj}^{+}(x) - s_{j\max} + p \\ s_{j\min} - s_{Nj}^{-}(x) + w \\ x - x_{\max} + t \\ x_{\min} - x + g \end{bmatrix} \right\|_2$$

where $\nu > 0$ is a penalty parameter — not to be confused with μ. We use Ψ to determine whether the calculated step d will sufficiently improve the approximate solution. There are different ways of approaching this criteria; many IPMs formulations simply attempt to achieve gradient-related status so that a minimization rule, such as Armijo, can be applied. Here we begin by defining the predicted merit function reduction for a step d, $\text{pred}\,(d) = -\nabla_\gamma \Psi(\gamma_k;\nu)^T d \approx \Psi(\gamma_k;\nu) - \Psi(\gamma_k + d;\nu)$, and a predicted reduction in the residual functions (normal to the constraints) as a result of d^{n}, $\text{vpred}(d^{\text{n}})$. We require that ν be large enough to ensure a predicted decrease in Ψ that is positive and proportional to the decrease in the direction normal to constraints. This can be achieved by choosing $\nu \geq \max\{0,\ \text{pred}\,(d)/\,(\rho \text{vpred}\,(d^{\text{n}}))\}$ for a factor $\rho \in (0,1)$ (chosen here to be 0.3).

Given this ν, and that $\text{pred}\,(d) \geq 0$, a step d is deemed acceptable if the actual merit reduction, $\text{ared}(d) = \Psi(\gamma_k;\nu) - \Psi(\gamma_l + d;\mu)$, is positive and at least a small fraction η (we take $\eta = 10^{-8}$) of the predicted step:

$$\gamma_{\text{red}} \ = \ \frac{\text{ared}\,(d)}{\text{pred}\,(d)} \ \geq \ \eta. \tag{14}$$

If the above criterion is not met we conclude that our trust region is too large for an accurate SQP approximation: we reject the step and reduce the trust region Δ_T by a factor in $(0.1, 0.3)$. Otherwise, we can use γ_{red} as a measure of accuracy of the SQP approximation, and adjust Δ_T accordingly:

$$\Delta_T = \begin{cases} \max\{\ 7\|d\|, \Delta_T\ \} & \text{if } \gamma_{\text{red}} > 0.9; \\ \max\{\ 2\|d\|, \Delta_T\ \} & \text{if } 0.3 \leq \gamma_{\text{red}} \leq 0.9; \\ \Delta_T & \text{if } \eta \leq \gamma_{\text{red}} < 0.3. \end{cases}$$

The detriment of choosing a nondifferentiable merit function Ψ becomes apparent here since ν may not be defined at the solution (where constraints are exactly satisfied). However, this is not a significant problem in practice since we never allow μ_k to reach its limit point and expect to approach the constraints close to orthogonally from the interior of $\mathcal{F}$. We plan to experiment in the future with differentiable merit functions (based, for instance, on the $\|\cdot\|_2^2$ metric).

D.6 Barrier Parameter Update

We now address the strategy of updating μ at the homotopy level. We have already stated the need to develop a decreasing sequence $\{\mu\}^k$ for convergence of IPMs. It can be shown from KKT conditions that any *exact* solution of the barrier subproblem satisfies $\mu e = P\lambda^p = W\lambda^w = G\lambda^g = T\lambda^t$ Therefore, a good *estimate* for a non-exact, non-converged solution of the barrier subproblem (that is, the solution to an SQP subproblem) would be $\tilde{\mu}_k \ \approx \ (p^T\lambda^p + w^T\lambda^w + g^T\lambda^g + t^T\lambda^t)\ /\ (2(M+P))$. To assure that we decrease μ_k we set

$$\mu_k \ \leftarrow \ \alpha_\mu \tilde{\mu}_k \tag{15}$$

at the homotopy level, where α_μ can typically be some aggressively small fraction (for example, $\alpha_\mu = 0.1$), and we typically take a *single* SQP step per μ_k. In practice we have found it more useful to take advantage of available information that reflects the degree of progress to dynamically determine this value. For example:

$$\alpha_\mu = \left(\frac{1/\alpha_\lambda - 1}{1/\alpha_\lambda + 10} \right)^2, \tag{16}$$

where α_λ is determined as before. "Good" steps provide $\alpha_\lambda \approx 0.995$ and aggressively small α_μ, whereas "poor" steps provide small α_λ that result in conservative decreases (α_μ close to 1).

D.7 General Algorithm

The following is a high-level summary of the algorithm:

Determine initial interior point $\xi_0 = (\gamma_0, \lambda_0)$.
Determine initial barrier parameter μ_0.
  Loop in k to solve barrier subproblem μ_k for ξ_k:
    1. Approximate Lagrange multipliers λ_k.
    2. Define trust region $\mathcal{T}_k$.
    3. Find normal step to SQP approximation d_k^{n}.
    4. Find tangential step to SQP approximation d_k^{t}.
    5. Set $d_k = d_k^{\text{n}} + d_k^{\text{t}}$.
    6. If step meets merit function criteria:
       a. Set $\gamma_{k+1} = \gamma_k + d_k$.
       b. Maintain/increase trust region diameter Δ_T.
       c. Set $\mu_{k+1} = \alpha_{\mu k}\mu_k$ for some $\alpha_{\mu k} \in (0,1)$.
       Else: decrease trust region diameter Δ_T.
  End loop.

Traditional IPMs adjust the merit function barrier parameter to guarantee that a given Newton iteration produces a descent direction. The effect is to move towards stationarity, but information about curvature of the original problem is lost, and directions that decrease the merit function (rather than the objective cost) dominate the algorithm. For nonconvex problems, the tendency is to move to non-optimal stationary points.

The power in the algorithm proposed above is in the use of trust regions in the SQP context and their relationship to the merit function. The trust regions impose a constraint that guarantees the accuracy of the SQP subproblem. At each iteration, feasibility is improved by decreasing the merit

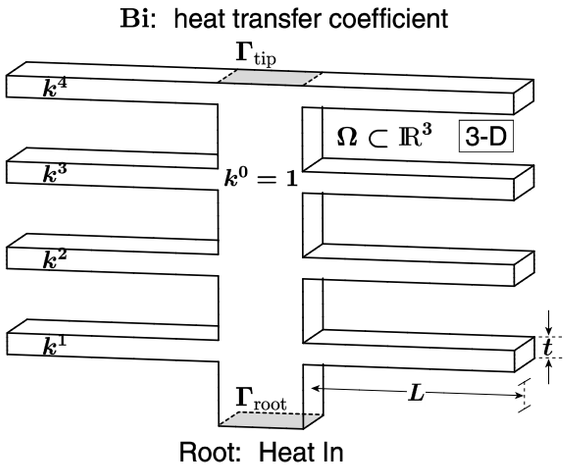

Fig. 1
3D Thermal Fin

function orthogonally to the constraints, while optimality is improved by decreasing the merit function tangentially. If, however, negative curvature is detected in a direction that lies in the tangential subspace, it must be true that the cost can be improved in that direction, regardless of the normal component. By taking this combined step, we have thus allowed for reduction of both merit function and objective cost.

V. Illustration: A Three Dimensional Fin

To illustrate our method, we consider a three-dimensional thermal fin designed to effectively remove heat from a surface. The fin, shown in Figure 1, consists of a vertical central "post" and four horizontal "subfins"; its function is to conduct heat from a prescribed uniform flux "source" at the root, Γ_{root}, through the large-surface-area subfins to surrounding fluid.

The fin is characterized by a two-component parameter vector, or "input," $x = (x_1, x_2)$, where $x_1 = \text{Bi}$ (Biot number) and $x_2 = t$ (thickness of subfins normalized with respect to fin depth); x may take on any value in a specified design space $\mathcal{D} \subset \mathbb{R}^2$. We consider the nondimensional problem, and normalize all lengths with respect to the depth (taken to be unity). The root is square, the hight is 4 times the depth, and the length of subfins is $L = 2.1$.

We consider several outputs of interest. The first is the output $T_{\text{root}} \in \mathbb{R}$, the average temperature of the fin root normalized with respect to the prescribed heat flux (unity) at that surface. This output relates directly to the cooling efficiency of the fin — lower values of T_{root} imply better performance. More importantly, if T_{root} exceeds a maximum value, the fin may fail due to thermal stresses, melting, or yielding. We consider the volume of the fin

$$V = 4 + 8Lt$$

another important metric. For a given density, the weight of the fin will be reflected by this quantity. It is thus reasonable to design a fin which minimizes this output. Finally, we consider Bi itself as an output of interest, since it reflects the power requirements of the design. In order to optimize the design, we must be able to rapidly evaluate $T_{\text{root}}(x)$, $V(x)$, and Bi for a large number of parameter values $x \in \mathcal{D}$.

A. Problem Statement

The steady–state temperature distribution within the fin, $u(\,\cdot\,)$, (i.e. the forward problem) is governed by the elliptic partial differential equation

$$-k\,\nabla^2 u = 0,$$

where ∇^2 is the Laplacian operator and k is the material thermal conductivity. We introduce a Neumann flux boundary condition at the fin root

$$-(\nabla u \cdot \hat{\mathbf{n}}) = -\mathbf{1} \text{ on } \boldsymbol{\Gamma}_{\text{root}},$$

to model the heat source, and a Robin boundary condition

$$-k(\nabla u \cdot \hat{\mathbf{n}}) = \text{Bi}\ \mathbf{u} \text{ on } \boldsymbol{\Gamma}^{\mathbf{i}}_{\text{ext}},\ \mathbf{i} = \mathbf{0}, \ldots, \mathbf{4},$$

to model convective heat losses. Here Γ^i_{ext} is the part of the boundary of Ω^i exposed to the fluid; that is, $\cup_{i=0}^4 \Gamma^i_{\text{ext}} = \partial\Omega \setminus \Gamma_{\text{root}}$.

Solutions of the above system of equations yield the temperature distribution $u(\,\cdot\,; x)$ for any feasible choice of the design parameter-vector x. The outputs of interest are: the average root temperature (a compliant output), the fin volume, and the Biot number. They can be expressed, respectively, as

$$T_{\text{root}}(x) = \ell_{\text{root}}(u(x)) = \int_{\Gamma_{\text{root}}} u(x),$$

$V(x) = 4 + 8L\,x_2$, and $\text{Bi}(x) = x_1$.

Suppose our design calls for minimizing a combination of material volume (reflected by $V(x)$) and power (reflected by $\text{Bi}(x)$), weighted by a ratio of 1:85. Suppose further that a maximum temperature constraint is $T_{\max} = 0.85$, at which point the material fails. Since we expect T_{root} to be the highest temperature (averaged over an area) on the fin, and since this value must be positive, we require that $0 \le T_{\text{root}} \le 0.85$. We also impose constraints on the parameters $x \in [0.1, 0.9] \times [0.2, 0.4]$. The optimization problem can therefore be stated as:

$$\text{find} \qquad x^* = \arg \min_{x \in \mathbb{R}^2} \{\, V(x) + 85\,\text{Bi}(x) \,\}$$

$$\text{subject to} \quad \begin{cases} 0 \le T^-_{\text{root}}(x)\ (\le)\ T^+_{\text{root}}(x) \le 0.85, \\ 0.1 \le x_1 \le 0.9, \\ 0.2 \le x_2 \le 0.4. \end{cases}$$

The statement above is deceptively simple and we will see in the following section that traditional IPMs formulations will not yield acceptable results.

k	μ_k	x_1^μ	x_2^μ	γ_{PD}	Ψ
0	10	0.5	0.3	0.3398	53.371
1	1.309×10^{-1}	0.12	0.2956	0.3625	53.412
3	1.418×10^{-2}	0.2608	0.2946	0.6742	56.830
5	4.259×10^{-1}	0.2934	0.3371	0.0002	40.094
7	3.064×10^{-2}	0.2889	0.3583	0.2481	40.549
10	2.347×10^{-5}	0.3117	0.2432	0.9419	34.457
12	6.491×10^{-10}	0.3124	0.2402	0.95	34.405

TABLE I
BARRIER ITERATIONS FOR PD IPMS FORMULATION. TOTAL CALCULATION TIME = 1.0 SEC.

k	μ_k	x_1^μ	x_2^μ	γ_{red}	Ψ
0	10	0.5245	0.3209	0.9760	143.82
1	1.448×10^{-4}	0.2311	0.3048	0.6732	28.656
3	3.113×10^{-6}	0.2054	0.3151	1.4362	26.526
5	$4.205 \times 10^{-7}(*)$	0.2803	0.3882	0.9207	34.038
7	5.758×10^{-8}	0.2779	0.3986	1.4111	34.000
10	1.106×10^{-9}	0.2776	0.3999	1.5512	33.989
12	9.023×10^{-11}	0.2775	0.4000	1.5741	33.987

TABLE II
BARRIER ITERATIONS FOR SQP IPMS FORMULATION. TOTAL CALCULATION TIME = 1.7 SEC. (*) = DETECTED NON-POSITIVE DEFINITE TANGENTIAL HESSIAN.

B. Computational Results

Before presenting results of our SQP IPMs formulation, we solve the above problem with reduced-basis techniques in a traditional primal-dual IPMs formulation. We call these "PD results" (see [2], [3], [11] for primal-dual IPMs formulations).

First, we note that to accurately carry out the above calculation we must first build a reduced-basis space W^N that appropriately represents the domain in question. We choose here $\mathcal{D} = [0,1]\times[0.1,0.5]$, for example, so that the constraints on x lie in $\mathcal{D}$. It should be noted that even if there were no upper or lower bounds imposed on x in the original problem, these would have to be appropriately introduced since the reduced-basis approximation can only be trusted for points in $\mathcal{D}$. For the following calculations, we choose $N = 40$ random points in $\mathcal{D}$.

Table I shows the results of the calculation. A single Newton step is taken per homotopy level, and the merit function is similarly defined to before (with the notable exception that ν is chosen to guarantee gradient related Ψ given the Newton step d, i.e. $\nabla_\gamma \Psi^T d < 0$). The quantity γ_{PD} represents the size of the Newton step that is taken to guarantee feasibility, and iterations are stopped when the KKT conditions are satisfied. For a fair comparison with our approach, μ_k was determined from expressions (15) and (16).

We see from the table that $\hat{x} = (0.3124, 0.2402)$, and thus neither of the parameter inequality constraints is active. At the solution point $T^-_{\text{root}}(\hat{x}) = 0.84953$ and $T^+_{\text{root}}(\hat{x}) = 0.85$, so we observe that the upper temperature constraint is active. More importantly, we note that by including the error estimation (and respective derivatives) in the problem statement and solution, we have achieved rigorous feasibility for the *true* result. That is: we know that the true value of $T_{\text{root}}(\hat{x})$ lies in $[0.84953; 0.85]$, and is thus feasible, $T_{\min} \le T^-_{\text{root}}(\hat{x}) \le T_{\text{root}}(\hat{x}) \le T^+_{\text{root}}(\hat{x}) = T_{\max}$, but this did not hinder the convergence of the approximate solution or the significant computational savings of reduced-basis (the entire "on-line" computation of this PDE optimization problem takes 1.0 seconds on a 1.60GHz laptop).

Unfortunately, this solution is not optimal. Though the point satisfies the KKT conditions, this is only a necessary condition for optimality for the current nonconvex problem. We can check the optimality of the solution by considering the Hessian of the cost in the null-space of the active constraints. Here we can simply check the following: consider an arbitrarily small δx that will adhere to (approximately active) feasibility

$$T^+(\hat{x}) + T^{+\prime}(\hat{x})^T \delta x = T_{\max}$$

(by staying in the null-space of $T^{+\prime}(\hat{x})$), and consider the cost difference

$$J_N(\hat{x} + \delta x) - J(\hat{x}).$$

Here we have $T^{+\prime}(\hat{x}) = (-1.0758, -0.2009)^T$, with null-space $\delta x = (-0.1835, 0.9830)$. The upper-bound for the temperature at the point $(\hat{x} + 10^{-5}\delta x)$ is $T^+(\hat{x} + 10^{-5}\delta x) = 0.8499$ (and therefore the true temperature is certainly feasible). But $J_N(\hat{x} + 10^{-5}\delta x) - J_N(\hat{x}) = -5.5560 \times 10^{-7} < 0$: the solution $\hat{x}$ cannot be optimal.

Table II presents the results obtained from the SQP framework. Note that there was a modest increase in computational time to 1.7 seconds on the same machine, mainly due to the extra amount of bookkeeping required (since this is independent of the size of the PDE system, we expect a lesser increase for larger problems). The results are based on single SQP step for each homotopy level, analogously to the previous PD calculations. We note that the general behavior of the method is similar, but at iteration 5 we detect a non-positive definite tangential Hessian. At this critical point we are able to move away from stationary but non-optimal points, and the final result can be shown to be optimal.

	Bi	t	T^-_{root}	T^+_{root}	V	J_N
PD	0.312	0.240	0.8495	0.85	7.844	**34.41**
SQP	0.278	0.400	0.8490	0.85	10.40	**33.99**

TABLE III
COMPARISON OF RESULTS AT PD AND SQP SOLUTIONS.

We finally point out Table III for a comparison of the results. We see that, as expected, the SQP approach produces a lower cost at its solution than the PD approach. It is interesting to note, however, that the optimum fin has a significantly higher volume — offset by the lower Biot number in the cost.

Acknowledgements

We would like to thank Dr. Christophe Prud'homme, Karen Veroy, and Yuri Solodukhov of MIT for helpful discussions concerning the reduced-basis Method, and Prof. Robert Freund of MIT and Shidrati Ali of the Singapore-MIT Alliance for discussion, collaboration, and experimentation with Interior-Point Methods. This work was supported by the Singapore-MIT Alliance, by DARPA and ONR under Grant N00014-01-1-0523 (subcontract 340-6218-3), and by DARPA and AFOSR under Grant F49620-01-1-0458.

References

[1] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 2000.

[2] A. Forsgren and P. E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM J. Optim.*, 8(4):1132–1152, 1998.

[3] P. E. Gill, W. Murray, D. B. Ponceleon, and M. A. Saunders. Primal-dual methods for linear programming. *Mathematical Programming*, 70:251–277, 1995.

[4] L. Machiels, Y. Maday, I. B. Oliveira, A.T. Patera, and D.V. Rovas. Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *C. R. Acad. Sci. Paris, Série I*, 331(2):153–158, July 2000.

[5] L. Machiels, Y. Maday, and A. T. Patera. Output bounds for reduced-order approximations of elliptic partial differential equations. *Comp. Meth. Appl. Mech. Engrg.*, 190(26-27):3413–3426, 2001.

[6] Y. Maday, L. Machiels, A. T. Patera, and D. V. Rovas. Blackbox reduced-basis output bound methods for shape optimization. In *Proceedings* 12^{th} *International Domain Decomposition Conference*, pages 429–436, Chiba, Japan, 2000.

[7] M. Paraschivoiu, J. Peraire, Y. Maday, and A. T. Patera. Fast bounds for outputs of partial differential equations. In J. Borgaard, J. Burns, E. Cliff, and S. Schreck, editors, *Computational methods for optimal design and control*, pages 323–360. Birkhäuser, 1998.

[8] C. Prud'homme, D. Rovas, K. Veroy, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bounds methods. *Journal of Fluids Engineering*, 124(1):70–80, March 2002.

[9] C. Prud'homme, D. Rovas, K. Veroy, and A.T. Patera. Mathematical and computational framework for reliable real–time solution of parametrized partial differential equations. *M2AN*, 2002. Submitted.

[10] C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici. Reduced–basis output bound methods for parametrized partial differential equations. In *Proceedings SMA Symposium*, January 2002.

[11] R. J. Vanderbei. Loqo: an interior point code for quadratic programming. Technical Report SOR-94-15, Statistics and Operations Reseach, Princeton University, September 1998.

[12] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. Technical Report SOR-97-21, Statistics and Operations Reseach, Princeton University.