Author version

Published as:

Joshy, A. J., Zerez, J., & Hwang, J. T. (2026). Enhancing robustness and efficiency in large-scale system design optimization using hessian-vector products. In ASME 2026 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE 2026). https://lsdolab.github.io/lsdo-publications/publication/joshy2026enhancing/

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/joshy2026enhancing/

```
@inproceedings{joshy2026enhancing,
  author = {Anugrah Jo Joshy and Jonathan Zerez and John T. Hwang},
  title = {Enhancing Robustness and Efficiency in Large-Scale System Design Optimization
    Using Hessian-Vector Products},
  booktitle = {ASME 2026 International Design Engineering Technical Conferences and
    Computers and Information in Engineering Conference (IDETC/CIE 2026)},
  year = {2026}
}
```

Proceedings of the ASME 2026
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE2026
August 23–26, 2026, Houston, TX

DETC2026-XXXX

ENHANCING ROBUSTNESS AND EFFICIENCY IN LARGE-SCALE SYSTEM DESIGN OPTIMIZATION USING HESSIAN-VECTOR PRODUCTS

Anugrah Jo Joshy[1], *, Jonathan Zerez[1], John T. Hwang[1]

[1]Department of Mechanical and Aerospace Engineering, University of California San Diego, La Jolla, CA

ABSTRACT

Large-scale system design optimization applies numerical optimization techniques to solve complex, multidisciplinary engineering design problems involving hundreds or more design variables. Among these techniques, quasi-Newton algorithms are a popular choice in practice, as forming exact second-order derivatives is often too expensive. However, their performance is sensitive to problem formulation and model characteristics, requiring careful tuning of initial guesses and scaling strategies to achieve reliable convergence in challenging problems. In particular, these algorithms struggle in highly nonlinear, poorly scaled, discontinuous, or high-dimensional problems, which commonly arise in large-scale system design. To mitigate this challenge, we investigate optimization algorithms that exploit Hessian-vector products (HVPs) to provide accurate curvature information without forming full Hessians, with the goal of improving robustness and efficiency. In many applications, Hessian-vector products can be computed at a cost comparable to that of gradient evaluations, making them particularly attractive in large-scale settings. As an initial study, we propose two Hessian approximation strategies that make use of arbitrary numbers of HVPs, and apply them to test problems. First, we propose a modification to the BFGS algorithm that repeatedly applies the BFGS update formula itself in-place, once for each HVP. Second, we propose a direct-fit strategy that solves a numerical optimization problem over symmetric positive definite matrices using distance-weighted HVPs. The in-place BFGS update approach is evaluated on a broad range of constrained and unconstrained problems from the CUTEst test collection and a high-dimensional cantilever beam design problem. Both methods are tested on the Rosenbrock optimization problem with varying numbers of HVPs. Results with both methods show accelerated optimization convergence compared to standard quasi-Newton, approaching the performance of Newton's method with an exact, convexified Hessian with far lower computational cost.

*Corresponding author: ajoshy@ucsd.edu

Keywords: design optimization, multidisciplinary design optimization, large-scale optimization, optimization algorithm, quasi-Newton, BFGS, second-order optimization

1. INTRODUCTION

Design optimization is a powerful engineering tool that enables principled improvements in system performance using computational models and optimization algorithms. It is typically formulated as a numerical optimization problem that maximizes or minimizes an objective with respect to a set of design variables subject to constraints. By formalizing the search for optimal solutions, optimization reduces development effort and helps identify high-performing designs, particularly in problems that are too complex or unintuitive for manual exploration.

Large-scale system design optimization (LSDO) addresses a particularly challenging class of such problems, characterized by hundreds or more design variables and complex, multidisciplinary system models. The resulting optimization problems require navigating trade-offs among multiple coupled disciplines while satisfying many competing constraints in high-dimensional design spaces. As engineering systems grow in scale and complexity, LSDO is becoming an increasingly important tool for efficient design exploration and decision-making.

LSDO has become increasingly feasible in recent years due to advances in automatic differentiation (AD) and the emergence of modeling and optimization frameworks [1, 2], which together enable the construction of complex models with efficiently computed derivatives. However, feasibility does not necessarily translate to reliability or efficiency. In practice, many problems still require substantial manual tuning to achieve consistent convergence, increasing development effort and prolonging design cycles. This challenge often arises from poor scaling and ill-conditioning in high-dimensional design spaces defined by heterogeneous variables and tightly coupled multidisciplinary analyses. Although exact second-order information can help alleviate these issues, forming and using full Hessians is often prohibitively expensive in this setting.

For this reason, quasi-Newton methods are often the practical default for LSDO. Rather than forming exact Hessians, these methods build Hessian approximations using gradient differences accumulated over the optimization iterations. They are computationally efficient and effective across a broad range of problems, and they naturally support limited-memory formulations and update schemes that preserve positive definiteness of the Hessian approximation.

The recent integration of the high-performance AD library JAX [3] into LSDO frameworks such as CSDL [2] and OpenMDAO [1] has substantially improved access to second-derivative information. While forming and storing full Hessians remains impractical for many LSDO problems, this integration enables efficient computation of Hessian-vector products (HVPs), which provide exact curvature information at much lower cost. Existing second-order methods commonly exploit HVPs within inexact Newton or Newton–Krylov frameworks [4, 5], primarily to avoid the cost of computing and storing full Hessians. Nevertheless, they remain fundamentally Newton-type methods, employing HVPs to solve linear systems involving the exact Hessian. The availability of efficient HVPs therefore presents an opportunity to enhance quasi-Newton methods by injecting exact curvature information without incurring the prohibitive cost of full Hessians. Although prior work on incorporating multiple curvature conditions into quasi-Newton updates, such as block BFGS [6], suggests the potential for improved Hessian approximations, these ideas have seen limited adoption in practical optimization workflows.

Bridging this gap between inexpensive quasi-Newton methods and more costly second-order approaches is the focus of this work. In this paper, we investigate how Hessian-vector products can be utilized effectively within quasi-Newton methods for LSDO, specifically through two methods. First, we propose a modification of the BFGS algorithm that replaces the secant condition associated with the step direction with one or more "tangent" conditions. These tangent conditions enforce exact curvature information along selected directions using Hessian-vector products. We also introduce a distance-weighted Hessian approximation strategy that formulates and solves a numerical optimization problem to find a symmetric positive definite matrix that fits the distance-weighted HVPs.

We present numerical results for the HVP-enhanced BFGS method on a large set of constrained and unconstrained CUTEst [7] problems, together with additional studies on the Rosenbrock problem and a cantilever beam design problem. These studies evaluate whether incorporating additional HVP-based curvature information can improve robustness and efficiency of standard BFGS. We also present preliminary studies on the distance-weighted Hessian approximation strategy to highlight its potential as an alternative Hessian model.

The remainder of the paper is organized as follows. Section 2 reviews relevant background in optimization and modeling. Section 3 presents the HVP-enhanced BFGS method and the distance-weighted Hessian approximation strategy. Section 4 reports numerical results for the proposed approaches. Section 5 concludes with a summary of the key findings and final remarks.

2. BACKGROUND

2.1. Problem Statement

In LSDO applications, the workflow consists of a general-purpose optimization algorithm, or optimizer, and a computational model. The optimizer iteratively evaluates the model at different design variable values to obtain the objective and constraint function values and, in some cases, their derivatives. Because the multidisciplinary nature of LSDO problems often prevents assuming special structure, such as convexity, these problems are generally solved as nonlinear programming (NLP) problems.

We consider nonlinear programming problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) \geq 0, \tag{1}$$

where $x \in \mathbb{R}^n$ represents the design variable vector, $f : \mathbb{R}^n \to \mathbb{R}$ represents the scalar objective function, and $c : \mathbb{R}^n \to \mathbb{R}^m$ represents the vector-valued constraint function. We assume that f and c are twice continuously differentiable with respect to x. Note that equality constraints can be represented as pairs of inequalities, and general inequality constraints can be shifted so that all constraints have zero lower bounds, as in Eq. (1).

2.2. Modeling

Design optimization practitioners first formulate their problems in the form of Eq. (1). They then implement the objective and constraint functions within a modeling framework and couple the resulting model with an optimizer to solve the problem. Advanced modeling frameworks such as OpenMDAO and CSDL enable the modeling of large-scale and complex multidisciplinary systems by allowing the construction of larger models through the modular composition of smaller component models.

Many optimization models are also built within algebraic modeling languages (AMLs), such as AMPL [8], GAMS [9], Pyomo [10], and JuMP [11]. Algebraic modeling languages allow users to define optimization models using simple algebraic expressions. They often provide automatic differentiation capabilities, reducing the need for users to implement derivatives manually. CasADi [12] is a widely used modeling tool for engineering optimization with built-in support for algorithmic differentiation.

2.3. State of the Art in Nonlinear Programming

Although numerous gradient-free and gradient-based algorithms have been developed for solving NLP problems, gradient-based algorithms are generally preferred when derivatives are available at a reasonable cost. Because the number of model evaluations required by gradient-free methods often increases rapidly with the number of design variables, gradient-based algorithms can be orders of magnitude faster for large-scale NLP problems and are often the only practical choice [13].

Among gradient-based algorithms, sequential quadratic programming (SQP) and interior point (IP) methods are the most effective approaches for smooth constrained NLP, particularly in large-scale settings. SQP methods solve the NLP problem (1) by successively solving a sequence of quadratic programming (QP) subproblems that locally approximate the original problem. Interior point methods, on the other hand, solve a sequence of barrier subproblems that approximate the original constrained problem

by enforcing inequality constraints through barrier terms, often after introducing slack variables.

SNOPT [14] and IPOPT [15] are respectively the most widely used SQP and IP optimizers for large-scale nonlinear programming. SNOPT is a commercial software package that has been successfully applied to problems with over 40,000 variables and constraints. IPOPT is an open-source solver designed to handle large-scale problems with up to millions of variables and constraints. SLSQP [16] is a widely used SQP solver, largely because open-source implementations are available through SciPy and several other scientific computing packages.

2.4. Sequential Quadratic Programming

We focus on sequential quadratic programming in this paper, as our numerical experiments in Sect. 4 employ an SQP algorithm. Although the proposed Hessian approximation strategies are also applicable to interior point algorithms, their performance in that setting is not studied in this paper.

We briefly review the notation associated with the SQP formulation considered in this work. The Lagrangian function for the NLP problem (1) is defined as $\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$, where $\lambda \in \mathbb{R}^m$ denotes the Lagrange multipliers corresponding to the constraints $c(x) \geq 0$. We use $g : \mathbb{R}^n \to \mathbb{R}^n$ to denote the gradient of the objective function, $J : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ to denote the Jacobian of the constraint function, and $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n\times n}$ to denote the Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}(x, \lambda)$.

Let (x_k, λ_k) denote the iterates for the design variables and Lagrange multipliers after iteration k. At iteration $k + 1$, an SQP algorithm solves the quadratic programming (QP) subproblem

$$\begin{aligned} \underset{p_k\in\mathbb{R}^n}{\text{minimize}} \quad & f(x_k) + g(x_k)^T p_k + \frac{1}{2} p_k^T B_k p_k \\ \text{subject to} \quad & c(x_k) + J(x_k)p_k \geq 0, \end{aligned} \tag{2}$$

to obtain the search direction p_k and the QP multiplier estimate $\hat{\lambda}_k$. The QP objective uses a quadratic model of the Lagrangian, while the constraints are linear approximations of the NLP constraints. Since exact Lagrangian Hessians may lead to nonconvex QP subproblems, SQP methods often employ positive definite Hessian approximations B_k in place of $H(x_k, \lambda_k)$. When exact Hessians are unavailable or expensive to compute, these approximations are commonly constructed using quasi-Newton updates.

Line search or trust region methods are typically used to ensure global convergence from arbitrary starting points. The algorithm used in our numerical studies employs a line search method to obtain the step size α_k and computes the new iterates as $x_{k+1} = x_k + \alpha_k p_k$ and $\lambda_{k+1} = \lambda_k + \alpha_k(\widehat{\lambda}_k - \lambda_k)$.

2.5. Quasi-Newton Hessian Approximations

Quasi-Newton methods use first-derivative information to build explicit approximations of the Hessian matrix. These approximations are defined recursively through additive updates to the approximation from the previous iteration. Classical quasi-Newton updates enforce the quasi-Newton condition, which requires the updated Hessian approximation B_{k+1} to match the observed change in gradient along the most recent step $s_k = x_{k+1} - x_k$. For constrained optimization, this condition is commonly written using the Lagrangian gradient difference $y_k = \nabla_x\mathcal{L}(x_{k+1}, \bar{\lambda}) - \nabla_x\mathcal{L}(x_k, \bar{\lambda})$, where $\bar{\lambda}$ is an estimate of the Lagrange multipliers. The quasi-Newton condition is then given by $B_{k+1}s_k = y_k$.

Different quasi-Newton updates are obtained by enforcing additional conditions such as symmetry, positive definiteness, and minimal change from the previous approximation with respect to a chosen matrix norm. The BFGS update [17–20] can be interpreted as the solution of an optimization problem that minimizes a weighted Frobenius norm of the change in the inverse Hessian approximation, subject to symmetry and the quasi-Newton condition. The related Davidon–Fletcher–Powell (DFP) method [21–23] can be interpreted similarly, except that the objective minimized is the change in the Hessian approximation itself rather than its inverse.

Since quasi-Newton methods use recursive updates, an initial Hessian approximation must be specified. Typically, a diagonal matrix or an identity matrix is used as the initial approximation B_0. If B_k and B_{k+1} denote the current and updated Hessian approximations, respectively, a general rank-one update satisfying the quasi-Newton condition is given by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)u_k^T}{u_k^T s_k}, \tag{3}$$

where $u_k \in \mathbb{R}^n$ is any vector such that $u_k^T s_k \neq 0$. The symmetric version sets $u_k = y_k - B_k s_k$ and is known as the symmetric rank-one (SR1) update [21, 22]. The Broyden update [24] instead sets $u_k = s_k$.

The most commonly used quasi-Newton method is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [17–20], a rank-two update given by

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k}. \tag{4}$$

Among quasi-Newton updates, BFGS is widely regarded as the most successful, as evidenced by several studies [14, 25]. It preserves symmetry and maintains positive definiteness when $y_k^T s_k > 0$. Positive definiteness of Hessian approximations is important in line-search algorithms because it helps ensure that the computed step is a descent direction for the local model or merit function.

The standard BFGS update enforces curvature consistency only along the most recent step direction through the quasi-Newton condition. When Hessian-vector products are available, exact directional curvature information can be obtained along selected directions without forming the full Hessian. This observation motivates the HVP-enhanced BFGS method and distance-weighted Hessian approximation strategy developed in this work.

3. METHODOLOGY

Hessian-vector products (HVPs) are used in many existing optimization methods to access second-order information without explicitly forming or storing a full Hessian matrix. In most such methods, however, the directions along which HVPs are evaluated are dictated by the linear-algebra subproblem arising within

the optimizer, such as a conjugate-gradient iteration, a Krylov solve, or a truncated Newton step. Representative examples include conjugate-gradient methods that approximate HVPs using finite-difference directional derivatives of the gradient [26], inexact Newton methods that employ inexact Krylov solves [27], and truncated Newton methods based on exact HVP evaluations [28]. In these approaches, HVPs are primarily used to compute or approximate Newton-type search directions rather than to explicitly construct Hessian approximations.

In contrast, the methods presented in this section uses HVPs to construct quasi-Newton-inspired Hessian approximations. Rather than evaluating curvature only along directions generated internally by a Newton or Krylov solve, we use selected directions to introduce exact second-order information directly into Hessian approximations. This enables the use of HVPs within quasi-Newton updates or related Hessian models while avoiding the cost of computing the full Hessian matrix.

3.1. Notation

Let m_k be the number of HVPs computed at optimization iteration k. For each $i = 1, \ldots, m_k$, let $v_{k,i} \in \mathbb{R}^n$ denote the HVP input vector, and let $q_{k,i} \in \mathbb{R}^n$ denote the corresponding HVP output vector. For constrained problems, the HVP is then given by $q_{k,i} = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)v_{k,i}$, and for unconstrained problems, this reduces to $q_{k,i} = \nabla^2 f(x_k)v_{k,i}$.

3.2. HVP-enhanced BFGS Updates

This section presents a modified BFGS method that uses exact curvature information along one or more directions obtained from Hessian-vector products to update the approximate Hessian.

3.2.1. Single-Direction Curvature Update with BFGS We first consider the simplest version of the proposed approach, in which one HVP is incorporated per optimization iteration. In this method, $m_k = 1$ for all k, and the HVP input vector is chosen as the most recent step, $v_{k,1} = s_k = x_{k+1} - x_k$. The corresponding Hessian-vector product is $q_{k,1} = \nabla^2_{xx}\mathcal{L}(x_{k+1}, \lambda_{k+1})v_{k,1}$. The pair $(v_{k,1}, q_{k,1})$ is then used in place of the standard BFGS pair (s_k, y_k) to update the Hessian approximation.

Compared with standard BFGS, the key difference is that the curvature vector y_k is not formed from a gradient difference. Instead, it is obtained directly from the Hessian action at the updated iterate, so that the update incorporates exact local curvature information along the step direction. The method therefore preserves the structure of BFGS while replacing the standard secant condition $B_{k+1}s_k = y_k$ with an HVP-based tangent condition $B_{k+1}s_k = \nabla^2_{xx}\mathcal{L}(x_{k+1}, \lambda_{k+1})s_k$.

Although the same idea can be extended to multiple directions within a single iteration (as described in Sect. 3.2.3), the single-direction case provides a simple setting for assessing whether exact curvature information can improve the robustness and efficiency of standard BFGS.

3.2.2. Maintaining Positive Definiteness A standard BFGS update preserves positive definiteness of B_k only when the curvature condition $y_k^T s_k > 0$ is satisfied. In the HVP-enhanced update, the same requirement applies after replacing the standard gradient-difference vector y_k with the HVP-based curvature vector. However, this condition may be marginally satisfied or violated in practice, particularly when the Lagrangian Hessian is indefinite or when the step direction is poorly aligned with positive curvature. If such curvature information is used directly, the BFGS update may lose positive definiteness, which can degrade the quality of the search direction in subsequent QP subproblems.

To address this issue, we employ the damping strategy proposed by Powell [29]. In this approach, the curvature vector y_k in (4) is replaced by the damped vector

$$\widehat{y}_k = \theta_k y_k + (1 - \theta_k)B_k s_k, \tag{5}$$

where $\theta_k \in [0, 1]$ is defined as

$$\theta_k = \begin{cases} 1, & \text{if } y_k^T s_k \geq 0.2\, s_k^T B_k s_k, \\ \dfrac{0.8\, s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{otherwise.} \end{cases} \tag{6}$$

The BFGS update is then applied using $\widehat{y}_k$ in place of y_k. This choice ensures that $\widehat{y}_k^T s_k \geq 0.2\, s_k^T B_k s_k$, so that the curvature condition required for maintaining positive definiteness is satisfied when B_k is positive definite and $s_k \neq 0$.

Qualitatively, damping modifies unfavorable curvature information rather than discarding it. When the raw curvature vector satisfies the curvature condition, the update is unchanged. When the condition is weakly satisfied or violated, the damped vector blends the new curvature information with the curvature vector predicted by the current Hessian approximation, $B_k s_k$. Thus, damping preserves the structure of the BFGS update while preventing a single unfavorable curvature vector from compromising positive definiteness.

3.2.3. Multi-Direction Curvature Update with BFGS The single-direction update is useful in isolating the effect of introducing exact curvature information into an otherwise standard BFGS framework. More general variants are also possible. In particular, multiple HVPs could be incorporated sequentially through repeated BFGS updates, or simultaneously through block quasi-Newton constructions such as block-BFGS methods [6]. These extensions can improve the quality of the Hessian approximation by enforcing exact curvature information along more than one direction per iteration.

Here we consider a sequential multi-direction strategy. The first direction is always chosen as the most recent step, $v_{k,1} = s_k = x_{k+1} - x_k$. The corresponding pair $(v_{k,1}, q_{k,1})$ recovers the single-direction update described in Sect. 3.2.1. Additional directions are then selected to introduce curvature information along directions not already represented by the recent step.

Given the HVP pairs $\{(v_{k,i}, q_{k,i})\}_{i=1}^{m_k}$, the Hessian approximation is updated sequentially. Let $B_k^{(0)} = B_k$. For each $i = 1, \ldots, m_k$, a BFGS update is applied using the pair $(v_{k,i}, q_{k,i})$:

$$B_k^{(i)} = B_k^{(i-1)} + \frac{q_{k,i}q_{k,i}^T}{q_{k,i}^T v_{k,i}} - \frac{\left(B_k^{(i-1)}v_{k,i}\right)\left(B_k^{(i-1)}v_{k,i}\right)^T}{v_{k,i}^T B_k^{(i-1)} v_{k,i}}. \tag{7}$$

The final Hessian approximation is then $B_{k+1} = B_k^{(m_k)}$.

As in the single-direction case, damping can be applied to each curvature pair before the corresponding BFGS update. Specifically, if the curvature condition is weak or violated, $q_{k,i}$ is replaced by a damped vector

$$\widehat{q}_{k,i} = \theta_{k,i} q_{k,i} + (1 - \theta_{k,i}) B_k^{(i-1)} v_{k,i}, \tag{8}$$

and the BFGS update is applied using $(v_{k,i}, \widehat{q}_{k,i})$ instead of $(v_{k,i}, q_{k,i})$. This safeguard helps preserve positive definiteness of the Hessian approximation during the sequence of updates.

The main question in the multi-direction update is how to choose the directions along which the new HVPs should be computed. The effectiveness of the update depends strongly on this choice, since the selected directions determine what additional curvature information is incorporated into the current approximation $B_k^{(1)}$. We consider three representative strategies.

The first strategy uses the history of recent steps:

$$v_{k,i} = s_{k+1-i}, \qquad i = 1, \ldots, m_k. \tag{9}$$

Thus, the first direction is the current step s_k, while subsequent directions use previous steps $s_{k-1}, s_{k-2}, \ldots$. This approach reuses directions naturally generated by the optimization trajectory and can incorporate curvature information associated with recent motion through the design space.

The second strategy uses Krylov directions generated from the current objective or Lagrangian gradient. With $v_{k,1} = s_k$, the remaining directions are chosen as

$$v_{k,i} = H_k^{i-2} g_k^L, \qquad i = 2, \ldots, m_k, \tag{10}$$

where $H_k = \nabla^2_{xx} \mathcal{L}(x_{k+1}, \lambda_{k+1})$ and $g_k^L = \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1})$. Equivalently, these directions belong to the Krylov sequence $g_k^L,\ H_k g_k^L,\ H_k^2 g_k^L,\ \ldots$. This choice targets curvature information in directions related to the local first-order optimality residual.

The third strategy uses Krylov directions generated from the recent step. In this case,

$$v_{k,i} = H_k^{i-1} s_k, \qquad i = 1, \ldots, m_k, \tag{11}$$

so that the direction sequence is $s_k,\ H_k s_k,\ H_k^2 s_k,\ \ldots$. This approach expands the curvature information around the actual step taken by the optimizer, rather than around the gradient direction.

For all three strategies, the selected HVP input directions are orthogonalized before they are used in the sequential BFGS updates. This prevents the algorithm from repeatedly replacing curvature information along directions that have already been incorporated.

For the Krylov-based strategies, the generation of new directions is terminated if the orthogonalized candidate direction becomes too small. In particular, if $\|v_{k,i}\|_\infty < \epsilon_{\text{mach}}$, the candidate direction is considered numerically unreliable, and no further Krylov directions are added at iteration k. This stopping criterion avoids using directions that are nearly linearly dependent on the directions already incorporated, or that have been degraded by roundoff error. Similarly, for the strategy using the recent step history, orthogonalized candidate directions below this threshold are discarded.

3.3. Distance-Weighted Hessian Approximation Method

This section introduces a second strategy for building Hessian approximations from a limited number of HVPs, called *HVP-Informed Positive-definite Hessian Approximation*, or HIPHA. HIPHA parametrizes a low-rank-plus-diagonal family of symmetric positive-definite matrices and solves a numerical optimization to identify the one that best agrees with distance-weighted HVP information from the current and all previous optimization iterations.

HIPHA assumes a form $B_{k+1} = \alpha I + Z_k Z_k^T$, where $\alpha > 0$ is a prescribed scalar and $Z_k \in \mathbb{R}^{n \times r}$ with $r \leq n$. This definition ensures that the approximate Hessian B_{k+1} is symmetric positive definite. At iteration $k+1$, HIPHA computes Z_k by solving

$$\begin{aligned} \min_{Z_k \in \mathbb{R}^{n \times r}} & \frac{\beta}{2} \left\| \alpha I + Z_k Z_k^T - B_k \right\|_F^2 \\ & + \frac{1}{2} \sum_{(j,i) \in \mathcal{J}_k} w_{kj} \left\| \left(\alpha I + Z_k Z_k^T \right) v_{j,i} - q_{j,i} \right\|_2^2, \end{aligned} \tag{12}$$

where $\mathcal{J}_k = \{(j,i) | 1 \leq j \leq k+1, 1 \leq i \leq m_j\}$. The distance weight w_{kj} is defined by

$$w_{kj} = 2^{-\frac{d_{kj}}{d_{1/2}}},\ d_{kj} = ||D^{-1}(x_{k+1} - x_j)||_2,\ D = \text{diag}(u - l), \tag{13}$$

where u and l are the upper and lower bound vectors for x, D is a diagonal matrix for design-variable scaling, and $d_{1/2}$ is a characteristic length-scale parameter, representing the normalized distance at which the weight decays to one-half. If finite bounds are unavailable, D can instead be constructed using reasonable estimates of the characteristic ranges of the design variables.

HIPHA is motivated by applications in which HVPs are computed selectively to warm-start or refresh the Hessian approximation. In engineering design settings, optimization iterates often jump across regions of piecewise-smooth functions due to changes in physical regimes or numerical representations. Examples include the appearance or disappearance of a shock in compressible fluid flows, contact events in multi-body dynamics, topology changes in geometric representations, and periodic remeshing. Immediately after transitions between regions in these examples, an increased number of HVPs can rapidly re-establish Hessian-approximation accuracy in the new region. Even in the absence of such changes in regimes, it can be beneficial in general to perform a larger number of HVPs in the first major iteration to provide a stronger initial Hessian approximation than a diagonal or identity matrix.

In these cases, the numbers of HVPs computed in each major iteration would be nonuniform. HIPHA provides a framework that can make use of all available HVP information, while weighting each according to the distance from the current design vector. Future extensions that account for changes in curvature between historical HVP locations and the current design point may further improve robustness, especially after regime changes.

3.4. SQP with HVP-Enhanced Hessian Approximations

The following algorithm summarizes the major steps in a line-search SQP algorithm that leverages HVPs to construct explicit Hessian approximations. The HVP computation itself is not

part of the optimization algorithm; rather, it is provided by the model used to evaluate the objective and constraints. However, when efficient HVP implementations are unavailable, the algorithm can estimate the required HVPs using finite differences of gradients.

Algorithm 1 HVP-Enhanced Sequential Quadratic Programming

Inputs: Initial design variables x_0, initial multipliers λ_0, initial Hessian approximation B_0, and tolerances ϵ_{opt} and ϵ_{feas}

Outputs: Solution x^* and multiplier estimate λ^*

Set $k = 0$
while optimality and feasibility criteria are not satisfied **do**
  Evaluate $f(x_k)$, $c(x_k)$, $g(x_k)$, and $J(x_k)$
  Solve the QP subproblem (2) to obtain the search direction p_k and multiplier estimate $\widehat{\lambda}_k$
  Compute a step size $\alpha_k \in (0, 1]$ using a line search that minimizes a chosen merit function along p_k
  Update the design variables and multipliers: $x_{k+1} = x_k + \alpha_k p_k$, $\lambda_{k+1} = \lambda_k + \alpha_k(\widehat{\lambda}_k - \lambda_k)$
  Compute the required HVPs at (x_{k+1}, λ_{k+1})
  Construct a positive-definite B_{k+1} using the HVPs
  Set $k \leftarrow k + 1$
end while

4. NUMERICAL RESULTS

This section presents numerical results for the proposed Hessian approximation methods using the OpenSQP [30] optimizer from the modOpt [31] library. The goal of these experiments is to assess whether incorporating exact curvature information through Hessian-vector products can improve the robustness and efficiency of standard quasi-Newton methods.

We first study the two methods on the Rosenbrock problem and a cantilever beam problem to examine their behavior on representative test cases. The HVP-enhanced BFGS method is then evaluated on a large set of constrained and unconstrained problems from the CUTEst test collection to assess its performance across a diverse benchmark set.

In this section, iBFGS-m denotes the HVP-enhanced BFGS method with m in-place applications of the HVP-based BFGS updates in each iteration, as discussed in Sect. 3.2.3. Similarly, HIPHA-m denotes the distance-weighted Hessian approximation method introduced in Sect. 3.3, where m new HVPs are computed in each iteration and combined with distance-weighted HVP information from previous iterations to construct the Hessian approximation. In addition, OpenSQP denotes the standard OpenSQP algorithm using BFGS, and cvxNewton denotes a modified OpenSQP algorithm that uses the exact Lagrangian Hessian when it is positive definite and otherwise convexifies it by clipping eigenvalues below a prescribed lower bound.

In our studies, HIPHA uses $\alpha = 1$, since smaller values of α can produce large eigenvalues in the inverse Hessian approximation along directions not represented by the low-rank term $Z_k Z_k^T$. This can lead to steps p_k that are dominated by poorly modeled curvature directions. We note that the current HIPHA studies are limited to a 20-dimensional Rosenbrock problem, as efficient solution strategies for the resulting inner optimization problems remain under investigation and are beyond the scope of this paper.

4.1. Multidimensional Rosenbrock Function Minimization

The n-dimensional Rosenbrock function is a standard benchmark problem defined as

$$f(x) = \sum_{i=1}^{n-1} \beta(x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \tag{14}$$

where β controls the coupling between dimensions. This function is characterized by a steep, curved valley leading to the global minimum $x = (1, 1, \ldots, 1)$

We apply iBFGS and HIPHA to a 20-dimensional Rosenbrock problem with $\beta = 10$. We test both methods with varying numbers of HVPs per iteration and compare with OpenSQP, cvxNewton, and an OpenSQP variant that uses the DFP method. In this study, cvxNewton uses eigenvalue clipping, with all eigenvalues below 1 set to 1, to facilitate a fair comparison with HIPHA. The problem dimension is limited to $n = 20$ because HIPHA has quadratic runtime complexity with respect to the number of design variables.

Figure 1 shows the convergence behavior for each method. We plot the relative Hessian error, $\frac{\|B_k - H_k\|_F}{\|H_k\|_F}$, as well as OpenSQP's optimality measure. Methods using a single HVP per iteration are directly comparable to SQP and DFP, which update the Hessian approximation using secant information from the latest step. The iBFGS-1 method (28 iterations, 124 evaluations) performed slightly better than OpenSQP (32 iterations, 134 evaluations), corroborating our findings from the CUTEst benchmark presented later. As the number of HVPs increases, additional curvature information gets incorporated into the Hessian approximation, thus reducing the iterations and evaluations required for convergence.

On this problem, the convexified Newton method provides an approximate upper bound on performance because it uses the exact Hessian with eigenvalue clipping. Critically, even when 20 HVPs are used, both iBFGS and HIPHA remain less effective than convexified Newton. For iBFGS, this is expected because each direction is updated sequentially through individual BFGS updates rather than through a single multi-secant update that can recover the full Hessian. For HIPHA, the approximation is influenced by distance-weighted curvature information from previous iterations, and therefore does not necessarily recover the true Hessian even when full local curvature information is available at x_k.

Figures 2 and 3 show that incorporating additional HVP-based curvature updates in iBFGS improves convergence on a 256-dimensional Rosenbrock problem. In Fig. 3, all iBFGS variants converge in fewer iterations than standard OpenSQP, with larger numbers of HVP updates generally producing earlier reductions in the optimality residual. Although using more HVPs increases the cost per iteration, the reduction in iteration count leads to lower total objective evaluation counts for the Krylov-based strategies, especially the step-based Krylov variant. The step-history strategy provides more modest improvements, while performance of Krylov directions based on the recent step almost approaches that of the cvxNewton with 50 HVPs per iteration.

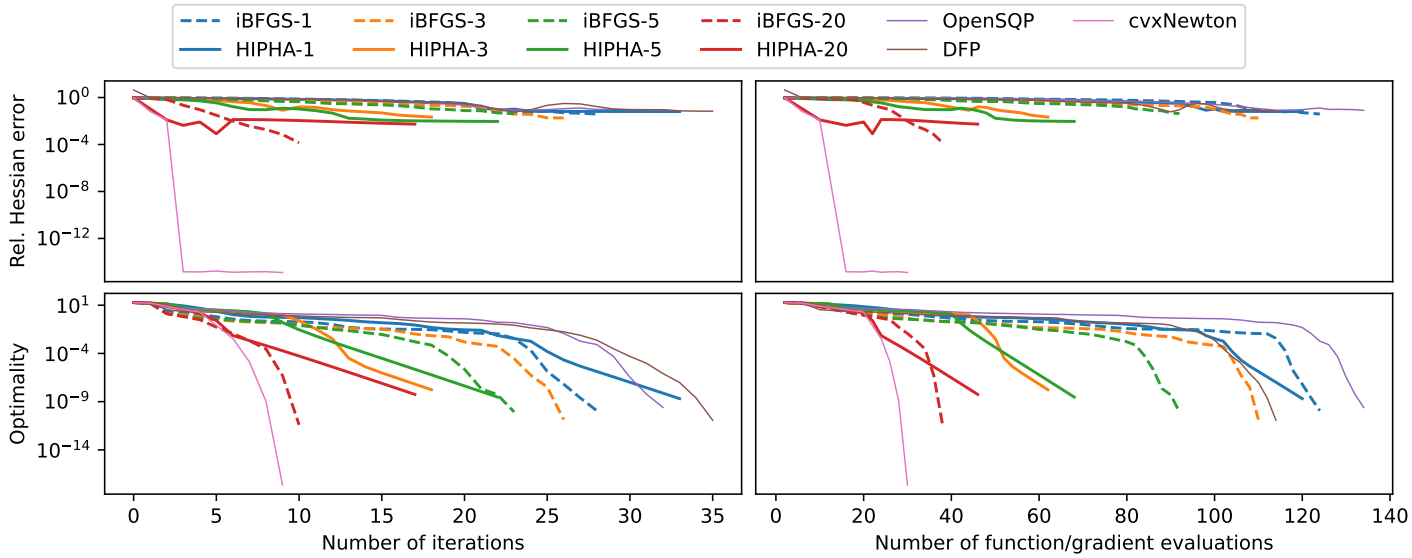

FIGURE 1: Convergence histories for the 20-dimensional Rosenbrock problem.

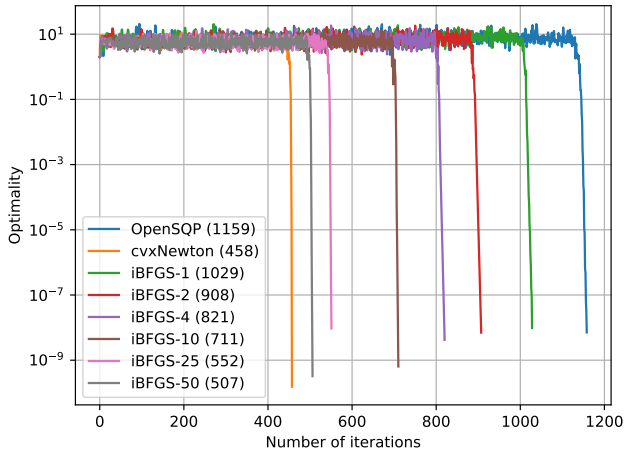

FIGURE 2: Optimality convergence histories for a 256-dimensional Rosenbrock problem. The numbers in parentheses indicate total SQP iterations. Increasing the number of HVP-based BFGS updates reduces the number of iterations required for convergence.

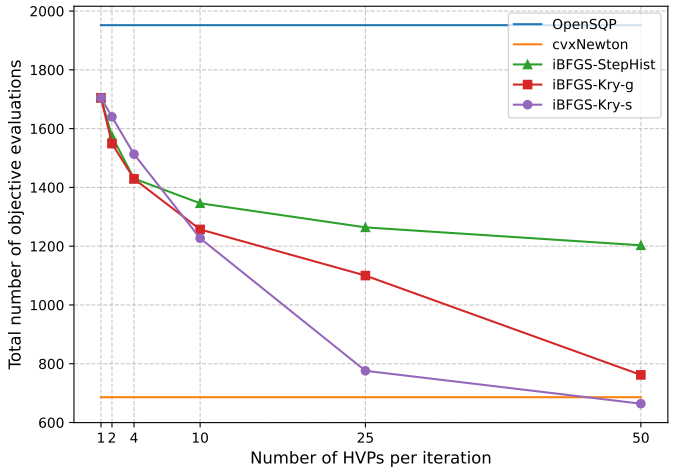

FIGURE 3: Total objective evaluation count for a 256-dimensional Rosenbrock problem. OpenSQP and cvxNewton are shown as reference methods, while the iBFGS variants compare different HVP direction-selection strategies as a function of the number of HVPs used per iteration.

4.2. Cantilever Beam Thickness Optimization

We next use a cantilever beam thickness optimization problem to study and compare the iBFGS variants. This problem is a structural compliance minimization problem for an Euler–Bernoulli beam discretized into n finite elements. The element thicknesses x_i are used as design variables. The beam is clamped at one end and subjected to a vertical load at the free end. The objective of the problem is the compliance $F^T u(x)$, where F is the force vector and $u(x)$ is the displacement vector obtained by solving the finite-element equilibrium equations. A fixed-volume equality constraint is imposed along with lower bounds on the design variables. The problem is modeled as described in [31], where additional details are provided. This problem serves as a scalable constrained benchmark for assessing the proposed HVP-based Hessian approximation strategies.

Similar to the high-dimensional Rosenbrock problem presented before, figures 4 and 5 show that incorporating additional HVP-based curvature updates in iBFGS improves convergence on a 200-dimensional cantilever beam optimization problem. We observe once again that step-history strategy provides only smaller improvements compared to the strategy based on Krylov directions.

4.3. CUTEst Benchmarking

4.3.1. Benchmarking Setup The benchmark set consists of 553 problems from the CUTEst [7] test collection, interfaced to modOpt through PyCUTEst [32]. The selected problems include

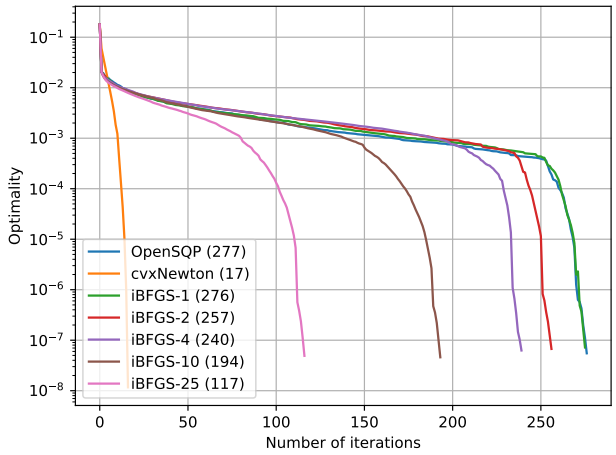

FIGURE 4: Optimality convergence histories for a 200-dimensional cantilever beam optimization problem. The numbers in parentheses indicate total SQP iterations. Increasing the number of HVP-based BFGS updates reduces the number of iterations required for convergence.

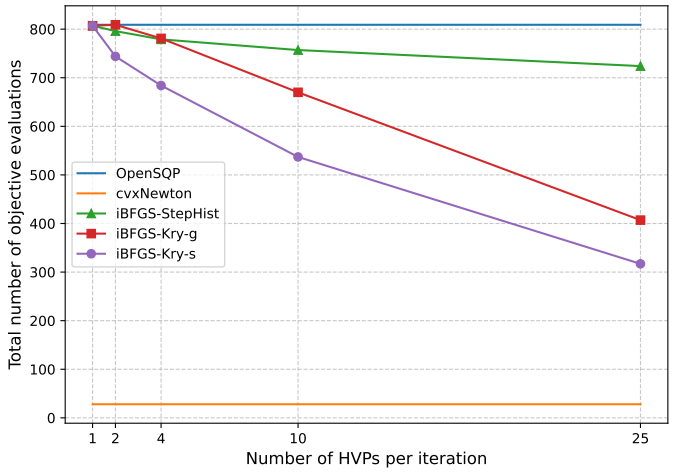

FIGURE 5: Total objective evaluation count for a a 200-dimensional cantilever beam optimization problem. OpenSQP and cvxNewton are shown as reference methods, while the iBFGS variants compare different HVP direction-selection strategies as a function of the number of HVPs used per iteration.

cases with up to 100 variables and constraints. For all runs, the OpenSQP options were fixed at `maxiter` = 500, `opt_tol` = 1.22×10^{-4}, `feas_tol` = 10^{-6}.

4.4. Performance Profiles

To compare the two solvers over the full test set, we use performance profiles in the sense of Dolan and Moré [33, 34]. For a given performance metric, the performance profile of a solver gives the fraction of problems for which that solver is within a factor τ of the best solver on that problem. In the plots shown (Figs. 6 and 7), the horizontal axis is reported as $\log_2(\tau)$. Thus, the value at $\log_2(\tau) = 0$ corresponds to the fraction of problems on which a solver is the best with respect to the chosen metric, while the limiting value as τ increases reflects the overall robustness of the solver on the benchmark set. Performance profiles therefore provide a compact way to assess both efficiency and reliability across a heterogeneous collection of problems.

Two performance metrics are considered in this study: (1) wall-clock time, and (2) total evaluation count. The total evaluation count includes objective/constraint evaluations, gradient/Jacobian evaluations, and Hessian-vector product evaluations. This aggregate count is used to reflect the total derivative-related computational work performed by each method.

4.4.1. Benchmark Results Standard OpenSQP successfully solved 479 of the 553 benchmark problems, while the single-HVP-enhanced iBFGS variant (denoted as OpenSQP-HVP in the figures) solved 486 problems. Figure 6 shows the performance profile based on wall-clock time. The HVP-enhanced OpenSQP variant lies above the standard OpenSQP curve over nearly the entire range of performance ratios, indicating better overall time performance. In particular, the HVP-enhanced method is the fastest solver on a larger fraction of the test problems and also attains a slightly higher asymptotic value, consistent with its higher total number of successfully solved problems. These results suggest that the additional exact curvature information can improve both efficiency and robustness when measured in elapsed time.

Figure 7 shows the corresponding performance profile based on total evaluation count. In this metric, standard OpenSQP performs better at $\log_2(\tau) = 0$, indicating that it achieves the lowest aggregate number of evaluations on a larger fraction of the problems. This behavior is expected because the HVP-enhanced method performs additional Hessian-vector product evaluations, even for simple problems that do not require and benefit from the additional curvature information. However, the HVP-enhanced method catches up rapidly as the performance ratio increases and ultimately achieves a slightly higher asymptotic value. This indicates that, although the proposed method may require more total evaluations on some problems, the added curvature information improves reliability and keeps the overall evaluation cost competitive across the benchmark set.

Taken together, the two profiles indicate that the proposed use of Hessian-vector products within the BFGS update can improve robustness while remaining computationally efficient. The time-based profile shows a clearer advantage for the HVP-enhanced method, while the evaluation-based profile suggests that this improvement is obtained without a substantial loss in total algorithmic efficiency, even when the cost of Hessian-vector products is explicitly included in the benchmark metric.

5. CONCLUSION

This paper investigated the use of Hessian-vector products to improve quasi-Newton Hessian approximations for large-scale system design optimization. We introduced an HVP-enhanced BFGS method that replaces standard gradient-difference curvature information with exact directional curvature information obtained from HVPs. We also presented HIPHA, a preliminary distance-weighted Hessian approximation strategy that fits a low-rank-plus-diagonal positive-definite model to current and historical HVP information.

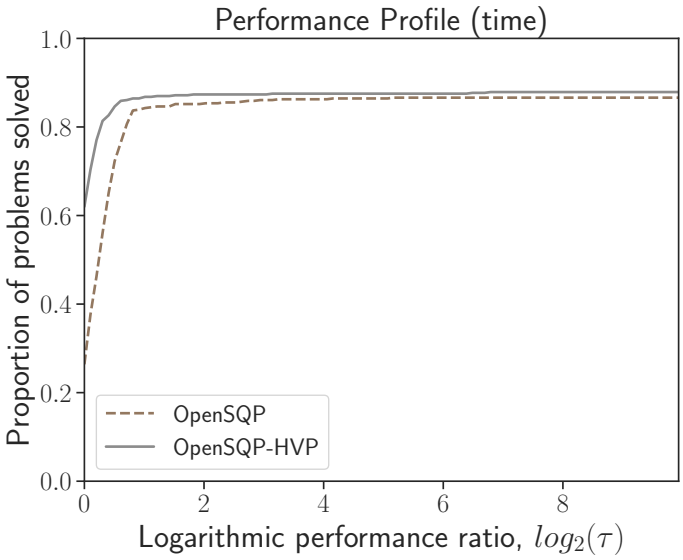

FIGURE 6: Performance profile based on wall-clock time for standard OpenSQP and the HVP-enhanced OpenSQP variant on the 553 CUTEst test problems.

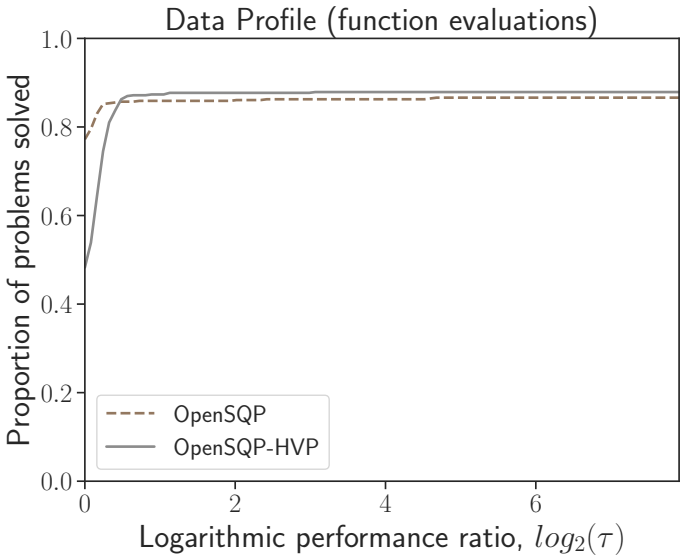

FIGURE 7: Performance profile based on total evaluation count, where the count includes objective, gradient, and Hessian-vector product evaluations.

Numerical results using the OpenSQP optimizer in modOpt show that incorporating HVP-based curvature information can improve convergence robustness and efficiency relative to standard BFGS. On the Rosenbrock and cantilever beam problems, the iBFGS variants generally reduced the number of iterations required for convergence, with the effectiveness depending on the number and choice of HVP directions. The CUTEst benchmark results further demonstrate that the single-direction HVP-enhanced BFGS update can improve performance across a broad set of constrained and unconstrained problems. The HIPHA results are more preliminary, but they illustrate the potential of distance-weighted HVP information as an alternative Hessian approximation strategy.

Future work will focus on more efficient solution strategies for the HIPHA inner optimization problem, improved policies for selecting HVP directions, and broader numerical studies on large-scale engineering design problems. These extensions will help clarify when additional HVP information provides the greatest benefit and how it can be incorporated most effectively into practical optimization algorithms.

ACKNOWLEDGMENTS

This material is based upon work supported by NASA under Award No. 80NSSC23M0217.

REFERENCES

[1] Gray, Justin S, Hwang, John T, Martins, Joaquim RRA, Moore, Kenneth T and Naylor, Bret A. "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization." *Structural and Multidisciplinary Optimization* Vol. 59 No. 4 (2019): pp. 1075–1104. DOI 10.1007/s00158-019-02211-z.

[2] Gandarillas, Victor, Joshy, Anugrah Jo, Sperry, Mark Z, Ivanov, Alexander K and Hwang, John T. "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis." *Structural and Multidisciplinary Optimization* Vol. 67 No. 5 (2024): p. 76. DOI 10.1007/s00158-024-03792-0.

[3] Bradbury, James, Frostig, Roy, Hawkins, Peter, Johnson, Matthew James, Leary, Chris, Maclaurin, Dougal, Necula, George, Paszke, Adam, VanderPlas, Jake, Wanderman-Milne, Skye and Zhang, Qiao. "JAX: composable transformations of Python+NumPy programs." (2018). URL http://github.com/google/jax.

[4] Byrd, Richard H, Hribar, Mary E and Nocedal, Jorge. "An interior point algorithm for large-scale nonlinear programming." *SIAM Journal on Optimization* Vol. 9 No. 4 (1999): pp. 877–900. DOI 10.1137/S1052623497325107.

[5] Byrd, Richard H, Gilbert, Jean Charles and Nocedal, Jorge. "A trust region method based on interior point techniques for nonlinear programming." *Mathematical programming* Vol. 89 (2000): pp. 149–185. DOI 10.1007/PL00011391.

[6] Gao, Wenbo and Goldfarb, Donald. "Block BFGS methods." *SIAM Journal on Optimization* Vol. 28 No. 2 (2018): pp. 1205–1231. DOI 10.1137/16M1092106.

[7] Gould, Nicholas IM, Orban, Dominique and Toint, Philippe L. "CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization." *Computational Optimization and Applications* Vol. 60 No. 3 (2015): pp. 545–557. DOI 10.1007/s10589-014-9687-3.

[8] Fourer, Robert, Gay, David M and Kernighan, Brian W. *AMPL. A modeling language for mathematical programming*. Thomson (2003).

[9] Rosenthal, Richard E. "GAMS – A User's Guide." 2004. URL https://api.semanticscholar.org/CorpusID:24434537.

[10] Hart, William E, Watson, Jean-Paul and Woodruff, David L. "Pyomo: modeling and solving mathematical programs in

Python." *Mathematical Programming Computation* Vol. 3 (2011): pp. 219–260. DOI 10.1007/s12532-011-0026-8.

[11] Dunning, Iain, Huchette, Joey and Lubin, Miles. "JuMP: A modeling language for mathematical optimization." *SIAM review* Vol. 59 No. 2 (2017): pp. 295–320. DOI 10.1137/15M1020575.

[12] Andersson, Joel A E, Gillis, Joris, Horn, Greg, Rawlings, James B and Diehl, Moritz. "CasADi – A software framework for nonlinear optimization and optimal control." *Mathematical Programming Computation* Vol. 11 No. 1 (2019): pp. 1–36. DOI 10.1007/s12532-018-0139-4.

[13] Martins, Joaquim RRA and Ning, Andrew. *Engineering design optimization*. Cambridge University Press (2021). DOI 10.1017/9781108980647.

[14] Gill, Philip E, Murray, Walter and Saunders, Michael A. "SNOPT: An SQP algorithm for large-scale constrained optimization." *SIAM review* Vol. 47 No. 1 (2005): pp. 99–131. DOI 10.1137/S0036144504446096.

[15] Wächter, Andreas and Biegler, Lorenz T. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." *Mathematical programming* Vol. 106 (2006): pp. 25–57. DOI 10.1007/s10107-004-0559-y.

[16] Kraft, Dieter. "A software package for sequential quadratic programming." *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[17] Broyden, Charles George. "The convergence of a class of double-rank minimization algorithms 1. general considerations." *IMA Journal of Applied Mathematics* Vol. 6 No. 1 (1970): pp. 76–90.

[18] Fletcher, Roger. "A new approach to variable metric algorithms." *The computer journal* Vol. 13 No. 3 (1970): pp. 317–322.

[19] Goldfarb, Donald. "A family of variable-metric methods derived by variational means." *Mathematics of computation* Vol. 24 No. 109 (1970): pp. 23–26.

[20] Shanno, David F. "Conditioning of quasi-Newton methods for function minimization." *Mathematics of computation* Vol. 24 No. 111 (1970): pp. 647–656.

[21] Davidon, WC. "Variable metric methods for minimization, AEC Res. and Develop." *Report ANL-5990, Argonne National Laboratory, Argonne, IL* (1959).

[22] Davidon, William C. "Variable metric method for minimization." *SIAM Journal on optimization* Vol. 1 No. 1 (1991): pp. 1–17. DOI 10.1137/0801001.

[23] Fletcher, Roger and Powell, Michael JD. "A rapidly convergent descent method for minimization." *The computer journal* Vol. 6 No. 2 (1963): pp. 163–168.

[24] Broyden, Charles G. "A class of methods for solving nonlinear simultaneous equations." *Mathematics of computation* Vol. 19 No. 92 (1965): pp. 577–593. DOI 10.2307/2003941.

[25] Gill, Philip E, Saunders, Michael A and Wong, Elizabeth. "On the performance of SQP methods for nonlinear optimization." *Modeling and Optimization: Theory and Applications: MOPTA, Bethlehem, PA, USA, August 2014 Selected Contributions*: pp. 95–123. 2015. Springer. DOI 10.1007/978-3-319-23699-5_5.

[26] Andrei, Neculai. "Accelerated conjugate gradient algorithm with finite difference Hessian/vector product approximation for unconstrained optimization." *Journal of Computational and Applied Mathematics* Vol. 230 No. 2 (2009): pp. 570–582.

[27] Hicken, Jason E. "Inexact Hessian-vector products in reduced-space differential-equation constrained optimization." *Optimization and Engineering* Vol. 15 No. 3 (2014): pp. 575–608.

[28] Canto, Eva Balsa, Banga, Julio R, Alonso, Antonio A and Vassiliadis, Vassilios S. "Restricted second order information for the solution of optimal control problems using control vector parameterization." *Journal of Process Control* Vol. 12 No. 2 (2002): pp. 243–255.

[29] Powell, Michael JD. "Algorithms for nonlinear constraints that use Lagrangian functions." *Mathematical programming* Vol. 14 (1978): pp. 224–248. DOI 10.1007/BF01588967.

[30] Joshy, Anugrah Jo and Hwang, John T. "OpenSQP: A Reconfigurable Open-Source SQP Algorithm in Python for Nonlinear Optimization." *Mathematical Programming Computation (under review)* (2025)DOI 10.48550/arXiv.2512.05392.

[31] Joshy, Anugrah Jo and Hwang, John T. "modOpt: A modular development environment and library for optimization algorithms." *Advances in Engineering Software* Vol. 213 (2026): p. 104084. DOI 10.1016/j.advengsoft.2025.104084.

[32] Fowkes, Jaroslav, Roberts, Lindon and Bűrmen, Árpád. "PyCUTEst: an open source Python package of optimization test problems." *Journal of Open Source Software* Vol. 7 No. 78 (2022): p. 4377. DOI 10.21105/joss.04377.

[33] Dolan, Elizabeth D and Moré, Jorge J. "Benchmarking optimization software with performance profiles." *Mathematical programming* Vol. 91 (2002): pp. 201–213. DOI 10.1007/s101070100263.

[34] Moré, Jorge J and Wild, Stefan M. "Benchmarking derivative-free optimization algorithms." *SIAM Journal on Optimization* Vol. 20 No. 1 (2009): pp. 172–191. DOI 10.1137/080724083.