

Nested differentiable geometric programming for gradient-based multidisciplinary design optimization

Teagan KJ Nakamoto^{1*} and Andrew Ning²

^{1*,2}Mechanical Engineering, Brigham Young University, 350 EB, Provo, 84602, UT, USA.

*Corresponding author(s). E-mail(s): teagan.nakamoto@gmail.com;

Abstract

Multidisciplinary design optimization (MDO) often relies on gradient-based nonlinear optimization, which can accommodate complex engineering models with minimal reformulation. While powerful, these methods are sensitive to parameter tuning and can struggle in highly nonconvex design spaces. Convex optimization techniques guarantee fast and reliable convergence, but are typically too restrictive for realistic engineering analyses. Geometric programming (GP) enables some nonlinear problems to be reformulated in convex form, but it can rarely capture all required relationships in practical MDO applications. In this paper, we develop a differentiable nested optimization approach that embeds GP-compatible components within a larger nonlinear MDO problem. An inner optimization layer solves a parameterized GP subproblem and provides gradients through implicit differentiation, while an outer layer performs a standard nonlinear optimization over the remaining variables, analyses, and constraints. This nesting approach retains the flexibility of nonlinear programming while leveraging the robustness of convex optimization for subproblems that can be reformulated into GPs. This approach is effective for problems in which a meaningful portion of the analysis can be reformulated as a GP, and for which the remaining nonlinear components are computationally expensive relative to the cost of GP construction, solution, and differentiation. Numerical experiments and a wind turbine design study demonstrate order-of-magnitude reductions in run time and improved solution quality compared to a purely nonlinear optimization approach. More broadly, this work enables existing GP engineering models to be integrated into larger nonlinear MDO workflows and motivates further efforts to exploit hidden convex structure within otherwise nonconvex design problems.

Keywords: nonlinear optimization, convex optimization, geometric programming, gradient-based methods, multidisciplinary design optimization

1 Introduction

Industries such as energy production, manufacturing, and transportation compete in a global environment in which systems must satisfy economic and regulatory constraints while continually improving performance (Mihai 2023; Canitez 2018; Jovane et al. 2008). Because these systems are large in scale and densely interconnected, even incremental improvements can have significant long-term impacts on efficiency, cost, and sustainability. Gradient-based multidisciplinary design optimization (MDO) provides a rigorous framework for realizing such improvements through computational modeling and algorithmic design exploration (Simpson and Martins 2011).

Many MDO applications rely on nonlinear (NL) optimization methods for their applicability to a wide variety of problem structures. While formulating such problems is often straightforward once a systems-analysis model is available, solving them can be challenging. Gradient-based NL algorithms are sensitive to initialization and scaling, and the nonconvexity typical of MDO problems leads to numerical issues that can trap optimizers and slow convergence. As a result, reliability and

efficiency are difficult to ensure in large-scale engineering applications. As computational resources grow (Sterling 2009; Hwang et al. 2014), the size and complexity of real-world design problems increase faster still, making reliability and efficiency ever more difficult to ensure.

By contrast, convex optimization methods provide strong theoretical guarantees and converge quickly to the global minimum without the intricacies of parameter tuning and starting point selection. This makes convex approaches particularly attractive in applications such as dynamics and controls, where speed and robustness matter more than model fidelity (Agrawal et al. 2020).

Geometric programming (GP) is a structured class of nonlinear problems that admits a convex reformulation via a log-log transformation (Duffin et al. 1967; Boyd and Vandenberghe 2004). When engineering models can be expressed in terms of GP monomials and posynomials, systems with thousands of variables and constraints can be optimized reliably and efficiently (Boyd et al. 2007; Hoburg and Abbeel 2014; Burton and Hoburg 2018b; Kim et al. 2025; Cole and Traykovski 2025).

The applicability of geometric programming is limited by its strict mathematical prescription. Engineering models often include trigonometric, exponential, piecewise, and other relations outside the narrow class of posynomials. While analytical reformulations and data-driven posynomial fitting can extend its reach (Calafiore et al. 2015), they are often labor-intensive. Signomial programming further relaxes GP structure (York et al. 2018), but sacrifices global convergence guarantees and scales poorly.

Together, these observations expose a practical gap in MDO workflows: convex methods are reliable but restrictive, while nonlinear methods are flexible but fragile.

Attempts to bridge this gap commonly take the form of decomposed or multi-strategy optimization methods. Broadly, these include hybrid approaches that combine distinct algorithms (Manios et al. 2019; Stanford et al. 2018), sequential methods that solve a series of approximate subproblems (Gill et al. 2005; Karcher and Haimes 2023; Kim et al. 2025), and nested formulations in which inner solvers provide information to an outer optimizer (Herber and Allison 2019; Li and Cheng 2022).

Within MDO, these ideas are formalized through architectural frameworks such as simultaneous analysis and design (SAND), multidisciplinary feasible (MDF), and individual discipline feasible (IDF), which prescribe how analyses, solvers, and coupling variables are coordinated. In all of these architectures, a top-level optimizer explores the design space while inner solvers enforce disciplinary feasibility. A persistent challenge is that the outer optimization often drives inner solvers into poorly scaled or atypical regimes, placing heavy demands on robustness and often requiring manual tuning.

In contrast, a parameterized GP subproblem behaves as a specialized optimizer/solver with reliable, tuning-free convergence to tight tolerances—precisely the characteristics required for accurate derivative propagation. This motivates a nested nonconvex–convex architecture in which GP subproblems act as inner solves within a broader nonlinear framework.

More generally, nested optimization with implicit differentiation has been studied extensively, with efficient methods developed for differentiating through least-squares solvers, quadratic programs, and smooth constrained optimizations, particularly in machine learning, control, and inverse problems (Bell and Burke 2008; Blondel et al. 2022; Gould et al. 2016). However, this literature primarily considers nested gradient-based solvers and does not exploit inner problems that admit convex reformulation. To our knowledge, geometric programming has not previously been used as a differentiable inner optimization layer within nonlinear MDO.

In this paper, we propose a nested nonconvex–convex optimization framework in which a differentiable GP subproblem is embedded within a nonlinear optimization. The outer nonlinear program accommodates the overall objective and constraints, while the inner GP efficiently solves problem components that permit convex reformulation. Gradients are propagated through the parameterized GP subproblem using implicit differentiation, enabling GP-compatible models to be integrated seamlessly into larger nonlinear MDO problems.

We demonstrate our methods with numerical examples and a simplified wind turbine design problem. Comparing to conventional nonlinear formulations, we show that our nested GP approach can reduce computational cost and improve solution quality for some MDO problems. These results highlight the potential of nested convex optimization for exploiting hidden convex substructure in nonlinear optimization problems.

The remainder of this paper is organized as follows. Section 2 reviews necessary background on geometric programming and convex optimization. Section 3 presents the nested differentiable parameterized GP method. Section 4 describes numerical benchmarking studies. In Section 5, we formulate a geometric program for wind turbine tower design. In Section 6 we detail a demonstrative

wind turbine design problem and compare results from nonlinear and nested-GP optimization studies. We conclude with Section 7.

2 Methods—Preliminaries

Nested optimization is challenging because the inner optimization must converge tightly at every iteration to ensure accurate derivative computation. For general nonlinear programs, this often necessitates hand-tuning of problem scale and algorithmic parameters; worse, convergence behavior can vary significantly across the parameter space. The use of convex optimization methods guarantees efficient, parameter-insensitive convergence to tight tolerances. In this work, we exploit this property by embedding geometric programs (GPs)—nonlinear problems that admit convex reformulation—as inner solves within a nonlinear optimization framework.

In this section, we present background information preliminary to developing our differentiable parameterized GP framework. We begin with a brief definition of geometric programming. We then present some practical considerations for implementation, including our choice of modeling interface and optimizer.

2.1 Geometric and convex programming

Geometric programming is a class of nonlinear optimization problems built from monomial and posynomial functions. Here, a monomial is defined as

$$h(x) \equiv cx_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \quad c > 0, \quad a \in \mathbb{R}^n, \quad (1)$$

and a posynomial function is defined as a sum of one or more monomials:

$$f(x) \equiv \sum_{j=1}^m c_j \prod_{i=1}^n x_i^{a_{ij}}, \quad c_j > 0, \quad a \in \mathbb{R}^{n \times m}. \quad (2)$$

The canonical form of a GP consists of a posynomial objective, N posynomial inequality constraints, and M strictly monomial equality constraints as given in Equation 3. In this notation, the symbol f denotes a posynomial function, while h specifies a monomial function.

$$\begin{aligned} & \underset{x > 0}{\text{minimize}} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 1, \quad i = 1, \dots, N, \\ & && h_i(x) = 1, \quad i = 1, \dots, M \end{aligned} \quad (3)$$

In posynomial form, a GP is not convex. However, through a log–log change of variables, it can be transformed into a convex conic program—typically an exponential cone program, and in special cases a second-order cone program. This convex reformulation enables reliable and efficient solution using modern interior-point and first-order convex optimization methods. Further details on geometric programming and its convex formulation can be found in [Boyd and Vandenberghe \(2004\)](#).

2.2 Implementation and solver considerations

Several modeling frameworks support geometric programming and convex optimization, including Convex.jl ([Udell et al. 2014](#)), JuMP ([Lubin et al. 2023](#)), and GPkit ([Burnell and Hoburg 2018a](#)). These tools provide convenient high-level interfaces for constructing and verifying GP models. However, their standard usage relies on expression parsing and dynamic typing, adding overhead that can become significant when an inner optimization is solved repeatedly within a nested framework.

To reduce this overhead, we implement our GP subproblems directly using MathOptInterface (MOI), the low-level abstraction layer underlying both JuMP and Convex.jl ([Legat et al. 2021](#)). To quantify the impact of modeling-layer overhead, we benchmarked a simple GP (the box design problem from Section 2.4 of [Boyd et al. 2007](#)) implemented using GPkit, Convex.jl, and MOI. Table 1 reports median run times over approximately 10,000 solves. The MOI implementation achieves roughly an order-of-magnitude speedup relative to higher-level interfaces, indicating substantial overhead associated with expression parsing and symbolic graph construction.

Table 1: Median run times for a simple convex benchmark problem implemented using different combinations of a modeling framework and an optimizer. Statistics were collected using BenchmarkTools (Chen and Revels 2016) with $\approx 10,000$ samples for each method tested

Method	Time (μs)
GPkit–Mosek	8267
GPkit–cvxopt	2549
Convex–Mosek	1726
Convex–Clarabel	1015
MOI–Mosek	834
MOI–Clarabel	101

Although geometric programs can also be solved directly as nonlinear programs, solver choice has significant implications for scalability and robustness. To quantify this effect, we construct an unconstrained posynomial objective function of $4(N - 1)$ sparse monomial terms:

$$f_{sparse}(x) = \sum_{i=1}^{N-1} x_i^{2.7+0.3(-1)^i} x_{i+1}^{5.1} + 2.4x_i^{-2.3} + x_i^{3.7} + x_{i+1}^{-4.1} \quad (4)$$

Here, the term “sparse,” indicates that each monomial term is a function of only one or two design variables. An analogy can be drawn to sparse Jacobian matrices, where columns correspond to variables and rows correspond to equations. A posynomial can similarly be expressed as a matrix, with columns corresponding to variables and rows corresponding to monomial terms. Each variable has an exponent in every monomial term, but in the sparse case many of those exponents are zero. We optionally add two “dense” monomial terms to the sparse objective function, so we can assess the effect of posynomial sparsity on optimizer performance.

$$f_{dense}(x) = \prod_{i=1}^N x_i^{3(-1)^i} + \prod_{i=1}^N x_i^{-2.1(-1)^i} \quad (5)$$

We compare the performance of a nonlinear optimizer (SNOPT) and a dedicated convex solver (Clarabel) as the number of variables and monomial terms increases. Figure 1 shows that while SNOPT performs well for small, sparse problems, its run time degrades rapidly as problem size increases or dense monomial terms are introduced. In contrast, the convex solver scales more favorably and dominates for moderate to large problems. These results indicate that although nonlinear optimizers can solve GPs, dedicated convex methods are substantially more robust and scalable for problems of practical complexity.

Among the optimizers tested, Clarabel (Goulart and Chen 2024) exhibited performance comparable to Mosek (ApS 2025) while offering two practical advantages: it is fully open source and implemented in Julia. Unless otherwise noted, we use the Clarabel optimizer through the MOI interface for all GP solutions. Although defining problems at the MOI level requires additional setup compared to JuMP and Convex.jl, it reduces run time computational overhead in the nested optimization.

3 Methods—Nested GP Architecture

In this section, we present the mathematical formulation of the nested GP approach and derive a method for post-optimality implicit differentiation of a parameterized geometric program.

3.1 Nested optimization

Following common conventions for bilevel optimization, we define the top-level (outer) optimization with objective function F and constraint functions G and H , while the nested (inner) optimization problem is defined by functions f , g , and h (see Equation 6). Design variables are designated as X

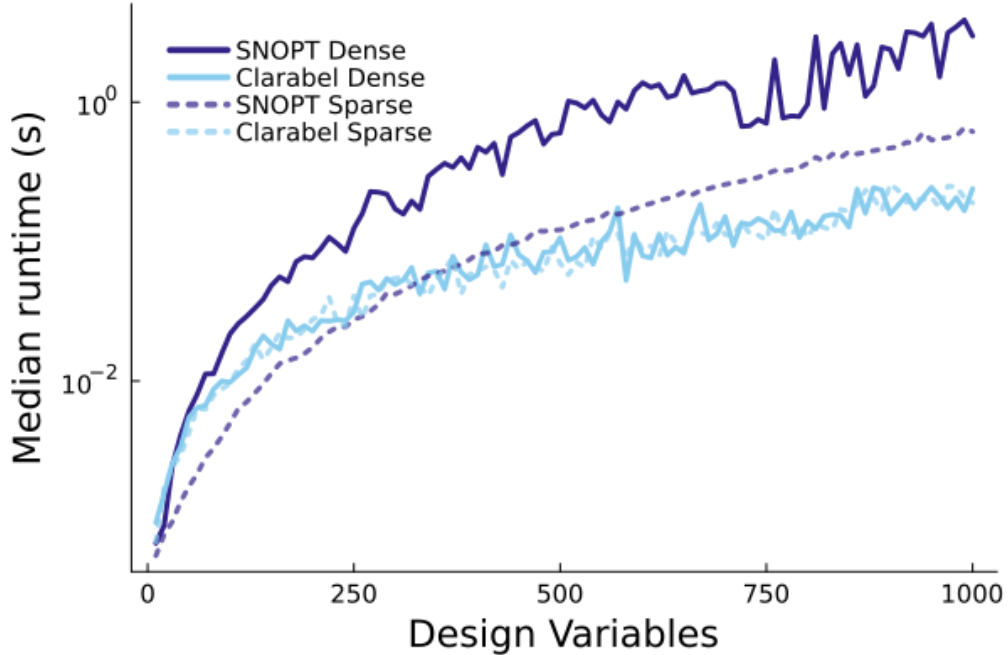


Fig. 1: Comparison of run times for nonlinear (SNOPT) and convex (Clarabel) optimizers. The addition of dense monomial terms significantly increases the run time of the nonlinear optimizer. Note that for N design variables there are $4(N - 1)$ sparse monomial terms and 2 dense monomial terms

and x for the outer and inner optimizations, respectively. The two levels are linked by parameters θ , which are functions of, or otherwise derived from, the outer design variables X . The inner GP solver returns x^* and λ^* , the primal and dual variables at optimality: the superscripted star indicates that the x and λ values are optimal with respect to the inner, GP variables and constraints.

$$\begin{aligned}
 & \underset{X \in \mathbb{R}^n}{\text{minimize}} && F(X; x^*) \\
 & \text{subject to} && H(X; x^*) = 0, \\
 & && G(X; x^*) \leq 0, \\
 & && x^*, \lambda^* = \text{GPsolve}(\theta) \\
 & && \text{where } \theta = f(X)
 \end{aligned} \tag{6}$$

We assume F , G , and H are compositions of functions, solvers, and analytic models that are all differentiable with respect to X and x^* . Gradient-based solution of the outer problem requires derivatives of the GP solution with respect to the parameters θ , which are developed in the following subsection.

3.2 Differentiation through a parameterized geometric program

Following Equations 3 and 6, we express the parameterized geometric program as:

$$\begin{aligned}
 & \underset{x > 0}{\text{minimize}} && f(x; \theta) \\
 & \text{subject to} && h_i(x; \theta) = 1 \quad i = 1, \dots, M, \\
 & && g_i(x; \theta) \leq 1 \quad i = 1, \dots, N
 \end{aligned} \tag{7}$$

We solve this parameterized GP with a convex optimization algorithm. From standard optimization theory, each constraint is associated with a Lagrange multiplier λ_i . For notational simplicity, we use λ to denote the multipliers corresponding to all post-optimality active constraints, including active inequality constraints. Active inequality constraints are combined with equality constraints in

the symbol h , yielding the post-optimality Lagrangian

$$\mathcal{L}(x^*, \lambda^*; \theta) = f(x^*; \theta) + \sum_{i=1}^{N_{\text{active}}} \lambda_i^* h_i(x^*; \theta) \quad (8)$$

To propagate sensitivities from the outer optimization, we require derivatives of the optimal GP solution x^* with respect to the parameters θ . These derivatives are obtained via implicit differentiation of the Karush–Kuhn–Tucker (KKT) optimality conditions (Karush 1939; Kuhn and Tucker 1951). At optimality, the KKT conditions enforce stationarity and feasibility:

$$\frac{\partial \mathcal{L}}{\partial x^*} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda_i^*} = h_i(x^*; \theta) = 0, \quad i = 1, \dots, N_{\text{active}} \quad (9)$$

These equations define residual functions $R(\theta, z) = 0$, where $z = [x^*, \lambda^*]$ collects the primal and dual optimizer outputs. Although the outer optimization ultimately requires only the sensitivities

$$\dot{x} = \frac{dx^*}{d\theta} \dot{\theta}, \quad \text{where} \quad \dot{\theta} = \frac{d\theta}{dX}, \quad (10)$$

the KKT conditions couple x^* and λ^* . Consequently, the primal sensitivities cannot be computed independently.

For this reason, we collect the primal and dual variables into a single vector $z = [x^*, \lambda^*]$ and apply implicit differentiation to the residual equations $R(\theta, z) = 0$. From the KKT conditions in residual form we obtain \dot{z} :

$$\frac{\partial R}{\partial z} \dot{z} = -\frac{\partial R}{\partial \theta} \dot{\theta} \quad (11)$$

Because the residuals are defined by the first-order optimality conditions, the required Jacobians can be expressed in terms of second derivatives of the Lagrangian:

$$\frac{\partial R}{\partial z} = \frac{\partial^2 \mathcal{L}}{\partial z^2}, \quad \frac{\partial R}{\partial \theta} = \frac{\partial^2 \mathcal{L}}{\partial z \partial \theta}. \quad (12)$$

Thus, the derivative computation reduces to solving the linear system

$$A \dot{z} = b, \quad (13)$$

where

$$A = \frac{\partial^2 \mathcal{L}}{\partial z^2}, \quad b = -\frac{\partial^2 \mathcal{L}}{\partial z \partial \theta} \dot{\theta}. \quad (14)$$

Here, A is the Hessian of the Lagrangian with respect to the primal and dual variables. This Hessian is best obtained by employing a forward-mode AD Jacobian over the reverse-mode gradient of the Lagrangian. While this Hessian is sparse for a typical constrained optimization problem, we are not able to exploit this effectively because the sparsity pattern changes from iteration to iteration with the changing active set of constraints. The right-hand side b represents a mixed-partial derivative of the Lagrangian contracted with $\dot{\theta}$, forming a Jacobian–vector product that propagates sensitivities from the outer design variables. Solving this system yields \dot{z} , from which the required sensitivities $\dot{x} = dx^*/dX$ are obtained by extracting the primal components.

In summary, implicit differentiation provides analytic sensitivities of the GP solution with respect to its input parameters, enabling geometric programming subproblems to be embedded as differentiable components within larger nonlinear optimization frameworks.

3.3 Log Lagrange multipliers

In the preceding sections, we have considered differentiation through a GP in its posynomial form. If we solve a GP with a nonlinear optimizer, we can apply implicit differentiation directly using the Lagrange multipliers returned by the optimizer. However, convex optimizers operate in the logarithmic space of the GP. As a result, the solver returns log-domain Lagrange multipliers, which differ in scale and interpretation from those of the linear domain. (Typical solvers will also return log-domain values of the objective function and the design variables, but these are simply converted back

via exponentiation.) Here we derive the relationship between log-domain Lagrange multipliers and linear-domain Lagrange multipliers.

Lagrange multipliers can be interpreted as the derivative of the objective function with respect to the value of its constraint function: $\lambda_i = df/dh_i$. Log Lagrange multipliers thus have the interpretation: $\lambda_{\log} = d \log f / d \log h$ (index subscripts omitted for readability). Expanding via the chain rule, we arrive at a relationship between λ_{\log} and λ_{linear} :

$$\lambda_{\log} = \frac{d \log f}{df} \frac{df}{dh} \frac{dh}{d \log h} = \frac{h}{f} \frac{df}{dh} = \frac{h}{f} \lambda_{\text{linear}} \quad (15)$$

At optimality, GP constraints are normalized such that $h = 1$, and therefore the two multipliers are related by

$$\lambda_{\text{linear}} = f(x^*; \theta) \lambda_{\log}. \quad (16)$$

This conversion allows log-domain multipliers obtained from convex solvers to be interpreted and used consistently within the implicit differentiation framework.

4 Simple Numerical Cases

In this section, we examine simple, unconstrained numerical optimization problems designed to illustrate the behavior of nested GP methods when applied to two distinct coupling structures. In the *feed-forward* case, the nonlinear portion of the objective is independent of the GP variables and the GP subproblem can be solved independently at each outer iteration, reducing the effective dimension of the outer nonlinear search. In the *coupled* case, the nonlinear objective depends explicitly on GP variables; naive feed-forward nesting becomes inconsistent, and additional coordination is required to obtain the correct solution.

These examples highlight two practical roles of nested GP subproblems: (i) reduction of the effective dimension of the outer nonlinear optimization when GP-compatible components are feed-forward, and (ii) provision of structured sensitivity information in coupled problems once consistency is enforced.

4.1 Problem structure and nested formulations

We consider objective functions $F(x)$ whose variables can be partitioned as $x = (x_{\text{nl}}, x_{\text{gp}})$, where x_{gp} denotes variables optimized within a GP subproblem and x_{nl} denotes the remaining nonlinear variables. We assume that F admits a decomposition of the form

$$F(x) = \Phi(f_{\text{nl}}(x_{\text{nl}}; x_{\text{gp}}), f_{\text{gp}}(x_{\text{gp}}; x_{\text{nl}})), \quad (17)$$

where f_{gp} is posynomial in x_{gp} and Φ is strictly increasing in each argument. This monotonicity ensures that decreasing either component cannot increase F , so minimizing $\Phi(f_{\text{nl}}, f_{\text{gp}})$ is consistent with minimizing F .

Two cases arise depending on the dependence structure of f_{nl} .

Feed-forward nested form

If f_{nl} is independent of x_{gp} , then minimizing f_{gp} for fixed x_{nl} is aligned with minimizing F with respect to x_{gp} . In this case, the inner problem is a parameterized GP

$$x_{\text{gp}}^*(x_{\text{nl}}) = \arg \min_{x_{\text{gp}} > 0} f_{\text{gp}}(x_{\text{gp}}; x_{\text{nl}}), \quad (18)$$

and the outer optimizer minimizes the reduced objective

$$\underset{x_{\text{nl}}}{\text{minimize}} \quad F(x_{\text{nl}}, x_{\text{gp}}^*(x_{\text{nl}})). \quad (19)$$

The objective is unchanged from the original problem; only the optimization architecture is modified.

Coupled case and corrected nested form.

If f_{nl} depends on x_{gp} , minimizing f_{gp} alone is no longer, in general, consistent with minimizing F . One way to restore consistency is to introduce target variables x_{gp}^t chosen by the outer optimizer and enforce agreement through GP-compatible penalty terms. This yields a corrected nested formulation that applies to the coupled regime while preserving the convex structure of the inner solve. Equations 18 and 19 still define the nested architecture, but with the outer variable set expanded to include x_{gp}^t and with the inner GP objective augmented by the GP-compatible penalty. Consequently, the inner solution mapping becomes $x_{\text{gp}}^*(x_{\text{nl}}, x_{\text{gp}}^t)$. This corrected formulation is developed and demonstrated in Section 4.3.

4.2 Feed-forward case: exact nesting and dimensionality reduction

When the nonlinear component is independent of the GP variables, the nested formulation is equivalent to a purely nonlinear optimization. In this case, the outer problem is reduced to the dimension of x_{nl} . As a concrete example, consider the unconstrained two-dimensional objective

$$F(x) = \frac{1}{30}x_{\text{nl}}^2 - 2\cos(1.2x_{\text{nl}}) - 2x_{\text{nl}} + \frac{4x_{\text{nl}}^{1.5}}{x_{\text{gp}}} + 5x_{\text{gp}}^{0.5} \tag{20}$$

Here, the posynomial component of the objective is highlighted in blue. The nonlinear portion of the objective function is independent of x_{gp} , yielding a feed-forward structure. The nested problem becomes

$$\begin{aligned} &\underset{x_{\text{nl}} \in \mathbb{R}}{\text{minimize}} && F(x_{\text{nl}}, x_{\text{gp}}^*) \\ &\text{where } && x_{\text{gp}}^* = \arg \min_{x_{\text{gp}} > 0} \left\{ \frac{4x_{\text{nl}}^{1.5}}{x_{\text{gp}}} + 5x_{\text{gp}}^{0.5} \right\} \end{aligned} \tag{21}$$

Figure 2 and Table 2 compare this nested GP formulation with a standard nonlinear optimization of F . Although the GP solve introduces nontrivial overhead relative to the simplicity of the fully nonlinear formulation, the number of major iterations is reduced by approximately a factor of three for the same starting point. This reduction suggests that the nested GP approach becomes advantageous when each nonlinear iteration is computationally expensive relative to the cost of the convex subproblem.

Geometrically, the GP subproblem identifies a one-dimensional manifold $x_{\text{gp}}^*(x_{\text{nl}})$ during the first iteration. Subsequent iterations of the nested method proceed along this manifold toward the minimum. By contrast, the fully nonlinear method follows a more generic search trajectory before converging to the same optimum. In higher-dimensional problems, this mechanism generalizes to dimensionality reduction of the outer optimization along the GP-optimal manifold.

Table 2: Solution statistics for the feed-forward nested-GP optimization problem. Wall time is the median statistic provided by BenchmarkTools.jl, with over 200 samples for each method. The starting point was held constant

	Nonlinear	Nested GP
Starting point	[6.26, 2.0]	—
End point	[5.13948, 7.03071]	[5.13948, 7.02998]
Function calls	15	7
Major iterations	13	4
Wall time (ms)	1.786	9.522
Objective	8.501526	8.501526

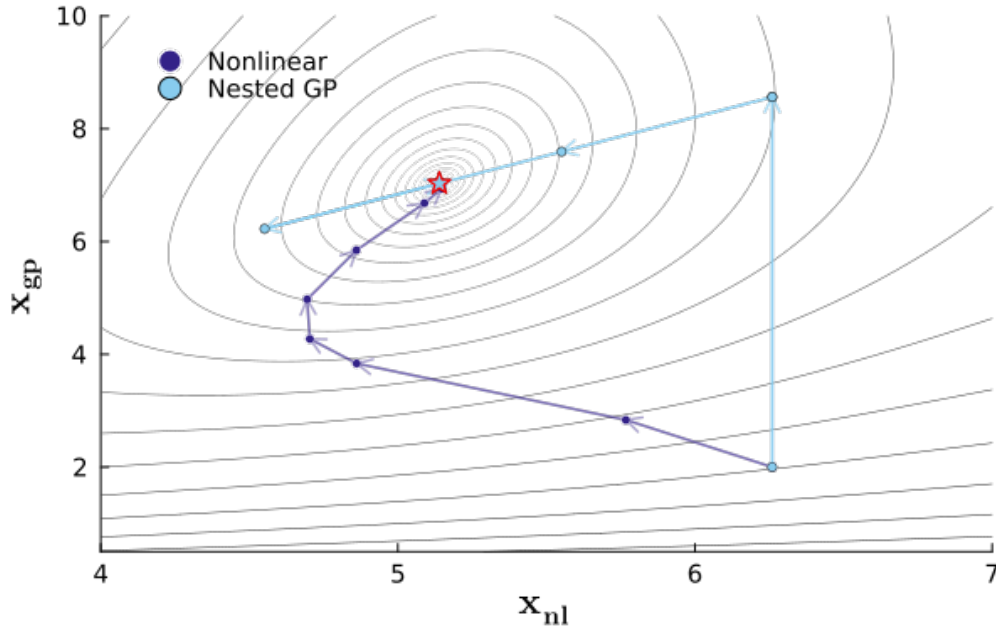


Fig. 2: Iteration path for a feed-forward nested-GP optimization problem. After the first GP solve, the nested optimizer causes all iterations to lie on the one-dimensional manifold corresponding to x_{gp}^* , given x_{nl} . Both the purely nonlinear and nested-GP methods converge to the same optimum, indicated by the star marker

4.3 Coupled case: failure of naive nesting and corrected formulation

In general, the convex (GP) and nonconvex components of the objective may be mutually coupled. In particular, the nonlinear analysis may depend explicitly on variables optimized within the GP subproblem. In this case, the feed-forward assumption no longer holds: minimizing the posynomial component alone is not, in general, consistent with minimizing the full objective.

To illustrate this, we modify the objective in Equation 20 by introducing an explicit nonlinear dependence on x_{gp} . Once again, we consider a two-dimensional unconstrained objective, with the posynomial component shown in blue:

$$\begin{aligned}
 F(x) &= 0.15x_{\text{nl}}^2 \sin(x_{\text{gp}}) - 2 \cos(1.2x_{\text{nl}}) \\
 &\quad - 2x_{\text{nl}} + \frac{4x_{\text{nl}}^{1.5}}{x_{\text{gp}}} + 5x_{\text{gp}}^{0.5}
 \end{aligned} \tag{22}$$

The periodic and subtracted terms are not GP-compatible, and so must remain in the nonlinear part of the objective. However, because f_{nl} now depends explicitly on x_{gp} , the GP subproblem no longer captures the full dependence of F on x_{gp} .

Naive feed-forward nesting fails

If the feed-forward nested formulation is applied to this problem, it converges to an incorrect solution. As shown in Figure 3, the true optimum no longer lies on the manifold defined by the GP minimizer of the posynomial terms alone.

Corrected nested formulation

To restore consistency, we introduce a target variable x_{gp}^t selected by the outer optimizer and enforce agreement through a posynomial penalty weighted by a scalar parameter w , yielding the corrected nested formulation in Equation 23.

$$\underset{x_{\text{nl}} \in \mathbb{R}, x_{\text{gp}}^t > 0}{\text{minimize}} \quad F(x_{\text{nl}}, x_{\text{gp}}^*) \tag{23}$$

$$\text{where } x_{\text{gp}}^* = \arg \min_{x_{\text{gp}} > 0} \left\{ \begin{aligned} & \frac{4x_{\text{nl}}^{1.5}}{x_{\text{gp}}} + 5x_{\text{gp}}^{0.5} \\ & + w \left(x_{\text{gp}} + \frac{(x_{\text{gp}}^t)^2}{x_{\text{gp}}} \right) \end{aligned} \right\}$$

The two-term posynomial $x + a^2/x$ attains its minimum at $x = a$; thus, sufficiently large values of w enforce approximate agreement between x_{gp} and x_{gp}^t in the presence of other posynomial objective terms. For more complex problems, w may be changed adaptively as the optimization proceeds. For the present example, a constant value of $w = 100$ was sufficient.

With this modification, derivatives of the GP solution with respect to its parameters provide the outer optimizer with sensitivity information that accounts for the effect of x_{gp} on the nonlinear objective. The corrected nested formulation therefore converges to the true optimum of the original objective. Table 3 shows that the corrected nested method reaches the correct solution while requiring fewer major iterations than the fully nonlinear formulation.

In this example, x_{gp} is scalar, so the nonlinear optimization space remains two-dimensional, unlike in the feed-forward case. In higher-dimensional problems, the coupling between the GP and nonlinear portions of the problem may be sparse; in such cases, only a subset of GP outputs would require target variables to ensure consistency.

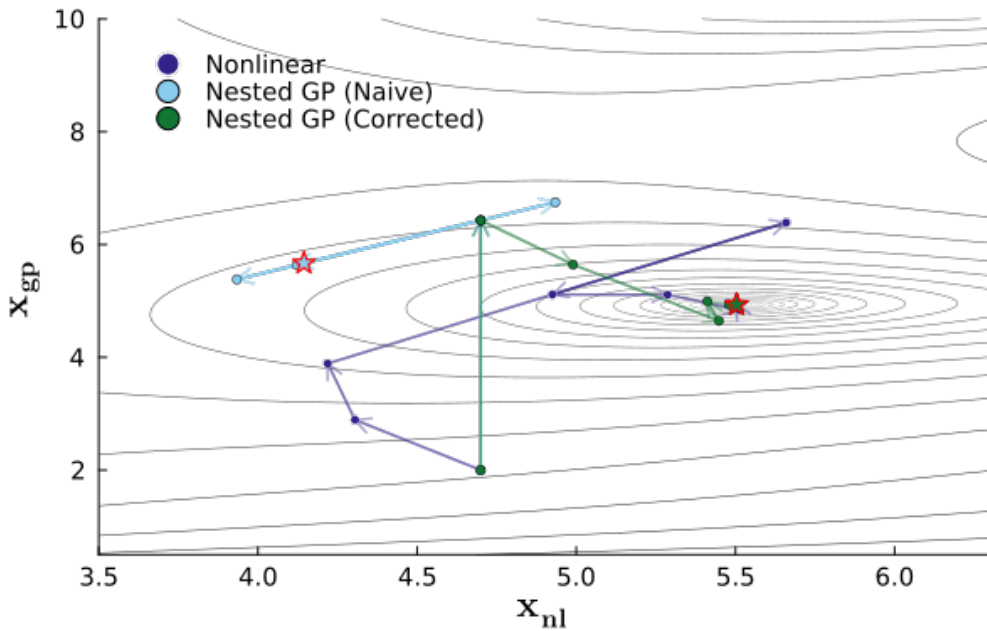


Fig. 3: Iteration path for the nonconvex-convex coupled problem. The addition of a coupling variable and collaborative objective terms in the GP changes the trajectory and allows for the correct minimum to be found

4.4 Observations and summary

These numerical examples illustrate two principal roles of nested GP subproblems.

First, when GP components are feed-forward, nested optimization reduces the effective dimensionality of the outer problem and constrains the nonlinear search to a manifold of GP-optimal designs. The outer problem becomes univariate in the two-dimensional example, and the nested GP

Table 3: Solution characteristics of the nonconvex–convex coupled optimization. Run time was calculated using the median statistic provided by BenchmarkTools.jl, with over 200 samples for each method and a fixed starting point

	Nonlinear	Nested GP	Corrected GP
Starting point	[4.7,2.0]	—	—
End point	[5.50, 4.93]	[4.14, 5.67]	[5.50, 4.93]
Function calls	11	7	9
Major iterations	8	4	7
Minor iterations	10	5	9
Wall time (ms)	1.487	8.769	11.265
Objective	4.2362	7.5682	4.2362

converges to the correct optimum without target variables or consistency treatments. Iterations were significantly reduced.

Second, when GP and nonlinear components are mutually coupled, naive nesting fails. Introducing GP-compatible consistency terms restores correctness while preserving the structured sensitivity information supplied by the convex subproblem.

Both effects persist in higher-dimensional problems and motivate the use of nested GP subproblems wherever GP-compatible structure can be identified and reformulated.

5 Geometric Program for Wind Turbine Tower Design

We now demonstrate the nested GP framework on a practical engineering problem: wind turbine tower design. Full wind turbine optimization involves tightly coupled aerodynamic, structural, and economic models and is inherently nonlinear. However, the tower subproblem admits a feed-forward, parameterized geometric program structure: for fixed system-level quantities, tower mass can be minimized subject to structural and dynamic constraints without introducing feedback into the aerodynamic or cost models.

In common system-level objectives such as mass normalized by annual energy production (m/AEP) or cost of energy (COE), reducing tower mass is strictly desirable and does not introduce competing optima. Consequently, the tower can be optimized independently as an inner convex subproblem whose solution informs the outer nonlinear optimization.

In this section, we derive a GP for the tower design based primarily on the TowerSE model (Ning 2018). The GP minimizes tower mass subject to constraints on stress, buckling-related quantities, natural frequency, and fatigue. The resulting formulation can be embedded as a differentiable, feed-forward subproblem within a larger wind turbine design optimization.

The GP is parameterized by tower height h , rotor–nacelle assembly mass m_{RNA} , and rotor thrust and torque loads T and Q , obtained from aerodynamic and cost models (Laxson 2006). These quantities enter strictly as parameters; no tower design variables feed back into the aerodynamic or system-level models considered here.

5.1 Tower geometry and discretization

The tower is modeled as a sequence of tapered cylindrical shell segments, discretized along the height. Each segment is characterized by its mean radius r , wall thickness t , and segment height h . The cross-sectional area of each annulus is

$$A = 2\pi r t. \quad (24)$$

Assuming a thin-walled shell (diameter-to-thickness ratio exceeding 10:1), the second moment of area is approximated by

$$I = \pi r^3 t, \quad (25)$$

consistent with standard thin-shell theory and industry practice.

To obtain a GP-compatible expression for segment volume, we approximate each tapered segment by the average of the top and bottom cylindrical shells. This yields

$$V = \pi h (r_1 t_1 + r_2 t_2), \quad (26)$$

which is exact for cylindrical segments and introduces only second-order error for modest taper ratios while preserving posynomial structure.

5.2 Stress components

Using the geometric relations above, the dominant stress components can be written in GP-compatible form.

Axial stress arises from the weight of the rotor–nacelle assembly and the tower segments above the section of interest:

$$\sigma_{\text{axial}} = \frac{F_{\text{axial}}}{A} = \frac{m_{\text{RNA}}g + \rho g V_{\text{above}}}{A}. \quad (27)$$

Bending stress is given by

$$\sigma_{\text{bending}} = \frac{Mc}{I}, \quad (28)$$

where M is the applied bending moment and $c = r + t/2$ is the distance to the outer surface. This expression induces a posynomial inequality constraint.

Shear stress follows directly as

$$\tau = \frac{V}{A}, \quad (29)$$

where applied shear force from rotor thrust is treated as a parameter.

Together, these relations provide the local stress measures required for strength and fatigue constraints.

5.3 Natural frequency constraint

Dynamic stability requires that the tower’s first natural frequency avoid resonance with the rotor’s excitation frequencies. Approximating the tower as a cantilever beam with an end mass, the first natural frequency is modeled as

$$\omega_n = \sqrt{\frac{k}{m_{\text{RNA}} + 0.23m_{\text{tower}}}}, \quad (30)$$

where $k = 3EI_{\text{avg}}/h^3$ and I_{avg} is the average second moment of area along the height.

To preserve GP structure, we introduce an auxiliary equivalent mass variable m_{eq} and enforce

$$m_{\text{eq}} \geq m_{\text{RNA}} + 0.23m_{\text{tower}}, \quad \omega_n = \sqrt{\frac{k}{m_{\text{eq}}}}. \quad (31)$$

For a uniformly discretized tower, the arithmetic mean of the sectional inertias is conservatively bounded by

$$I_{\text{avg}} \geq \frac{\sum I_k + I_{\text{base}}}{N_{\text{div}} + 1}, \quad (32)$$

which maintains GP compatibility. Alternatively, a geometric mean may be used, though this tends to underestimate stiffness:

$$I_{\text{geoavg}} = \left(\prod_{k=1}^{N_{\text{div}}} I_k \right)^{\frac{1}{N_{\text{div}}+1}}. \quad (33)$$

In practice, a safety factor $\gamma > 1$ is applied and the constraint $\omega_n \in (\gamma\omega_{1P}, \omega_{3P}/\gamma)$ is enforced to avoid resonance with the rotor’s 1P and 3P harmonics.

5.4 Fatigue damage constraint

Fatigue damage constraints are adapted from TowerSE and reformulated into GP-compatible form. The geometric variables are rescaled to millimeters to match the empirical fatigue model:

$$r_{\text{mm}} = 10^3 r, \quad t_{\text{mm}} = 10^3 t, \quad (34)$$

with corresponding inertia

$$I_{\text{mm}} = \pi r_{\text{mm}}^3 t_{\text{mm}}. \quad (35)$$

The weld factor satisfies

$$C_{\text{weld}} \leq 1.0, \quad C_{\text{weld}} \leq \left(\frac{25.0}{t_{\text{mm}}} \right)^{0.25}. \quad (36)$$

The allowable fatigue life is

$$N_f = \left(\frac{\sigma_{\text{max}} C_{\text{weld}} / \eta}{M_{\text{DEL}} r / I_{\text{mm}}} \right)^m, \quad (37)$$

and cumulative damage is constrained by

$$\text{damage} = \frac{N_{\text{DEL}}}{N_f} \leq 1.0, \quad (38)$$

where M_{DEL} , N_{DEL} , η , and m follow the TowerSE fatigue definitions (Ning 2018).

5.5 Summary

Through these reformulations, the wind turbine tower design problem is expressed as a geometric program minimizing tower mass subject to structural, dynamic, and fatigue constraints. This GP depends on system-level quantities only through parameters and therefore acts as a feed-forward, differentiable inner optimization. It can be embedded directly within the nested architecture of Section 3 to support gradient-based wind turbine design optimization.

6 Wind turbine design optimization with a nested tower design geometric program

In Section 4 we used small numerical examples to illustrate how nested GP subproblems can reduce nonlinear iteration count and support the tight convergence needed for accurate derivative propagation. With the tower geometric program developed in Section 5, we now evaluate the method on a tractable but representative wind turbine design optimization problem that captures dominant couplings among rotor aerodynamics, tower structural design, and system cost.

Our objective is to minimize cost of energy (COE), which rewards higher annual energy production (AEP) while penalizing capital and operating costs. The rotor and tower designs are coupled through loads and mass: rotor thrust and torque determine tower demands, and the rotor–nacelle assembly (RNA) mass contributes to tower axial loading and dynamic behavior. In the formulation considered here, tower design variables do not affect the aerodynamic or cost models directly; consequently, the tower subproblem enters strictly as a feed-forward, parameterized GP and does not require target variables or consistency constraints.

6.1 Problem definition

Rotor design variables and analysis

The rotor is discretized into N_{blade} radial analysis sections. At each section, design variables include chord and twist, while airfoil designations and relative radial locations are held fixed. The baseline NREL 5-MW definition specifies the design at 17 radial locations (Jonkman et al. 2009). We fit an Akima spline (Akima 1970) to enable interpolation and allow the number of analysis sections to vary without changing the baseline geometry description. The blade length is allowed to vary, along with chord and twist at each analysis location.

Aerodynamic performance metrics—including thrust T , torque Q , and AEP—are evaluated using CCBlade (Ning 2021), which implements blade element momentum theory (BEMT). AEP is computed from a Weibull wind distribution ($k = 2.0$, $\lambda = \bar{V} / \Gamma(1 + k^{-1})$), integrating over wind speeds between 5 and 25 m/s using N_{pitch} discrete bins. Each wind-speed bin has an independent pitch design variable. The total number of nonlinear (outer) design variables is therefore

$$2N_{\text{blade}} + N_{\text{pitch}} + 3 \quad (39)$$

corresponding to chord, twist, pitch schedule, tower height, rotor diameter, and tip-speed ratio.

Rotor mass and RNA mass are estimated using cost-and-scaling models (Laxson 2006) as functions of rotor diameter and operating metrics.

Tower design variables and GP subproblem

On the tower side of the problem, we vary the number of tower discretizations, N_{tower} . Each discretization has an associated section mass, and each end of the section has a specified diameter and thickness. There are therefore $2N_{\text{tower}} + 2$ design variables in the tower problem. The GP formulation requires additional design variables for analysis and constraint definition purposes (Section 5); for the fully nonlinear formulation, we instead evaluate constraints explicitly where possible rather than replicating the GP’s auxiliary structure.

Under the nested GP architecture, the tower subproblem is formulated as a mass-minimizing geometric program parameterized by tower height and loading distribution. These loads are input parameters determined from the rotor mass and aerodynamic forces. Tower design constraints were outlined in Section 5. With the NREL cost model (Laxson 2006), COE increases monotonically with tower mass, ensuring consistency between the nested objective reformulation and the original nonlinear objective.

Nested optimization

The nested optimization can be summarized schematically as

$$\begin{aligned}
 \text{min.} \quad & \text{COE} = \frac{\text{Cost}}{\text{AEP}} & (40) \\
 \text{var.} \quad & X_{\text{rotor}} \equiv \left\{ \begin{array}{l} \text{chord}(N_{\text{blade}}, \text{twist}(N_{\text{blade}}), \\ \text{pitch}(N_{\text{pitch}}), R_{\text{rotor}}, \lambda_{\text{tsr}} \end{array} \right\} \\
 & h_{\text{tower}}, \\
 \text{s. t.} \quad & [T, Q, \text{AEP}] = \text{BEMT}(X_{\text{rotor}}, h_{\text{tower}}) \\
 & m_{\text{RNA}} = \text{ScalingModel}(X_{\text{rotor}}, \text{AEP}) \\
 & x_{\text{tower}}^* = \text{GPsolve}(m_{\text{RNA}}, h_{\text{tower}}, T, Q) \\
 & \text{Cost} = \text{CostModel}(X_{\text{rotor}}, x_{\text{tower}}^*, \text{AEP}) \\
 \text{where} \quad & x_{\text{tower}} \equiv \{m_{\text{tower}}, r(N_{\text{tower}}), t(N_{\text{tower}})\}
 \end{aligned}$$

Here, X_{rotor} collects the rotor sectional variables, pitch schedule, and tip speed ratio λ_{tsr} . GPsolve denotes the tower mass-minimizing GP of Section 5. CostModel follows the NREL family of cost models (Laxson 2006) and includes estimates for balance of station, land lease, levelized replacement, operation and maintenance, and capital costs; the ScalingModel for the RNA mass estimation comes from the same family of models. AEP is the annual energy production measured in kWh and is calculated (along with rotor thrust T and torque Q) by blade element momentum theory (BEMT) as implemented in CCBlade. Design variables and CCBlade parameters are summarized in Table 4.

6.2 Study design

We compare two optimization strategies. The nonlinear (NL) method uses SNOPT and solves the entire problem directly. For the nested GP, SNOPT manages the outer optimization while the tower GP subproblem is solved as a parameterized GP using Clarabel through MOI. These strategies are compared to assess the benefit of replacing a portion of the nonlinear design space with a tightly convergent convex subproblem.

We report results from over 1,500 optimization runs comparing NL and nested-GP architectures. Unless otherwise stated, the rotor analysis uses 41 wind-speed bins (41 pitch variables) and the baseline 17 blade sections from the NREL 5-MW definition. The GP solver tolerance was fixed at 10^{-10} in all studies; experimentation showed that Clarabel solver performance had negligible dependence on tolerance level for these problems.

To evaluate robustness and convergence behavior, we vary the SNOPT optimality tolerance from 10^{-2} to 10^{-7} . At each tolerance, we perform 30 random-start optimizations by perturbing the initial values of all design variables by a uniform random perturbation in the range $\pm 0\%$ to $\pm 10\%$. We report

Table 4: Optimization variables and parameters

Variable	Dimensionality	Value Range
Rotor Variables and CCBlade Inputs		
Hub height	1	100–120 m
Rotor diameter	1	60–120 m
Relative radial location*	N_{blade}	0–1
Chord	N_{blade}	0.2–3 m
Twist	N_{blade}	0–45°
Pitch angle	N_{pitch}	0–3°
Tip speed ratio	1	2–7
Wind Speed*	N_{pitch}	5–25 m/s
Precone*	1	2.5°
Yaw*	1	0°
Tilt*	1	5°
Shear exponent*	1	0.15
Tower Variables		
Height	1	= Hub Height
Diameter	$N_{\text{tower}} + 1$	2.0–6.2 m
Thickness	$N_{\text{tower}} + 1$	5 mm–0.5 m
Modulus*	1	210×10^9 Pa
Density*	1	8500 kg/m^3
Strength*	1	450×10^6 Pa
* Fixed parameter		

median statistics and interquartile ranges to reflect variation across initializations and optimization trajectories.

6.3 Effect of optimality tolerance

Figure 4 shows solve time as a function of SNOPT optimality tolerance. At tolerances of 10^{-4} and looser, the nested-GP and NL methods have overlapping interquartile ranges, indicating similar time-to-convergence for each tolerance level. As the tolerance tightens, the nested-GP method becomes significantly faster than the fully nonlinear approach.

Figure 5 compares objective value against computation time. Across tolerances, the nested-GP method reaches lower objective values in substantially less time. Even at the tightest tolerance (10^{-7}), the NL method typically requires more than twice the computation time while converging to higher (lower-quality) objective values.

Table 5 summarizes optimizer statistics at the standard tolerance 10^{-6} . The nested-GP method attains a slightly lower COE, with a modestly higher AEP supported by a heavier tower. The fully nonlinear method requires substantially more time and exhibits greater variability, as reflected in the difference between mean and maximum solve times.

Table 5: Comparison of median optimizer statistics at standard tolerance (10^{-6})

Statistic	Nonlinear (SNOPT)	Nested GP
Optimality tolerance	10^{-6}	10^{-6}
Objective value	7.466e-2	7.452e-2
AEP (kWh)	1.008e7	1.011e7
Tower Mass (kg)	216,060	250,951
Major iterations	2,047	326
Function calls	11,311	1,407
Mean solve time (s)	1,270	207
Max time (s)	5,108	611

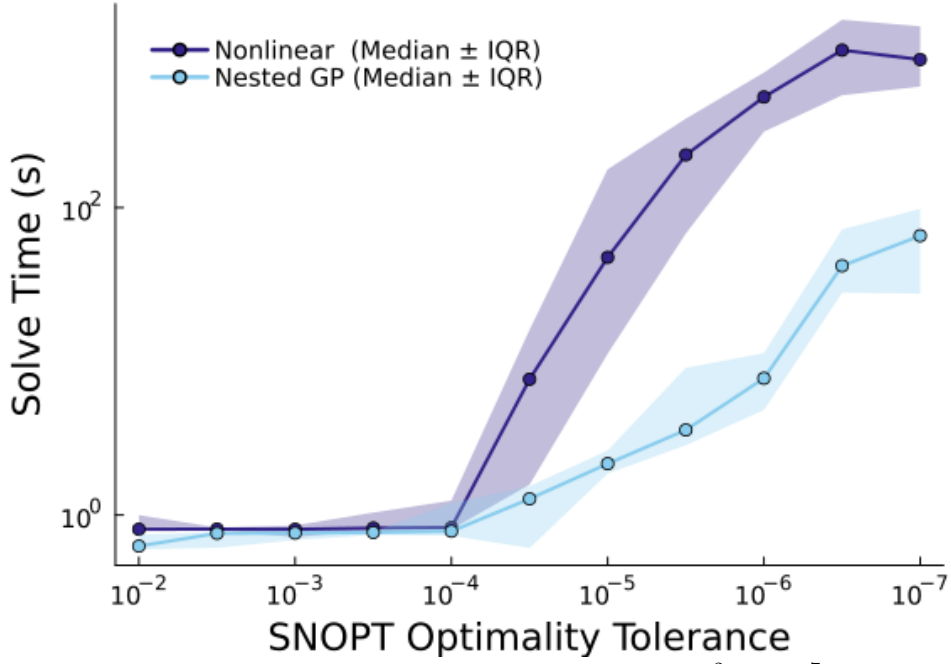


Fig. 4: Solve time as a function of SNOPT optimality tolerance (10^{-2} to 10^{-7}). GP tolerance fixed at 10^{-10} ; 41 pitch variables, 17 blade sections, and 9 tower divisions. Corresponds to 78 NL variables and 20 GP variables

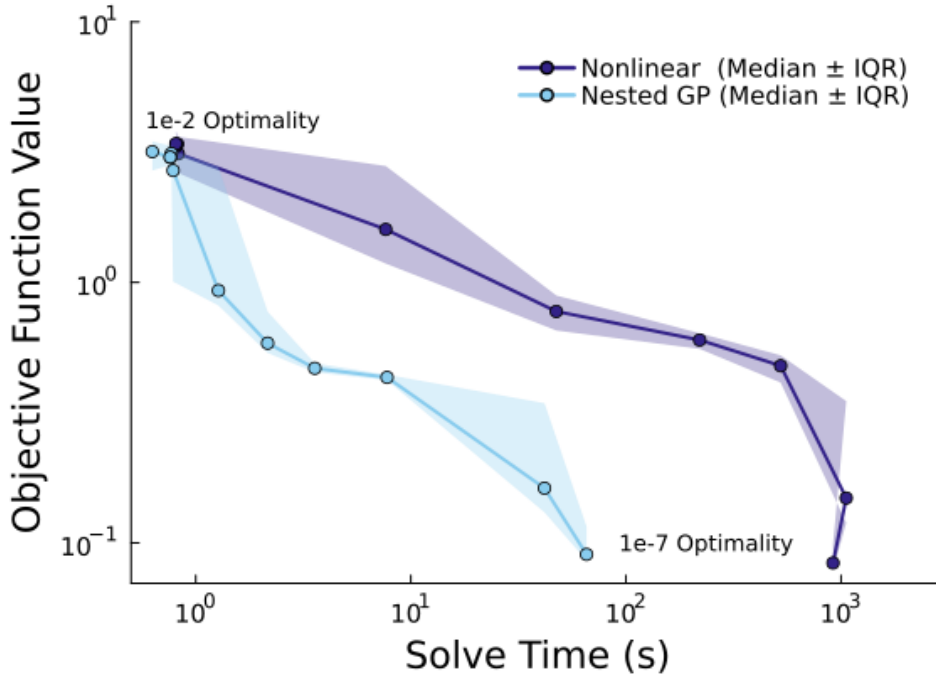


Fig. 5: Comparison of solve time and objective value between nonlinear and nested-GP methods. Optimality tightens from 10^{-2} (upper-left) to 10^{-7} (bottom-right). Points correspond to median solve time at each tolerance

6.4 Effect of problem dimensionality

We next evaluate how performance changes as the number of nonlinear and GP variables increases.

Increasing nonlinear dimensionality

Figure 6 varies the number of pitch variables (11, 41, 71, 101) while holding the tower discretization fixed at 9 divisions and the blade discretization fixed at 17 sections. As nonlinear dimensionality increases, the nested-GP method becomes increasingly efficient relative to the NL method: its interquartile range remains consistently below that of the nonlinear optimizer for all but the smallest case. Similar qualitative trends were observed when increasing the number of blade analysis sections.

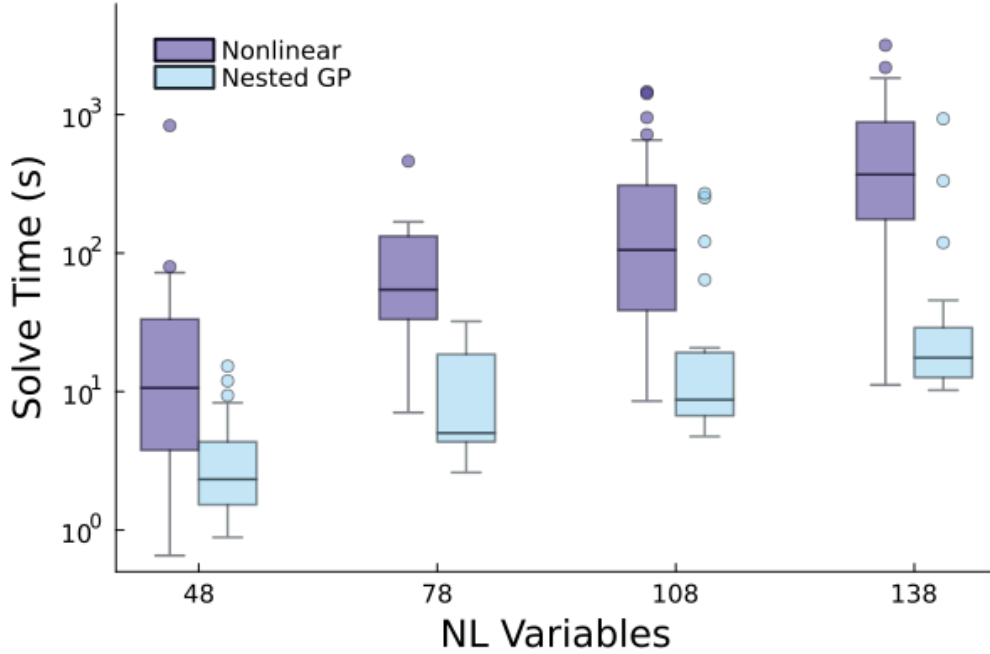


Fig. 6: Effect of increasing the number of pitch variables [11, 41, 71, 101] with 9 tower divisions (20 GP design variables). Each box summarizes 200 optimizations. Blade discretization fixed at 17 sections (34 blade variables)

Increasing GP dimensionality

Figure 7 varies the number of tower discretizations (3, 9, 18, 27) while holding rotor design dimensionality fixed (41 pitch variables, 17 blade sections). As the GP grows, the relative advantage of nesting diminishes. This behavior is consistent with the increasing cost of repeatedly solving and differentiating through the GP: solution and differentiation time increases rapidly with GP size, while the reduction in outer nonlinear iterations is comparatively modest. These results indicate that nested GP architectures are most effective when the convex subproblem remains moderate in size relative to the outer nonlinear design space.

6.5 Summary

Across this problem class, nested GP architectures achieve substantial run time savings by reducing nonlinear iteration count while preserving or improving solution quality. The benefit increases as the nonlinear portion of the design space grows, but decreases as the GP subproblem becomes large enough that repeated GP solves and implicit-differentiation operations dominate run time. This tradeoff suggests that nested GP methods are most effective when GP-compatible substructures are present and can be kept modest in size relative to the full nonlinear problem.

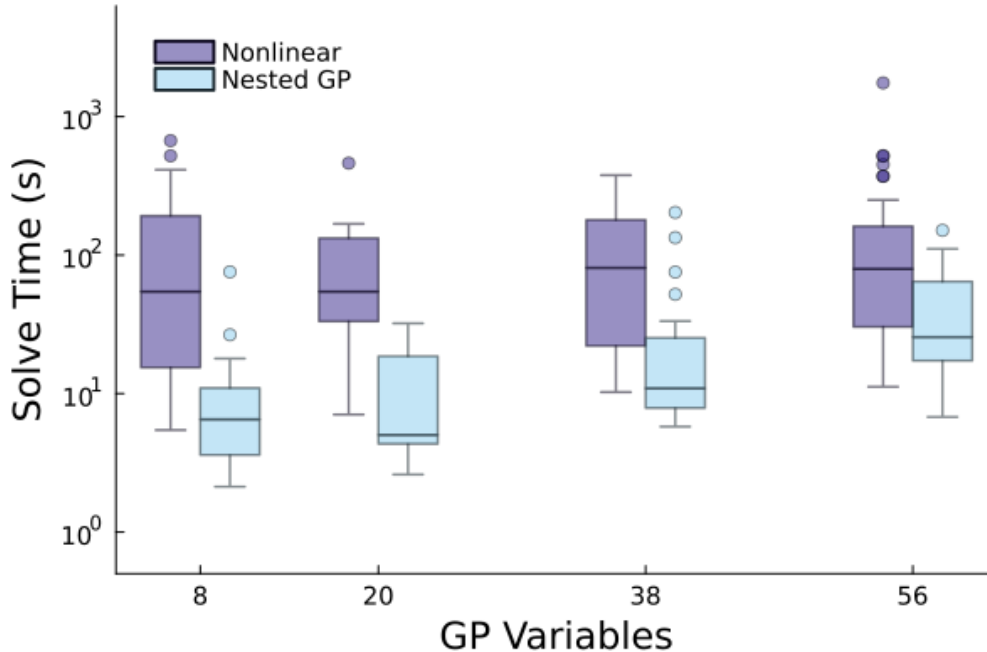


Fig. 7: Effect of increasing the number of tower discretizations [3, 9, 18, 27] at fixed 41 pitch variables and 17 blade discretizations (78 NL design variables). Larger GP subproblems reduce the relative advantage of the nested-GP approach

7 Conclusion

Nonlinear optimization remains central to modern mechanical design because it accommodates complex, high-fidelity models and a wide variety of problem structures. Convex optimization, by contrast, offers reliable convergence and global optimality guarantees, but only when the governing models possess the required mathematical structure.

Most engineering design problems cannot be wholly reformulated as a geometric program. This work showed that a differentiable geometric programming framework enables *partial* GP reformulation by embedding GP subproblems inside a larger nonlinear optimization. In this nested architecture, the GP acts as a convex subsolver that (i) removes GP-compatible variables from the outer search, reducing effective dimensionality in feed-forward settings, and (ii) provides tightly converged solutions and sensitivities that support accurate gradient-based optimization of the full system.

Across wind turbine optimization studies, the nested GP architecture consistently reduced nonlinear iteration count and, at tighter convergence tolerances, produced lower COE solutions in substantially less time than a conventional SNOPT-only formulation. Observed speedups depended strongly on problem structure: gains were modest when nonlinear analysis was inexpensive or the GP subproblem dominated run time, and largest when nonlinear evaluations were costly while the GP remained moderate in size. In the best cases we observed speedups approaching two orders of magnitude, with 6–10× speedups typical across the tested configurations.

Several challenges remain for broader use of convex optimization in engineering design. First, nested GPs are simplest in feed-forward settings; when bidirectional coupling is present, additional coordination is required to maintain consistency between GP and nonlinear components. Here we demonstrated a GP-compatible penalty construction, but alternative coupling strategies may offer improved stability and scalability. Second, building GP-compatible models remains labor-intensive. The tower GP in this work was derived by hand, and automating such reformulations would substantially lower the barrier to adoption.

Overall, these results indicate that nested, differentiable GP subproblems can provide a practical path toward scalable nonlinear–convex design optimization by exploiting convex substructure where it exists, without requiring wholesale reformulation of the full engineering model.

Declarations

Funding The material presented in this paper is based on work supported by the National Aeronautics and Space Administration under award number 80NSSC23M0218.

Conflict of Interest The authors declare that they have no conflicts of interest.

Replication of results The core differentiable geometric programming method is implemented as part of ImplicitAD and is available in the open-source repository (<https://github.com/byuflowlab/ImplicitAD.jl>) The scripts used to generate results and figures from the paper are available in a separate repository (<https://github.com/byuflowlab/nakamoto2026-ndgp>).

References

- Agrawal, A., Barratt, S., Boyd, S., Stellato, B.: Learning convex optimization control policies. In: Bayen, A.M., Jadbabaie, A., Pappas, G., Parrilo, P.A., Recht, B., Tomlin, C., Zeilinger, M. (eds.) Proceedings of the 2nd Conference on Learning for Dynamics and Control. Proceedings of Machine Learning Research, vol. 120, pp. 361–373 (2020). <https://doi.org/https://proceedings.mlr.press/v120/agrawal20a.html>
- Akima, H.: A new method of interpolation and smooth curve fitting based on local procedures. Journal of the Association for Computing Machinery **17**(4), 589–602 (1970) <https://doi.org/10.1145/321607.321609>
- ApS, M.: MOSEK Optimizer API for Julia 11.0.30. (2025). <https://docs.mosek.com/latest/juliaapi/index.html>
- Bell, B.M., Burke, J.V.: Algorithmic differentiation of implicit functions and optimal values. In: Bischof, C.H., Bücker, H.M., Hovland, P., Naumann, U., Utke, J. (eds.) Advances in Automatic Differentiation, pp. 67–77. Springer, Berlin, Heidelberg (2008)
- Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-Lopez, F., Pedregosa, F., Vert, J.-P.: Efficient and modular implicit differentiation. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 5230–5242 (2022)
- Burnell, E., Hoburg, W.: GPkit software for geometric programming. <https://github.com/convengineering/gpkit>. Version 0.7.0 (2018)
- Burton, M., Hoburg, W.: Solar and gas powered long-endurance unmanned aircraft sizing via geometric programming. Journal of Aircraft **55**(1) (2018)
- Boyd, S., Kim, S.-J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optimization and Engineering **8**(1), 67–127 (2007) <https://doi.org/10.1007/s11081-007-9001-7>
- Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, The Edinburgh Building, Cambridge, CB2 8RU, UK (2004)
- Canitez, F.: Urban public transport systems from new institutional economics perspective: a literature review. Transport Reviews **39**(4), 511–530 (2018) <https://doi.org/10.1080/01441647.2018.1552631>
- Calafiore, G.C., El Ghaoui, L.M., Novara, C.: Sparse identification of posynomial models. Automatica **59**, 27–34 (2015) <https://doi.org/10.1016/j.automatica.2015.06.003>
- Chen, J., Revels, J.: Robust benchmarking in noisy environments. arXiv e-prints (2016)
- Cole, B., Traykovski, P.: Geometric programming for aerodynamically-actuated wingsail design optimization. IEEE Journal of Oceanic Engineering **50**(3), 2063–2089 (2025) <https://doi.org/10.1109/>

- Duffin, R.J., Peterson, E.L., Zener, C.: Geometric Programming. Wiley, New York (1967)
- Goulart, P.J., Chen, Y.: Clarabel: An interior-point solver for conic programs with quadratic objectives (2024)
- Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R.S., Guo, E.: On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization (2016). <https://arxiv.org/abs/1607.05447>
- Gill, P.E., Murray, W., Saunders, M.A.: Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review* **47**(1), 99–131 (2005) <https://doi.org/10.1137/S0036144504446096>
- Hoburg, W., Abbeel, P.: Geometric programming for aircraft design optimization. *AIAA Journal* **52**(11), 2414–2426 (2014) <https://doi.org/10.2514/1.J052732>
- Herber, D.R., Allison, J.T.: Nested and simultaneous solution strategies for general combined plant and control design problems. *Journal of Mechanical Design* **141**(1) (2019) <https://doi.org/10.1115/1.4040705>
- Hwang, J.T., Lee, D.Y., Cutler, J.W., Martins, J.R.R.A.: Large-scale multidisciplinary optimization of a small satellite’s design and operation. *Journal of Spacecraft and Rockets* **51**(5), 1648–1663 (2014) <https://doi.org/10.2514/1.A32751> <https://doi.org/10.2514/1.A32751>
- Jonkman, J., Butterfield, S., Musial, W., Scott, G.: Definition of a 5-mw reference wind turbine for offshore system development. Technical report, National Renewable Energy Laboratory (February 2009). <https://doi.org/10.2172/947422>
- Jovane, F., Yoshikawa, H., Alting, L., Boër, C.R., Westkamper, E., Williams, D., Tseng, M., Seliger, G., Paci, A.M.: The incoming global technological and industrial revolution towards competitive sustainable manufacturing. *CIRP Annals* **57**(2), 641–659 (2008) <https://doi.org/10.1016/j.cirp.2008.09.010>
- Karush, W.: Minima of functions of several variables with inequalities as side conditions. PhD thesis, Thesis (S.M.)—University of Chicago, Department of Mathematics, December 1939. (1939)
- Karcher, C., Haimes, R.: A method of sequential log-convex programming for engineering design. *Optimization and Engineering* **24**(3), 1719–1745 (2023) <https://doi.org/10.1007/s11081-022-09750-3>
- Kim, D., Seth, A., Liem, R.P.: Signomial programming for airfoil shape optimization with geometric constraints. In: *AIAA SciTech 2025 Forum* (2025). <https://doi.org/10.2514/6.2025-0653> . AIAA
- Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Neyman, J. (ed.) *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, vol. 2, pp. 481–492 (1951)
- Laxson, L.F.M.H.A.: Wind turbine cost and scaling model. Technical report, National Renewable Energy Laboratory (2006)
- Li, K., Cheng, G.: A nested algorithm of truss topology optimization for maximum plastic shakedown loading capacity. *Journal of Computational Design and Engineering* **9**(2), 670–688 (2022) <https://doi.org/10.1093/jcde/qwac022>
- Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., Vielma, J.P.: JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation* (2023) <https://doi.org/10.1007/s12532-023-00239-3>
- Legat, B., Dowson, O., Garcia, J.D., Lubin, M.: MathOptInterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing* (2021) <https://doi.org/10.1287/ijoc.2021.1067>

- Mihai, D.-M.: The world electricity production and the current global energy crisis in brief. *Ovidius University Annals* **23**(2), 292–299 (2023)
- Manios, S.E., Lagaros, N.D., Nassiopoulos, E.: Nested topology optimization methodology for designing two-wheel chassis. *Frontiers in Built Environment* **5** (2019) <https://doi.org/10.3389/fbuil.2019.00034>
- Ning, A.: TowerSE. Version 0.1.1 (2018). <https://github.com/WISDEM/TowerSE>
- Ning, A.: Using blade element momentum methods with gradient-based design optimization. *Structural and Multidisciplinary Optimization* **64**(2), 991–1014 (2021) <https://doi.org/10.1007/s00158-021-02883-6>
- Stanford, B.K., Jutte, C.V., Coker, C.A.: Sizing and Layout Design of an Aeroelastic Wingbox through Nested Optimization, (2018). <https://doi.org/10.2514/6.2018-2212>
- Simpson, T.W., Martins, J.R.R.A.: Multidisciplinary design optimization for complex engineered systems: Report from a national science foundation workshop. *Journal of Mechanical Design* **133**(10), 101002 (2011) <https://doi.org/10.1115/1.4004465>
- Sterling, T.: Models of computation — enabling exascale. *The International Journal of High Performance Computing Applications* **23**(4), 332–334 (2009)
- Udell, M., Mohan, K., Zeng, D., Hong, J., Diamond, S., Boyd, S.: Convex optimization in Julia. In: 2014 First Workshop for High Performance Technical Computing in Dynamic Languages, pp. 18–28 (2014). <https://doi.org/10.1109/HPTCDL.2014.5>
- York, M.A., Öztürk, B., Burnell, E., Hoburg, W.W.: Efficient aircraft multidisciplinary design optimization and sensitivity analysis via signomial programming. *AIAA Journal* **56**(11), 4546–4561 (2018) <https://doi.org/10.2514/1.J057020>