Optimization and Engineering (2023) 24:1719–1745
https://doi.org/10.1007/s11081-022-09750-3

RESEARCH ARTICLE

A method of sequential log-convex programming for engineering design

Cody Karcher[1] · **Robert Haimes**[1]

Received: 20 January 2022 / Revised: 4 July 2022 / Accepted: 4 July 2022 /
Published online: 25 July 2022

Abstract
A method of Sequential Log-Convex Programming (SLCP) is constructed that exploits the log-convex structure present in many engineering design problems. The mathematical structure of Geometric Programming (GP) is combined with the ability of Sequential Quadratic Program (SQP) to accommodate a wide range of objective and constraint functions, resulting in a practical algorithm that can be adopted with little to no modification of existing design practices. Three test problems are considered to demonstrate the SLCP algorithm, comparing it with SQP and the modified Logspace Sequential Quadratic Programming (LSQP). In these cases, SLCP shows up to a 77% reduction in number of iterations compared to SQP, and an 11% reduction compared to LSQP. The airfoil analysis code XFOIL is integrated into one of the case studies to show how SLCP can be used to evolve the fidelity of design problems that have initially been modeled as GP compatible. Finally, a methodology for design based on GP and SLCP is briefly discussed.

Keywords Geometric programming · Log-convexity · Non-linear programming · Sequential quadratic programming · Sequential convex programming

1 Optimization for engineering design

Two of the defining tasks of the engineering profession are *analysis* and *design*. Analysis is a process by which engineers obtain data to quantify the behavior of a part, component, or system. A full scale build and test is always the preferred approach to obtaining the highest quality analysis data, but external factors like

✉ Cody Karcher
ckarcher@mit.edu

Robert Haimes
haimes@mit.edu

[1] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

cost and schedule make it impossible to utilize these methods every time an analysis must be performed.

As a result, analysis is often performed via *simulation*, where a set of simplified physics is modeled mathematically and then used to obtain the necessary analysis data. Given the abundance of computing resources in the modern era, simulation has become the standard method for performing engineering analysis. Thousands of analysis models (often distributed as software or code packages) have been released, including models for computational fluid dynamics (CFD), finite element analysis (FEA), electromagnetic simulation, and thermal analysis, just to name a few.

In general these analysis models are highly complex, requiring great expertise both to develop and to run as a user. Due to this complexity, analysis is often conceptualized and implemented as a *black box*, where the user provides inputs and obtains outputs with no visibility into the actual simulation being run (Martins and Lambe 2013). In this way, analysis tools can be thought of as functions:

$$\mathbf{y} = f(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ represents the inputs to the analysis and $\mathbf{y}$ represents the obtained outputs. In essence, the vector $\mathbf{x}$ is the mathematical abstraction of the design of a particular engineering system, and vector $\mathbf{y}$ is how that system performs.

If the goal of analysis is to determine the performance $\mathbf{y}$ of some given system design $\mathbf{x}$, the *design* process seeks to determine some vector $\mathbf{x}$ that satisfies performance criteria $\mathbf{y}$. The first step in design is often to simplify the vector $\mathbf{x}$ to include only the most critical parameters that define the system. Once complete, the vector $\mathbf{x}$ exists in a vector space $\mathbb{R}^N$ (called the *design space*) where N is the number of design decisions that have been retained in $\mathbf{x}$, typically referred to as the *design variables*. Determining the appropriate values for these design variables is a challenging task because while analysis can be performed with a single "function call" to the black box method in Eq. 1, determining an appropriate design requires that many candidate designs be evaluated in order to determine the best one, $\mathbf{x}^*$.

Thus, design does not take the simple functional form of Eq. 1. Instead, engineering design problems cast in the language of mathematics are optimization problems:

$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) \tag{2}$$

But engineering design problems rarely take the form of Eq. 2. Instead, *constraints* are often imposed on the problem:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \; f(\mathbf{x}) \\ &\text{subject to } \mathbf{g}_i(\mathbf{x}) \leq 0, \; i = 1, \ldots, N \\ &\qquad\qquad \mathbf{h}_j(\mathbf{x}) = 0, \; j = 1, \ldots, M \end{aligned} \tag{3}$$

Constraints typically serve one of two purposes. First is to impose an artificial limit on the system, such as a minimum dimension or a maximum cost. Second is to represent a limit imposed by a fundamental law of physics, such as the maximum stress that can be carried in a material or the dynamics of Newton's Second Law.

These physics based constraints are another way in which analysis models are often included in engineering design problems, and so functions $f(\mathbf{x})$, $\mathbf{g}_i(\mathbf{x})$, and $\mathbf{h}_j(\mathbf{x})$ must all be assumed in the general case to be highly complicated black box functions that are expensive to evaluate. The optimization methods used most commonly in engineering design are therefore tailored to minimize the number of times the objective and constraint functions must be evaluated. Of these methods, Sequential Quadratic Programming (SQP), Geometric Programming (GP), and Logspace Sequential Quadratic Programming (LSQP) are most relevant to this work.

2 Foundations in existing optimization algorithms

2.1 Sequential quadratic programming

In general, the problem posed in Eq. 3 is referred to as a Non-Linear Program (NLP) and is difficult to solve. Many algorithms exist for solving NLPs, but the Sequential Quadratic Programming (SQP) algorithm is highly effective and has been utilized across a wide range of scientific and engineering fields. The SQP algorithm begins with an initial guess $\mathbf{x}_k$ and formulates a Quadratic Programming (QP) approximation of the NLP that is valid in the local region near $\mathbf{x}_k$. The QP sub-problem takes the form (Boggs and Tolle 1996; Nocedal and Wright 2006; Kraft 1988):

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \;& f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d} + \tfrac{1}{2}\mathbf{d}^T \nabla^2 \mathcal{L}(\mathbf{x}_k)\mathbf{d} \\ \text{subject to} \;& g_i(\mathbf{x}_k) + \nabla g_i(\mathbf{x}_k)^T \mathbf{d} \leq 0, \; i = 1, \ldots, N \\ & h_j(\mathbf{x}_k) + \nabla h_j(\mathbf{x}_k)^T \mathbf{d} = 0, \; j = 1, \ldots, M \\ & \mathbf{d} = \mathbf{x} - \mathbf{x}_k \end{aligned} \tag{4}$$

so named because of the quadratic objective function. Since QPs are known to be *convex* (Boyd and Vandenberghe 2009), the optimization problem in Eq. 4 can be used to reliably and efficiently produce a new guess $\mathbf{x}_{k+1}$. The process can then be iterated until some convergence criteria is reached, returning the optimal solution to the original NLP, $\mathbf{x}^*$.

Together with interior point methods, SQP represents the current state of the art for solving constrained continuous non-linear optimization problems (Martins and Ning 2021) despite being nearly 60 years old (Boggs and Tolle 1996). The widespread success of SQP can be traced back to two key attributes. First, the QP sub-problem is easy to construct since the sub-problem only requires the function evaluations $f(\mathbf{x}_k)$, $g_i(\mathbf{x}_k)$, and $h_j(\mathbf{x}_k)$ and the gradients $\nabla f(\mathbf{x}_k)$, $\nabla g_i(\mathbf{x}_k)$, and $\nabla h_j(\mathbf{x}_k)$. These quantities are generally simple to obtain regardless of the complexity of the true functions, making SQP applicable to a incredibly large number of problem formulations. Second, the QP sub-problem is easy to solve because it is convex, established in great detail by Boyd and Vandenberghe (2009). This convexity is key as convex optimization problems can be solved reliably and efficiently, unlike most other NLPs.

In fact, the QP sub-problem is the *best possible* convex approximation of the true NLP that can be derived from a Taylor series decomposition of functions $f(\mathbf{x})$, $\mathbf{g}_i(\mathbf{x})$, and $\mathbf{h}_j(\mathbf{x})$, since taking any more terms in either the objective or constraint approximations would result in a non-convex sub-problem. The desire for an accurate sub-problem should be relatively intuitive, since the number of iterations required to solve the NLP decreases as sub-problem accuracy increases.[1] But many other forms of convex optimization problems exist, including Linear Programs (LP), Semi-Definite Programs (SDP), Second Order Cone Programs (SOCP), and some Quadratically Constrained Quadratic Programs (QCQP), among others. If the role of the sub-problem is only to efficiently produce a new guess x_{k+1}, any one of these convex forms could easily be used in place of the QP sub-problem. These theoretical approaches are generalized under the classification of Sequential Convex Programming (SCP) (Boyd 2015; Duchi et al. 2018).

Of these SCP variations, only Sequential Quadratically Constrained Quadratic Programming (SQCQP) has received much attention in the literature (Anitescu 2002; Tang and Jian 2008; Liu et al. 2020; Jian et al. 2021), but these methods struggle with complications in computing constraint curvature and in handling non-convex QCQP sub-problems.[2] Why other forms of SCP have not been studied is not clear, but any method of SCP should abide by the following criteria:

1. Ease of construction should be comparable to the QP sub-problem of SQP (ie, use only $f(\mathbf{x}_k)$, $g_i(\mathbf{x}_k)$, and $h_j(\mathbf{x}_k)$ and the gradients $\nabla f(\mathbf{x}_k)$, $\nabla g_i(\mathbf{x}_k)$, and $\nabla h_j(\mathbf{x}_k)$ in sub-problem construction)
2. Exhibits a convex structure, and therefore easily solved
3. Captures the underlying NLP more accurately than the QP sub-problem of SQP

An LP based algorithm would rarely be superior to SQP, algorithms based on SDP or SOCP do not have an obvious construction method for the sub-problem, and the lack of convexity in some QCQPs has already been discussed. So, is it possible to develop a method of SCP that satisfies all three criteria? To answer this question one key building block remains, as recent literature has suggested a clear front runner for the type of convex optimization that should be used for engineering design: Geometric Programming.

2.2 Geometric programming

A Geometric Program (GP) is a specific type of optimization formulation built from two classes of functions: monomials and posynomials. A monomial function is defined as the product of a leading constant with each variable raised to a real power (Boyd et al. 2007):

[1] Consider as a thought experiment the extreme case where the sub-problem exactly represents the original NLP. The solution would be obtained in only one iteration.

[2] It is theoretically possible to model nearly any constrained continuous optimization as a QCQP regardless of whether convex structure exists or not, significantly limiting the general usefulness of the form.

$$m(\mathbf{x}) = c{x_1}^{a_1}{x_2}^{a_2}\ldots{x_n}^{a_n} = c\prod_{i=1}^{N} x_i^{a_i} \tag{5}$$

A posynomial is simply the sum of monomials (Boyd et al. 2007), which can be defined in notation as:

$$p(\mathbf{x}) = m_1(\mathbf{x}) + m_2(\mathbf{x}) + \ldots + m_n(\mathbf{x}) = \sum_{k=1}^{K} c_k \prod_{i=1}^{N} x_i^{a_{ik}} \tag{6}$$

From these two building blocks, it is possible to construct the definition of a GP in standard form (Boyd et al. 2007):

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \;\; p_0(\mathbf{x}) \\ &\text{subject to} \;\; m_i(\mathbf{x}) = 1, \;\; i = 1, \ldots, N \\ &\qquad\qquad p_j(\mathbf{x}) \leq 1, \;\; j = 1, \ldots, M \end{aligned} \tag{7}$$

If the general NLP (Eq. 3) can be written in GP standard form (Eq. 7) then it can be solved with great efficiency, since upon log transformation[3] geometric programs become *convex* (Boyd et al. 2007).

The advantage of the GP form is that it is far more representative of many engineering design problems than the QP formulation. The benefits of GP for engineering design has been well established in the literature (Clasen 1984; Greenberg 1995; Boyd et al. 2005; Boyd and Lee 2001; Li et al 2004; Xu et al. 2004; Jabr 2005; Chiang 2005; Chiang et al. 2007; Kandukuri and Boyd 2002; Marin-Sanguino et al. 2007; Vera et al. 2010; Preciado et al. 2014; Misra et al. 2014; Sela Perelman and Amin 2015) [see the original compilation in Agrawal et al. (2019)], and has seen specific benefit for aircraft design (Hoburg and Abbeel 2014; Torenbeek 2013; Hoburg and Abbeel 2013; Kirschen et al. 2016; Brown and Harris 2018; York et al. 2018; Burton and Hoburg 2018; Lin et al. 2020; Kirschen et al. 2018; York et al. 2018; Saab et al. 2018; Hall et al. 2018). Given that geometric programs are often more accurate at modeling engineering design problems, and that they can be solved just as efficiently as the quadratic programs that form the core of the SQP algorithm, GPs are a strong candidate for use in Sequential Convex Programming.

2.3 Logspace sequential quadratic programming

The first attempts to leverage geometric programming in a sequential optimization algorithm were simply applications of SQP under the log transformation that makes GPs convex (Kirschen et al. 2018; Karcher 2021). Consider a slight modification of the general NLP (Karcher 2021):

[3] In some of the literature, the transformation considered in this work is referred to as a log-log transformation since both dependent and independent variables are transformed. In all cases here, logspace, log-convexity, log transformation etcetera could equivalently be called log-log space, log-log convexity, log-log transformation and similar.

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \;& f(\mathbf{x}) \\ \text{subject to} \;& \mathbf{g}_i(\mathbf{x}) \leq 1,\ i = 1, \ldots, N \\ & \mathbf{h}_j(\mathbf{x}) = 1,\ j = 1, \ldots, M \end{aligned} \tag{8}$$

Under the GP transformation $y_i = \log x_i$, or equivalently $x_i = e^{y_i}$, the problem becomes:

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \;& \log f(e^{\mathbf{y}}) \\ \text{subject to} \;& \log \mathbf{g}_i(e^{\mathbf{y}}) \leq 0,\ i = 1, \ldots, N \\ & \log \mathbf{h}_j(e^{\mathbf{y}}) = 0,\ j = 1, \ldots, M \end{aligned} \tag{9}$$

which makes the new QP sub-problem (Karcher 2021):

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \;& \log f(\mathbf{x}_k) + \tfrac{1}{f(\mathbf{x}_k)} \left(\mathbf{x}_k \odot \nabla f(\mathbf{x}_k)\right)^T \mathbf{d} + \tfrac{1}{2} \mathbf{d}^T \nabla^2 \mathcal{L}(\mathbf{y}_k) \mathbf{d} \\ \text{subject to} \;& \log g_i(\mathbf{x}_k) + \tfrac{1}{g_i(\mathbf{x}_k)} \left(\mathbf{x}_k \odot \nabla g_i(\mathbf{x}_k)\right)^T \mathbf{d} \leq 0,\ i = 1, \ldots, N \\ & \log h_j(\mathbf{x}_k) + \tfrac{1}{h_j(\mathbf{x}_k)} \left(\mathbf{x}_k \odot \nabla h_j(\mathbf{x}_k)\right)^T \mathbf{d} = 0,\ j = 1, \ldots, M \\ & \mathbf{d} = \mathbf{y} - \log \mathbf{x}_k \\ & \mathbf{y} = \log \mathbf{x} \end{aligned} \tag{10}$$

The use of log-transformations with traditional SQP is a well known method of improving the scaling of the original non-linear program, and is therefore not a significant advance forward in the state of the art. However, LSQP is a general and systematic approach to non-linear optimization, based on an improved understanding of the underlying GP-compatible mathematics present in many engineering design problems (Karcher 2021).

But LSQP can be taken one step further. The sub-problem defined by Eq. 10 represents monomial constraints exactly, but gives no consideration to posynomial functions. Under transformation, a posynomial constraint becomes:

$$\log \left(\sum_j \exp \left(P_j(\mathbf{d} + \log \mathbf{x}_k) + q_j \right) \right) \leq 0 \tag{11}$$

which if inserted into Eq. 10 *leaves the sub-problem convex*. Since convex problems can be readily solved (Boyd and Vandenberghe 2009) it is possible to model these posynomial constraints directly, without resorting to the linearized form.

3 The mathematics of SLCP

3.1 Mathematical definition of the SLCP method

Consider that rather than representing the general non-linear program with Eq. 8, the constraints that are GP compatible (posynomials and monomials) are given special treatment:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}}\ & f(\mathbf{x}) \\ \text{subject to}\ & \mathbf{p}(\mathbf{x}) \leq 1 \\ & \mathbf{m}(\mathbf{x}) = 1 \\ & \mathbf{g}(\mathbf{x}) \leq 1 \\ & \mathbf{h}(\mathbf{x}) = 1 \end{aligned} \tag{12}$$

The constraints $\mathbf{g}(\mathbf{x}) \leq 1$ and $\mathbf{h}(\mathbf{x}) = 1$ will be linearized in the sub-problem just as in LSQP (Karcher 2021), but posynomials and monomials can be transformed and imposed directly in the sub-problem. Following this procedure yields the following sub-problem form:

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}}\ & \log f(\mathbf{x}_k) + \tfrac{1}{f(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla f(\mathbf{x}_k)\right)^T \mathbf{d} + \tfrac{1}{2}\mathbf{d}^T \nabla^2 \mathcal{L}_R(\mathbf{y}_k)\mathbf{d} \\ \text{subject to}\ & \log\left(\textstyle\sum_j \exp\left(P_j(\mathbf{d} + \log \mathbf{x}_k) + q_j\right)\right) \leq 0 \\ & A_m(\mathbf{d} + \log \mathbf{x}_k) + b_m \leq 0 \\ & \log g(\mathbf{x}_k) + \tfrac{1}{g(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla g(\mathbf{x}_k)\right)^T \mathbf{d} \leq 0 \\ & \log h(\mathbf{x}_k) + \tfrac{1}{h(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla h(\mathbf{x}_k)\right)^T \mathbf{d} = 0 \\ & \mathbf{d} = \mathbf{y} - \log \mathbf{x}_k \\ & \mathbf{y} = \log \mathbf{x} \end{aligned} \tag{13}$$

Solving the general non-linear program in Eq. 12 via a series of sub-problems defined by Eq. 13 is proposed here as a method of of Sequential Log-Convex Programming (SLCP), since the quadratic programing sub-problem of SQP has now been replaced with a log-convex programming (LCP) sub-problem.

3.2 The reduced Lagrangian

Though the primary distinction between the SLCP method proposed here and the LSQP algorithm in the literature (Karcher 2021) is in the handling of posynomial constraints, the objective function also requires a minor update. The LSQP sub-problem inherits its quadratic objective function directly from SQP, which utilizes the Hessian of the Lagrangian function, defined as:

$$\mathcal{L}(\mathbf{y}, \lambda) = \log f(\mathbf{x}) + \lambda \log \mathbf{p}(\mathbf{x}_k) + \lambda \log \mathbf{m}(\mathbf{x}_k) + \lambda \log \mathbf{g}(\mathbf{x}_k) + \lambda \log \mathbf{h}(\mathbf{x}_k) \tag{14}$$

The primary purpose of using the $\nabla^2 \mathcal{L}(\mathbf{x}_k)$ rather than $\nabla^2 f(\mathbf{x}_k)$ is to include some second order information from the constraints in the sub-problem (Boggs and Tolle 1996; Nocedal and Wright 2006). In the case of SLCP, some of the constraints with higher order curvature are now being represented directly, and so attempting to approximate the second order information of these constraints in the objective function causes a conflict between the approximated curvature and the true curvature that is now being fully captured. Thus, those constraints must be left out of the second order Hessian approximation.

The SLCP algorithm therefore utilizes a Reduced Lagrangian, which does not include the constraints which are exactly represented:

$$\mathcal{L}_R(\mathbf{y}, \lambda) = \log f(\mathbf{x}) + \lambda \log \mathbf{g}(\mathbf{x}_k) + \lambda \log \mathbf{h}(\mathbf{x}_k) \tag{15}$$

The use of this Reduced Lagrangian is critical to the success of the algorithm. Imposing exact constraints without this modification performs worse than strict LSQP.

3.3 Limitations and potential improvements

This proposed method of SLCP shares many of the same drawbacks as LSQP (Karcher 2021). Due to the log transformation, variables must be strictly positive, and remain strictly positive during the solve. Since it is not possible to have a negative mass, length, volume, or similar, this limitation proves remarkably non-intrusive for many engineering design problems.

Likewise, functions $f(\mathbf{x})$, $\mathbf{g}_i(\mathbf{x})$, and $\mathbf{h}_j(\mathbf{x})$ must be positive at the initial guess and stay positive throughout the solution. This concern is addressed in more depth in previous work (Karcher 2021), but essentially, it is possible to mitigate this concern through intelligent function construction, and the step size can always be constrained to ensure these functions remain positive.

One possible area for future improvement is the use of the quadratic objective function. Some effort here was given to replacing the quadratic objective with a GP compatible posynomial function, but utilizing the quadratic objective with the BFGS approximation provided the best result and so was carried over from LSQP (with the Reduced Lagrangian modification). A deep dive into modifying BFGS to accommodate a non-quadratic objective was viewed as being beyond scope.

Another compelling reason for keeping the quadratic objective in this work was that the sub-problem proposed in Eq. 13 reverts to the LSQP sub-problem in the absence of posynomial constraints, which provides a cornerstone for comparison and debugging. And since LSQP is simply an application of SQP in log transformed space (Karcher 2021), the body of literature surrounding SQP could still be utilized with only minimal modification.

A second potential improvement stems from the realization that the SLCP sub-problem defined by Eq. 13 is not the only possible SLCP sub-problem. Indeed, work by Agrawal et al. (2019) on Disciplined Geometric Programming suggests that there are families of functions beyond posynomials that are log-convex and could therefore receive similar treatment in Eq. 13. Developing a more general SLCP method that accounts for these functions will be the subject of future work, but was determined to be out of scope for this paper, especially given the success of GP form in engineering design without these additional functions.

4 An algorithm for the proposed SLCP method

Algorithm 1 outlines a method for implementing the proposed SLCP method programmatically. Though Algorithm 1 is very similar to the well known SQP algorithm and the LSQP algorithm outlined in the literature (Karcher 2021), a few minor changes should be noted.

Similar to LSQP, Algorithm 1 utilizes the gradients $\frac{\partial \log f(e^{\mathbf{y}})}{\partial y_i}$, which can be directly computed from the original gradients $\frac{\partial f}{\partial x_i}$ for minimal computational expense (Boyd et al. 2007; Karcher 2021):

$$\frac{\partial \log f(e^{\mathbf{y}})}{\partial y_i} = \frac{x_i}{f(\mathbf{x})}\frac{\partial f}{\partial x_i} \tag{16}$$

The use of the Reduced Lagrangian also necessitates modification to the damped BFGS method (Nocedal and Wright 2006):

$$\begin{aligned}
\mathbf{B}_{k+1} &= \mathbf{B}_k - \frac{\mathbf{B}_k s_k s_k^T \mathbf{B}_k}{s_k^T \mathbf{B}_k s_k} + \frac{r_k r_k^T}{s_k^T r_k} \\
s_k &= \alpha_k \mathbf{d}_x \\
z_k &= \nabla \mathcal{L}_{R_{k+1}}(f, g, h, \mathbf{y}_{k+1}, \mu_{k+1}) - \nabla \mathcal{L}_{R_k}(f, g, h, \mathbf{y}_k, \mu_{k+1}) \\
r_k &= \theta_k z_k + (1-\theta_k)\mathbf{B}_k s_k \\
\theta_k &= \begin{cases} 1 & \text{if } s_k^T z_k \geq 0.2 s_k^T \mathbf{B}_k s_k \\ (0.8 s_k^T \mathbf{B}_k s_k)/(s_k^T \mathbf{B}_k s_k - s_k^T z_k) & \text{if } s_k^T z_k < 0.2 s_k^T \mathbf{B}_k s_k \end{cases}
\end{aligned} \tag{17}$$

Finally, the sub-problem is relaxed here to handle the problem of inconsistent constraints frequently faced by SQP (Nocedal and Wright 2006):

$$\begin{aligned}
\underset{\mathbf{d}}{\text{minimize}} \quad & \log f(\mathbf{x}_k) + \tfrac{1}{f(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla f(\mathbf{x}_k)\right)^T \mathbf{d} + \tfrac{1}{2}\mathbf{d}^T \nabla^2 \mathcal{L}_R(\mathbf{y}_k)\mathbf{d} + K \textstyle\sum_{i=1}^{N_{con}} \sigma_i^2 \\
\text{subject to} \quad & \log\left(\textstyle\sum_j \exp\left(P_j(\mathbf{d} + \log \mathbf{x}_k) + q_j\right)\right) \leq \sigma_i \\
& A_m(\mathbf{d} + \log \mathbf{x}_k) + b_m \leq \sigma_i \\
& \log g(\mathbf{x}_k) + \tfrac{1}{g(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla g(\mathbf{x}_k)\right)^T \mathbf{d} \leq \sigma_i \\
& \log h(\mathbf{x}_k) + \tfrac{1}{h(\mathbf{x}_k)}\left(\mathbf{x}_k \odot \nabla h(\mathbf{x}_k)\right)^T \mathbf{d} = \sigma_i \\
& \mathbf{d} = \mathbf{y} - \log \mathbf{x}_k \\
& \mathbf{y} = \log \mathbf{x}
\end{aligned} \tag{18}$$

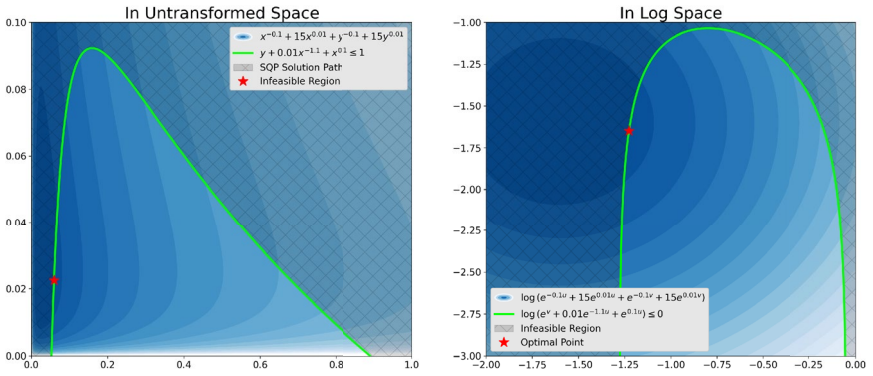
Algorithm 1 Sequential Log-Convex Programming (SLCP)

Given $\mathbf{x}_0$
Construct standard form
Compute $\mathbf{y}_0 = \log(\mathbf{x}_0)$
Initialize logspace Lagrange multipliers, $\mu_0 \leftarrow \mathbf{1}$
Initialize the matrix $\mathbf{B} \leftarrow \mathbf{I}$ the approximation of $\nabla^2 \mathcal{L}_R(\mathbf{y}, \mu)$ [40]
Compute $f(\mathbf{x}_0)$, $\mathbf{g}(\mathbf{x}_0)$, $\mathbf{h}(\mathbf{x}_0)$, $\mathbf{p}(\mathbf{x}_0)$, and $\mathbf{m}(\mathbf{x}_0)$
Compute $\log f(\mathbf{x}_0)$, $\log \mathbf{g}(\mathbf{x}_0)$, $\log \mathbf{h}(\mathbf{x}_0)$, $\log \mathbf{p}(\mathbf{x}_0)$, and $\log \mathbf{m}(\mathbf{x}_0)$
Compute $\nabla f(\mathbf{x}_0)$, $\nabla \mathbf{g}(\mathbf{x}_0)$, $\nabla \mathbf{h}(\mathbf{x}_0)$, $\nabla \mathbf{p}(\mathbf{x}_0)$, and $\nabla \mathbf{m}(\mathbf{x}_0)$
Compute $\nabla \log f(e^{\mathbf{y}_0})$, $\nabla \log \mathbf{g}(e^{\mathbf{y}_0})$, $\nabla \log \mathbf{h}(e^{\mathbf{y}_0})$, $\nabla \log \mathbf{p}(e^{\mathbf{y}_0})$, and $\nabla \log \mathbf{m}(e^{\mathbf{y}_0})$ via Equation 16
for $k = 0$ to maxIter **do**
    Solve the convex sub-problem (Equation 18) to obtain $\mathbf{d}_y$ and $\mathbf{d}_\mu$ [40]
    Compute the step size α_k via inexact line search [40,1]
    $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k + \alpha_k \mathbf{d}_y$
    $\mathbf{x}_{k+1} \leftarrow \exp(\mathbf{y}_{k+1})$
    $\mu_{k+1} \leftarrow \mu_k + \alpha_k \mathbf{d}_\mu$
    Compute $f(\mathbf{x}_{k+1})$, $\mathbf{g}(\mathbf{x}_{k+1})$, $\mathbf{h}(\mathbf{x}_{k+1})$, $\mathbf{p}(\mathbf{x}_{k+1})$, and $\mathbf{m}(\mathbf{x}_{k+1})$
    Compute $\log f(\mathbf{x}_{k+1})$, $\log \mathbf{g}(\mathbf{x}_{k+1})$, $\log \mathbf{h}(\mathbf{x}_{k+1})$, $\log \mathbf{p}(\mathbf{x}_{k+1})$, and $\log \mathbf{m}(\mathbf{x}_{k+1})$
    Compute $\nabla f(\mathbf{x}_{k+1})$, $\nabla \mathbf{g}(\mathbf{x}_{k+1})$, $\nabla \mathbf{h}(\mathbf{x}_{k+1})$, $\nabla \mathbf{p}(\mathbf{x}_{k+1})$, and $\nabla \mathbf{m}(\mathbf{x}_{k+1})$
    Compute $\nabla \log f(e^{\mathbf{y}_{k+1}})$, $\nabla \log \mathbf{g}(e^{\mathbf{y}_{k+1}})$, $\nabla \log \mathbf{h}(e^{\mathbf{y}_{k+1}})$, $\nabla \log \mathbf{p}(e^{\mathbf{y}_{k+1}})$, and $\nabla \log \mathbf{m}(e^{\mathbf{y}_{k+1}})$ via Equation 16
    Compute $\nabla \mathcal{L}_k(\mathbf{y}_k, \mu_{k+1})$ [40]
    Compute $\nabla \mathcal{L}_{k+1}(\mathbf{y}_{k+1}, \mu_{k+1})$ [40]
    Compute $\nabla \mathcal{L}_{R_k}(\mathbf{y}_k, \mu_{k+1})$ [40]
    Compute $\nabla \mathcal{L}_{R_{k+1}}(\mathbf{y}_{k+1}, \mu_{k+1})$ [40]
    if $\nabla \mathcal{L}_{k+1}(\mathbf{y}_{k+1}, \mu_{k+1}) < \varepsilon_{GL}$ [40] **then**
        return $\mathbf{x}_{k+1}$
    else if $||\mathbf{d}_x|| < \varepsilon_{dx}$ [1] **then**
        return $\mathbf{x}_{k+1}$
    else
        Perform a modified damped BFGS update on matrix $\mathbf{B}$ (Equation 17) [40]
        $k \leftarrow k + 1$
    end if
end for
return x_k, maximum iteration count reached

One important note is that unlike LSQP, this SLCP algorithm cannot utilize existing SQP solvers due to the fundamentally different nature of the sub-problem construction. In the work presented here, Algorithm 1 is implemented in a custom python suite, but the sub-problems are solved using the appropriate CVXOPT (Andersen et al. 2013) solver.

5 A simple test case

Before launching into complex design examples, it is valuable to gain intuition through the use of a simple example. Consider the following geometric program:



(a) As written (Equation 19) (b) Under log transformation (Equation 20)

Fig. 1 Visualizing the simple example problem in both the untransformed (**a**) and log transformed (**b**) spaces

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \;& \frac{1}{x^{0.1}} + 15x^{0.01} + \frac{1}{y^{0.1}} + 15y^{0.01} \\ \text{subject to} \;& 0.01x^{-1.1} + x^{0.1} + y \leq 1 \\ & x \geq \epsilon \\ & y \geq \epsilon \end{aligned} \tag{19}$$

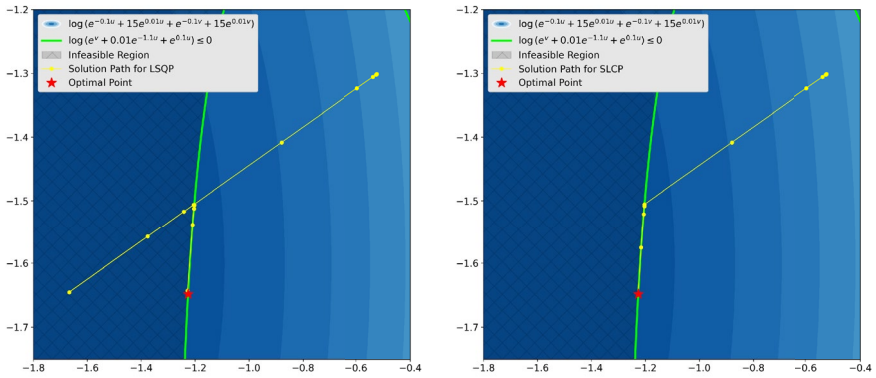
where ϵ is some small positive value, in this case 10^{-9}. Under the log transformation utilized by GP, LSQP, and SLCP, this problem becomes:

$$\begin{aligned} \underset{u,v}{\text{minimize}} \;& \log\left(e^{-0.1u} + 15e^{0.01u} + e^{-0.1v} + 15e^{0.01v}\right) \\ \text{subject to} \;& \log\left(e^{v} + 0.01e^{-1.1u} + e^{0.1u}\right) \leq 0 \end{aligned} \tag{20}$$

The problem can be directly visualized, both in the original space and in the log transformed space, as seen in Fig. 1.

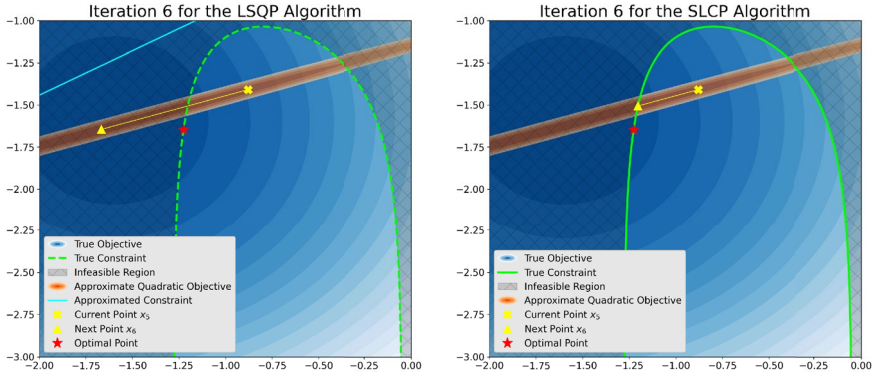
Figure 1 highlights the clear advantage of the log transformation. Figure 1a is characterized by long, skinny, irregular objective contours and a non-convex constraint, both of which make it poorly conditioned for solution with gradient based methods. In contrast, Fig. 1b has objective contours that are nearly circular in shape, and a constraint that carves out a convex set[4] with no cusps or drastic changes of curvature, making it highly conducive to gradient based optimization. While it is true that not all optimization formulations will benefit from this transformation [see discussion of the Rosenbrock problem in Karcher (Karcher 2021)], the abundance of literature showing the applicability of geometric programming to engineering design problems, along with the success of the LSQP algorithm, indicates this transformation is a useful tool in many cases of interest.

[4] The reader should not be confused by the use of the term 'convex set' here because the constraint function is in fact a concave function. A convex optimization problem is by definition the minimization of a convex objective function over a convex set generated by the constraint set, which is the case here.



(a) With LSQP (b) With SLCP

Fig. 2 Plotting the solution paths taken when solving the simple example problem with the LSQP (**a**) and SLCP (**b**) algorithms



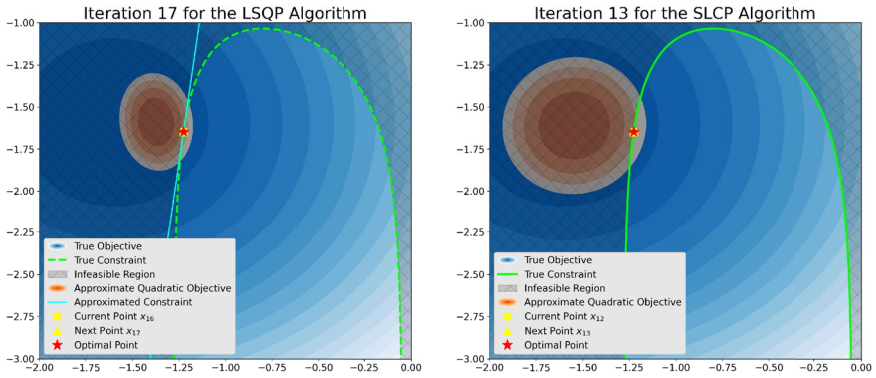
(a) The 6th iteration of LSQP (b) The 6th iteration of SLCP

Fig. 3 Comparing the critical 6th iteration of the LSQP and SLCP algorithms when solving the simple example problem

Now consider the difference between LSQP and SLCP. Starting from $(x_0, y_0) = (0.3, 0.05)$, Fig. 2 shows the two solution paths taken to reach the optimal solution (zoomed in to highlight the intermediate steps).

Clearly, the LSQP algorithm overshoots the constraint at some point during the solution, and must work back into the feasible region before finally reaching convergence (Fig. 2a). It is the 6th iteration of both algorithms that distinguishes the performance, shown in Fig. 3.

The linear approximation in Fig. 3a does not bound the QP sub-problem appropriately, resulting in an overshoot of the true constraint. In contrast, Fig. 3b shows



(a) The final iteration of LSQP (b) The final iteration of SLCP

Fig. 4 Comparing the second-order approximation of the objective function at the final iteration of the LSQP and SLCP algorithms

the enforced posynomial bounding the step as desired, preventing overshoot. In total, this enforcement saves the SLCP algorithm 4 iterations compared to LSQP.

The affect of the Reduced Lagrangian (Eq. 15) can also be visualized at the termination step of both algorithms, seen in Fig. 4.

The approximated objective function in Fig. 4b is a superior approximation of the true underlying objective function due to the absence of the constraint curvature terms that must be present in LSQP.

From this simple example, it is possible to draw the two conclusions that will be seen in the more extensive trials that follow. First, SLCP will have its widest performance gap with LSQP when more posynomial constraints are present in the formulation. Perhaps phrased more accurately, SLCP will outperform LSQP whenever it arises during the solution process that the linear approximation of a posynomial substantially deviates from the posynomial itself along the search direction. Second, SLCP will outperform LSQP by a wider margin when the initial guess is farther from the true optimal solution, as it is expected that the deviation between the posynomials and their linear approximations will grow wider over larger distances. These two trends will both be observed in the results below.

6 Evaluating algorithm performance

6.1 Methodology

To systematically test the effectiveness of the proposed SLCP method, three engineering design test problems were selected from the literature Floudas (Floudas et al. 2013), Kirschen-Ozturk (Kirschen et al. 2018), and Hoburg (Hoburg and Abbeel 2014) and solved using one of three methods:

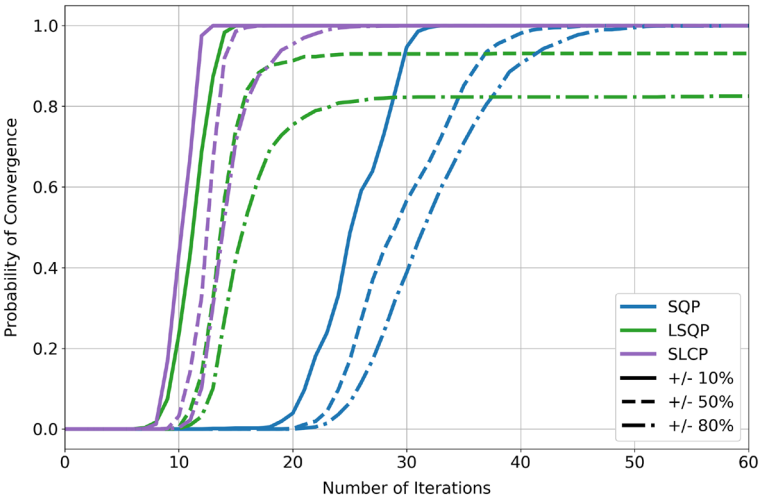


Fig. 5 Probability of convergence vs. iteration count for the floudas problem

1. A python implementation of SQP (Floudas and Kirschen-Ozturk only)
2. A python implementation of LSQP as described in Karcher (2021)
3. A python implementation of SLCP as described in Sect. 4

Note that three variations of the Hoburg problem were considered, as will be discussed below (see Sects. 6.5, 6.6, and 6.7). All three test problems (Floudas, Kirschen-Ozturk, and Hoburg) are relatively simple, but were selected to be representative of more complex cases like those published by Kirschen (Kirschen et al. 2016) and York (York et al. 2018).

This methodology is similar to the one used to demonstrate the effectiveness of the LSQP algorithm (Karcher 2021). This previous work (Karcher 2021) also validated the SQP implementation against the Matlab implementation of SQP and showed comparable performance.

For each of the 12 problem/algorithm combinations, 3000 trials were run starting from a random initial starting point. In 1000 of these cases, the initial guess was bounded to be within $\pm$ 10% of the known optimum, another 1000 were bound within $\pm$ 50% of the known optimum, and the final 1000 were bounded to be within $\pm$80% of the known optimum.

For each set of 1000 trials, a curve was constructed showing the fraction of cases that had converged within a certain number of iterations (Figs. 5, 6, 7, 8, and 9). In these plots the ideal algorithm would have a "Γ-like" shape, converging all cases in only one iteration, and so curves closest to the upper left of the graph represent the superior algorithms.

Due to the computational expense of 36000 trials, the computational resources of the MIT SuperCloud were utilized (Reuther et al. 2018), and a limit of 500 iterations was placed on all 3 algorithms.

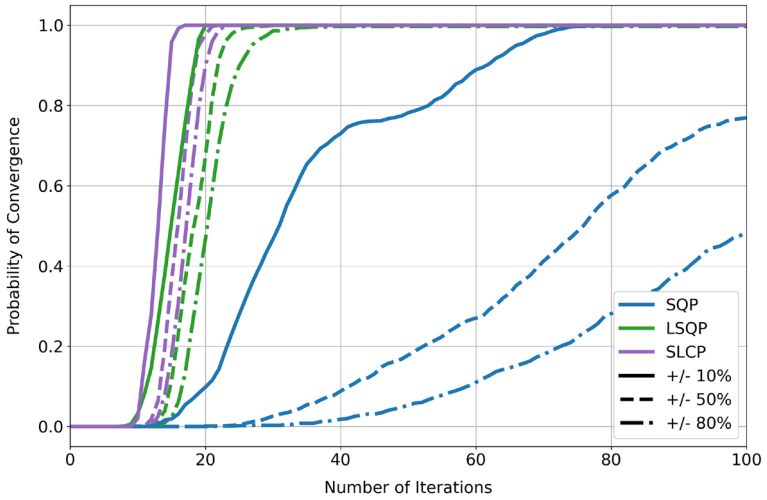


Fig. 6 Probability of convergence versus iteration count for the Kirschen-Ozturk Problem

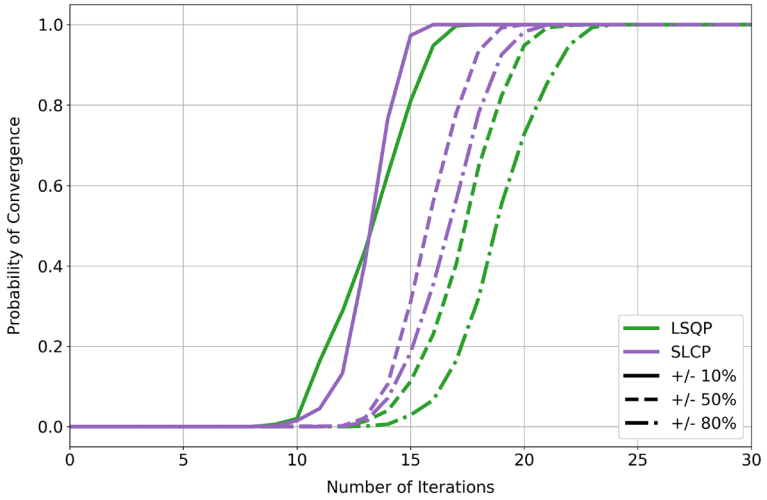


Fig. 7 Probability of convergence versus iteration count for the Hoburg problem with no black boxed constraints

6.2 Floudas problem

Floudas et al. (2013) offers the following optimization for the design of a heat exchanger:

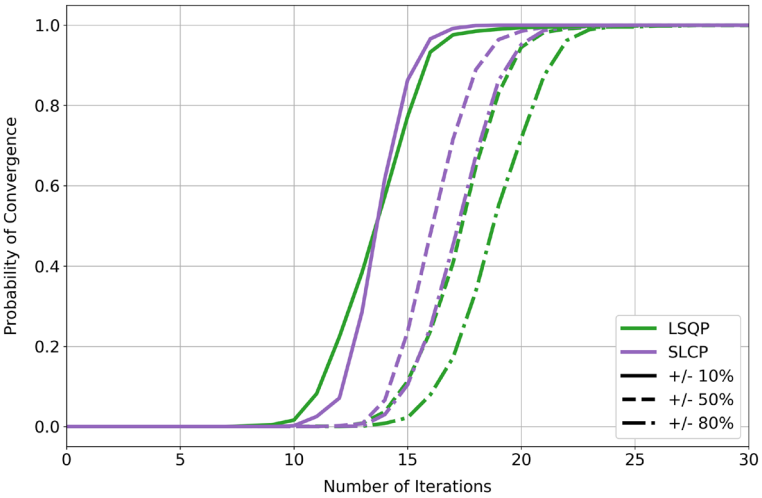


Fig. 8 Probability of convergence versus iteration count for the Hoburg problem with one black boxed constraint

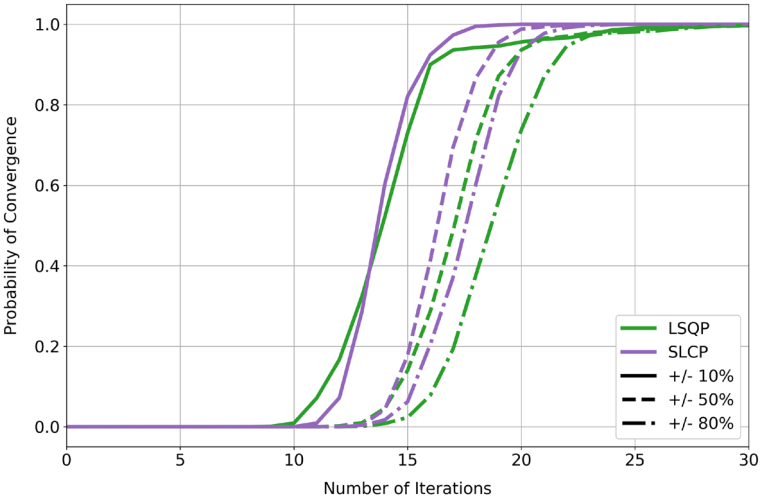


Fig. 9 Probability of convergence versus iteration count for the Hoburg problem with three black boxed constraints

$$
\begin{aligned}
\underset{x_1,\ldots,x_8}{\text{minimize}} \quad & x_1 + x_2 + x_3 \\
\text{subject to} \quad & \frac{833.33252x_4}{x_2x_6} + \frac{100}{x_6} - \frac{83333.333}{x_1x_6} \leq 1 \\
& \frac{1250x_5}{x_2x_7} + \frac{x_4}{x_7} - \frac{1250x_4}{x_2x_7} \leq 1 \\
& \frac{1250000}{x_3x_8} + \frac{x_5}{x_8} - \frac{2500x_5}{x_3x_8} \leq 1 \\
& 0.0025x_4 + 0.0025x_6 \leq 1 \\
& -0.0025x_4 + 0.0025x_5 + 0.0025x_7 \leq 1 \\
& -0.01x_5 + 0.01x_8 \leq 1
\end{aligned}
\tag{21}
$$

The problem has 8 variables and 6 constraints, none of which are mononomials and only one of which is a posynomial. The optimal solution is reported by Floudas et al. (2013) for comparison. Results for this test problem are presented in Fig. 5, and in Tables 2, 3, and 4.

Though this was one of the more interesting case studies for LSQP due to some unexpected tradeoffs with SQP (Karcher 2021), the results here paint a clear picture that SLCP strictly outperforms SQP and LSQP for this problem.

6.3 Kirschen-Ozturk problem

Kirschen et al. (2018) problem for aircraft sizing using low fidelity analysis models[5]:

$$
\begin{aligned}
\text{minimize} \quad & W_f \\
\text{subject to} \quad & W_f \geq c_T t D \\
& t \geq \frac{R}{V} \\
& D \geq \frac{1}{2}\rho V^2 S C_D \\
& C_D \geq \frac{A_{C_{D_0}}}{S} + kC_f\frac{S_{\text{wet}}}{S} + \frac{C_L^2}{\pi Ae} \\
& C_f \geq 0.074 Re^{-0.02} \\
& Re \leq \frac{\rho V\sqrt{S/A}}{\mu} \\
& \frac{1}{2}\rho V^2 S C_L \geq W_0 + W_w + \frac{1}{2}W_f \\
& \frac{1}{2}\rho V_{\min}^2 S C_{L_{\max}} \geq W \\
& W \geq W_0 + W_w + W_f \\
& W_w \geq W_{w_{\text{surf}}} + W_{w_{\text{strc}}} \\
& W_{w_{\text{surf}}} \geq C_{W_{w,1}} S \\
& W_{w_{\text{strc}}} \geq C_{W_{w,2}} \frac{N_{\text{ult}} A^{\frac{3}{2}} \sqrt{(W_0 + V_{f_{\text{fuse}}} g\rho_f) WS}}{\tau} \\
& V_f \leq V_{f_{\text{avail}}} \\
& V_f = \frac{W_f}{g\rho_f} \\
& V_{f_{\text{avail}}} \leq V_{f_{\text{wing}}} + V_{f_{\text{fuse}}} \\
& V_{f_{\text{wing}}}^2 \leq 0.0009\frac{S^3}{A}\tau^2 \\
& V_{f_{\text{fuse}}} \leq A_{C_{D_0}} 10[m]
\end{aligned}
\tag{22}
$$

[5] Attribution of this problem in the Kirschen paper is given to the uncredited Berk Ozturk (Kirschen et al. 2018), though much of the problem originates in the work of Hoburg and Abbeel (2014).

Variables were also constrained to be greater than a small positive constant in order to assist in the construction of sub-problems.

The problem is a signomial program, and is solved by Kirschen using the Difference of Convex Algorithm (DCA) (Kirschen et al. 2016; Burnell et al. 2020; Karcher 2021). That solution is used as the reference solution for this work, but it is important to note that solutions obtained using DCA do not hold any optimality guarantees as the convergence criteria for DCA is based on a relative change in the objective function and not on first order optimality conditions.

Results for this test problem are presented in Fig. 6, and in Tables 2, 3, and 4.

Again, the SLCP algorithm outperforms LSQP, both of which significantly outperform SQP.

6.4 Defining the hoburg problem

The final test problem comes from Hoburg and Abbeel (2014), and is rather extensive, consisting of 82 variables and 119 constraints. The problem defines the conceptual design and sizing of a UAV, which flies an outbound leg, a return leg, and has a separate set of sprint constraints that are used to size the powerplant. The problem seeks to minimize the objective:

$$W_{\text{fuel,out}} + W_{\text{fuel,ret}} \tag{23}$$

Subject to the following constraints, which are classified for readability.

Steady level flight relations:

$$\begin{aligned} W &= \frac{1}{2}\rho V^2 C_L S \\ T &\geq \frac{1}{2}\rho V^2 C_D S \\ Re &= \frac{\rho V S^{1/2}}{A^{1/2}\mu} \end{aligned} \tag{24}$$

Landing flight condition:

$$\begin{aligned} W_{\text{MTO}} &\leq \frac{1}{2}\rho_{\text{sl}} V_{\text{stall}}^2 C_{L,\max} S \\ V_{\text{stall}} &\leq 38 \end{aligned} \tag{25}$$

Sprint flight condition:

$$\begin{aligned} P_{\max} &\geq \frac{T_{\text{sprint}} V_{\text{sprint}}}{\eta_{0,\text{sprint}}} \\ V_{\text{sprint}} &\geq 150 \end{aligned} \tag{26}$$

Drag model:

$$
\begin{aligned}
C_D \geq & \frac{0.5}{S} + C_{D_p} + \frac{C_L^2}{\pi e A} \\
1 \geq & 2.56 \frac{C_L^{5.88}}{\tau^{3.32} Re^{1.54} C_{D_p}^{2.26}} + 3.80 \times 10^{-9} \frac{\tau^{6.23}}{C_L^{0.92} Re^{1.38} C_{D_p}^{9.57}} + \\
& 2.20 \times 10^{-3} \frac{\tau^{0.03} Re^{0.14}}{C_L^{0.01} C_{D_p}^{0.73}} + 1.19 \times 10^{4} \frac{C_L^{9.78} \tau^{1.76}}{Re^{1.00} C_{D_p}^{0.91}} + \\
& 6.14 \times 10^{-6} \frac{C_L^{6.53}}{\tau^{0.52} Re^{0.99} C_{D_p}^{5.19}}
\end{aligned}
\tag{27}
$$

Propulsive efficiency:

$$
\begin{aligned}
\eta_0 &\leq \eta_{\text{eng}} \eta_{\text{prop}} \\
\eta_{\text{prop}} &\leq \eta_i \eta_v \\
4\eta_i + \frac{T \eta_i^2}{\frac{1}{2} \rho V^2 A_{\text{prop}}} &\leq 4
\end{aligned}
\tag{28}
$$

Range constraints:

$$
\begin{aligned}
R &\geq 5000 \times 10^3 \\
z_{\text{bre}} &\geq \frac{gRT}{h_{\text{fuel}} \eta_0 W} \\
\frac{W_{\text{fuel}}}{W} &\geq z_{\text{bre}} + \frac{z_{\text{bre}}^2}{2} + \frac{z_{\text{bre}}^3}{6} + \frac{z_{\text{bre}}^4}{24}
\end{aligned}
\tag{29}
$$

Weight relations:

$$
\begin{aligned}
W_{\text{pay}} &\geq 500g \\
\tilde{W} &\geq W_{\text{fixed}} + W_{\text{pay}} + W_{\text{eng}} \\
W_{\text{zfw}} &\geq \tilde{W} + W_{\text{wing}} \\
W_{\text{eng}} &\geq 0.0372 P_{\text{max}}^{0.803} \\
\frac{W_{\text{wing}}}{f_{\text{wadd}}} &\geq W_{\text{web}} + W_{\text{cap}} \\
W_{\text{out}} &\geq W_{\text{zfw}} + W_{\text{fuel,ret}} \\
W_{\text{MTO}} &\geq W_{\text{out}} + W_{\text{fuel,out}} \\
W_{\text{sprint}} &= W_{\text{out}}
\end{aligned}
\tag{30}
$$

Wing structural model:

$$
\begin{aligned}
2q &\geq 1 + p \\
p &\geq 1.9 \\
\tau &\leq 0.15 \\
\bar{M}_r &\geq \frac{\tilde{W} A p}{24} \\
0.92\bar{w}\tau\bar{t}_{\text{cap}}^2 + \bar{I}_{\text{cap}} &\leq \frac{0.92^2}{2}\bar{w}\tau^2\bar{t}_{\text{cap}} \\
8 &\geq \frac{N_{\text{lift}}\bar{M}_r A q^2 \tau}{S\bar{I}_{\text{cap}}\sigma_{\max}} \\
12 &\geq \frac{A\tilde{W}N_{\text{lift}}q^2}{\tau S\bar{t}_{\text{web}}\sigma_{\max,\text{shear}}} \\
\nu^{3.94} &\geq 0.86p^{-2.38} + 0.14p^{0.56} \\
W_{\text{cap}} &\geq \frac{8\rho_{\text{cap}} g\tilde{w}\bar{t}_{\text{cap}} S^{3/2}\nu}{3A^{1/2}} \\
W_{\text{web}} &\geq \frac{8\rho_{\text{web}} g r_h \tau\bar{t}_{\text{web}} S^{3/2}\nu}{3A^{1/2}}
\end{aligned}
\tag{31}
$$

Additional information is available in the Hoburg paper Hoburg and Abbeel (2014). This formulation is GP compatible, and therefore has a known global optimum.

Three versions of this problem were considered in an effort to demonstrate the ability of SLCP to systematically evolve model fidelity. Note that many constraints in the Hoburg formulation are exact, and introduce no uncertainty to the final result (see Eqs. 24, 25, 26 in particular, though many of the other constraints are also exact). However, consider the constraint in Eq. 27:

$$
1 \geq 2.56\frac{C_L^{5.88}}{\tau^{3.32}Re^{1.54}C_{D_p}^{2.26}} + \cdots + 6.14\times 10^{-6}\frac{C_L^{6.53}}{\tau^{0.52}Re^{0.99}C_{D_p}^{5.19}} \tag{32}
$$

which is a posynomial fit to a set of XFOIL (Drela 1989) data for the NACA 24xx family of airfoils acting as a surrogate model for profile drag coefficient, C_{D_p}. This constraint is enforced for the outbound, return, and sprint segments, and so the form shown in Eq. 32 actually represents three separate constraints.

This model introduces uncertainty in two forms. First is the uncertainty of the fitted model itself. For points that were in the original fitting set, the model will not capture the exact C_{D_p} reported by XFOIL because the model fitting process is minimizing some RMS error. For points not in the original fitting set, interpolation uncertainty is also introduced (ie, features smaller than the sampling interval have been ignored). This model uncertainty can only be removed by tying XFOIL directly into the optimization problem, which though not possible in Hoburg's original GP formulation can be done with SLCP. Second is the epistemic uncertainty between the physics modeled in XFOIL and the true underlying physics. This epistemic uncertainty can only be reduced by using a higher fidelity analysis tool, and so the

ability to swap out an integrated XFOIL model with one of higher fidelity would be desirable. The following case studies will establish a path towards achieving this result.

6.5 Hoburg problem as formulated

First consider solving the Hoburg problem exactly as formulated. Figure 7 reports the results of the trial runs, with the averages reported in Tables 2, 3, and 4. Note that the SQP algorithm was not able to solve this problem as formulated from any initial guess, and so no results are reported for this or any of the subsequent cases.

There is little difference between the two algorithms when the initial guess is in a region close to the true optimum, since the initialization of the curvature matrix (in this case the identity matrix, see Algorithm 1) dominates the overall performance. But significant gains are seen as the initial guess decreased in quality, topping out at an average of 11.4% improvement, or about two iterations. The significant improvement is due to the posynomial constraints, particularly those characterized by Eq. 32, being exactly represented in the SLCP sub-problem. The next two cases serve to isolate the amount of computational savings that comes from directly implementing the posynomials of Eq. 32, while also evolving the problem towards something that can be used to integrate higher fidelity models.

6.6 Hoburg problem with one black boxed constraint

Consider that Eq. 32 is essentially a representation of the black boxed analysis function:

$$C_{D_p} = f(C_L, \tau, Re) \tag{33}$$

imposed in constraint form as:

$$1 \geq \frac{f(C_L, \tau, Re)}{C_{D_p}} \tag{34}$$

Many analysis models can be represented by Eq. 33, including low fidelity methods like XFOIL, high fidelity CFD methods, and even data from wind tunnel tests. So if the constraint posed in Eq. 34 can be included in the Hoburg formulation, then any analysis model which captures $f(C_L, \tau, Re)$ can be similarly used.

The goal of this test case is to introduce a single black boxed analysis model for $f(C_L, \tau, Re)$ while minimizing the impact on the rest of the GP compatible problem. Thus a single constraint, in this case the sprint segment instance of Eq. 32, was replaced by Eq. 34, where $f(C_L, \tau, Re)$ was determined by implicitly solving Eq. 32. In effect this keeps the function $f(C_L, \tau, Re)$ the same, but hides the true posynomial from the SLCP algorithm and should therefore reduce computational efficiency when compared to the previous case while maintaining the same solution. Results are reported in Fig. 8 and Tables 2, 3, and 4.

As expected, the loss of a posynomial constraint reduces the efficiency of SLCP and moves these curves in Fig. 8 closer to the LSQP curves, however SLCP still demonstrates a performance gain between 6–8% as the initial guesses get worse.

6.7 Hoburg problem with three black boxed constraints

For the final test problem, all three constraints represented by Eq. 32 (outbound, return, and sprint) were replaced with black boxed models for $f(C_L, \tau, Re)$. As with the previous case, the black boxes were implicit implementations of Eq. 32, leaving the problem identical to the original Hoburg problem but with these three posynomials hidden to the SLCP algorithm. The results are reported in Fig. 9 and in Tables 2, 3, and 4

As expected, the elimination of two more posynomial constraints further shrinks the gap between the SLCP and LSQP curves in Fig. 9. But critically, the fact that all three constraints have been black boxed means that the posynomial surrogate from Eq. 32 has been entirely eliminated, and a higher fidelity analysis model can simply be swapped in as a new black box that represents $f(C_L, \tau, Re)$.

To demonstrate this procedure, XFOIL was directly tied in to the optimization formulation, with gradients being computed using a finite difference scheme. Note that unlike the previous three problem formulations, the inclusion of XFOIL directly has fundamentally altered the structure of the problem and will affect the result (see Table 1 for a summary of the changes to the optimal solution). The GP optimal solution was used as the initial guess, and SLCP reached the optimal soluiton in 32 iterations.

Since XFOIL has been entirely contained in a black box of $f(C_L, \tau, Re)$, higher fidelity tools like MSES (Drela 2007), 2D Euler methods, or 2D Navier–Stokes simulations can readily integrated in exactly the same fashion. Furthermore, the design variables available in these higher fidelity analysis models can be optimized as a part of the SLCP run. For example, a NACA 4-series airfoil could be parameterized as $f(C_L, \tau, Re, c_{max}, l_{c_{max}})$ where c_{max} is the maximum camber in fraction of chord and $l_{c_{max}}$ is the location of that maximum camber, again as a fraction of chord. In this way, geometry fidelity can also evolve with the evolutions in analysis fidelity, all while anchored in a strong initial guess migrated from the previous fidelity level.

Together, Geometric Programming and Sequential Log-Convex Programming offer a potential approach to engineering design when multiple levels of analysis fidelity must be considered:

1. Develop low-fidelity conceptual models in a GP compatible form
2. Solve the GP to obtain a low fidelity candidate design to be used as an initial guess for higher levels of fidelity
3. Evolve analysis models to higher fidelity as desired and implement them in the optimization formulation as black box functions
4. Add any new design variables made available by the higer fidelity analysis models to the optimization formulation

Table 1 Comparison of variables with a greater than 1% variation when XFOIL is tied directly into the optimization formulation

Design variable	GP Optimal value	SLCP+XFOIL optimal value	Percent change
$C_{D_p,\text{sprint}}$	0.005732	0.005455	−4.82
$C_{D,\text{sprint}}$	0.007760	0.007477	−3.65
$P_{\max}$ [kW]	1186.1	1143.2	−3.61
T_{sprint} [N]	2209.6	2133.9	−3.42
$C_{D_p,\text{out}}$	0.005455	0.005615	2.93
W_{eng} [N]	2805.8	2724.1	−2.91
$C_{D_p,\text{ret}}$	0.005470	0.005627	2.87
$C_{D,\text{ret}}$	0.012682	0.012894	1.67
$C_{D,\text{out}}$	0.012668	0.012876	1.64
$\bar{t}_{\text{cap}}$	0.004544	0.004611	1.47
$\bar{I}_{\text{cap}}$	2.021e−5	2.555e−5	1.37
W_{cap} [N]	4103.8	4158.4	1.33
W_{wing} [N]	8613.4	8723.7	1.28
$C_{Di,\text{sprint}}$	0.000232	0.000230	−1.12
$C_{Di,\text{ret}}$	0.005416	0.005475	1.08
$C_{L,\text{ret}}$	0.574	0.580	1.03
$W_{\text{fuel, out}}$ [N]	3082.6	3114.1	1.02

Table 2 Average over 1000 trials of the number of iterations required to obtain an optimal solution starting from a guess within ± 10% of the known optimum

	SQP	LSQP	SLCP
Floudas	25.87	11.70 (−54.77%)	10.74 (−58.48%)
Kirschen-Ozturk	33.93	15.36 (−54.73%)	13.34 (−60.68%)
Hoburg-0	–	13.70	13.65 (−0.32%)
Hoburg-1	–	14.06	14.17 (+0.78%)
Hoburg-3	–	14.65	14.32 (−2.27%)

Table 3 Average over 1000 trials of the number of iterations required to obtain an optimal solution starting from a guess within ± 50% of the known optimum

	SQP	LSQP	SLCP
Floudas	30.17	14.30 (−52.60%)	12.92 (−57.18%)
Kirschen-Ozturk	72.55	18.89 (−73.96%)	16.38 (−77.42%)
Hoburg-0	–	17.79	16.29 (−8.44%)
Hoburg-1	–	17.82	16.66 (−6.47%)
Hoburg-3	–	17.66	16.87 (−4.49%)

Table 4 Average over 1000 trials of the number of iterations required to obtain an optimal solution starting from a guess within ± 80% of the known optimum

	SQP	LSQP	SLCP
Floudas	32.79	16.29 (−50.32%)	14.92 (−54.50%)
Kirschen-Ozturk	91.18	21.06 (−76.90%)	17.66 (−80.63%)
Hoburg-0	–	19.33	17.13 (−11.40%)
Hoburg-1	–	19.30	17.70 (−8.30%)
Hoburg-3	–	19.33	18.02 (−6.77%)

5. Solve the new optimization problem with SLCP, using the GP solution as an initial guess

Future work will dive into this methodology in great depth, particularly in developing a systematic process for evolving analysis models and the integration of new design variables into the outer optimization loop.

6.8 Further evolution of design problems

As additional constraints were black boxed in the Hoburg problem, the computational benefit of the SLCP algorithm was reduced compared to LSQP. It is further expected that as design problems evolve beyond the early phases of development, analysis models of increasing complexity will have to be implemented as black boxed constraints like the XFOIL constraint described in the final test case. When known structure is removed from the problem with increased fidelity and black boxing, it is expected that the gap between SLCP and LSQP will continue to shrink until the SLCP algorithm becomes exactly the LSQP algorithm when all constraints are black boxed. Though LSQP does still show some benefit over traditional SQP, the increased performance benefits of SLCP show that there can be significant benefits to preserving known log-log convex structure whenever possible and practical.

7 Conclusions

The method of Sequential Log-Convex Programming proposed in this work brings together the efficiency of Geometric Programming and the flexibility of Sequential Quadratic Programming in a practical algorithm for engineering design. An engineer who begins the design process using simple GP compatible models can utilize SLCP to evolve analysis fidelity and reduce uncertainty, while an engineer who begins the design process with higher fidelity models can utilize SLCP to exploit underlying mathematical structure inherent in many design problems and reduce overall computational expense with little to no modification of existing design practices.

Acknowledgements This material is based on research sponsored by the U.S. Air Force under agreement number FA8650-20-2-2002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions

contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Air Force or the U.S. Government. The authors would like to thank Mark Drela, Marshall Galbraith, John Hansman, and John Dannenhoffer for their input into the technical matter, along with the EnCAPS Technical Monitor Ryan Durscher. The authors also acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing high performance computing, database, and consultation resources that have contributed to the research results reported within this paper.

Funding Open Access funding provided by the MIT Libraries.

References

Agrawal A, Diamond S, Boyd S (2019) Disciplined geometric programming. Optim Lett 13(5):961–976
Andersen MS, Dahl J, Vandenberghe L (2013) Cvxopt: a python package for convex optimization
Anitescu M (2002) A superlinearly convergent sequential quadratically constrained quadratic programming algorithm for degenerate nonlinear programming. SIAM J Optim 12(4):949–978
Boggs PT, Tolle JW (1996) Sequential quadratic programming. Acta Numer 4:1–51
Boyd S (2015) Sequential convex programming, Lecture notes of EE364b. Stanford University, Spring Quarter
Boyd S, Kim SJ, Vandenberghe L, Hassibi A (2007) A tutorial on geometric programming. Optim Eng 8(1):67–127
Boyd S, Vandenberghe L (2009) Convex optimization, 7th edn. Cambridge University Press, Cambridge
Boyd SP, Kim SJ, Patil DD, Horowitz MA (2005) Digital circuit optimization via geometric programming. Oper Res 53(6):899–932
Boyd SP, Lee TH et al (2001) Optimal design of a cmos op-amp via geometric programming. IEEE Trans Comput Aided Des Integr Circuits Syst 20(1):1–21
Brown A, Harris W (2018) A vehicle design and optimization model for on-demand aviation. In: 2018 AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference
Burnell E, Damen NB, Hoburg W (2020) Gpkit: a human-centered approach to convex optimization in engineering design. In: Proceedings of the 2020 CHI conference on human factors in computing systems, pp 1–13
Burton M, Hoburg W (2018) Solar and gas powered long-endurance unmanned aircraft sizing via geometric programming. J Aircr 55(1):212–225
Constrained nonlinear optimization algorithms. https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html (2020)
Chiang M (2005) Geometric programming for communication systems. Now Publishers Inc
Chiang M, Tan CW, Palomar DP, Oneill D, Julian D (2007) Power control by geometric programming. IEEE Trans Wireless Commun 6(7):2640–2651
Clasen RJ (1984) The solution of the chemical equilibrium programming problem with generalized benders decomposition. Oper Res 32(1):70–79
Drela M (1989) Xfoil: an analysis and design system for low Reynolds number airfoils. In: Low Reynolds number aerodynamics. Springer, pp 1–12
Drela M (2007) A user's guide to mses 3.05. Tech. rep., Massachusetts Institute of Technology
Duchi J, Boyd S, Mattingley J (2018) Sequential convex programming. Lecture notes of EE364b. Stanford University, Spring Quarter

Floudas CA, Pardalos PM, Adjiman C, Esposito WR, Gümüs ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (2013) Handbook of test problems in local and global optimization, vol 33. Springer Science & Business Media

Greenberg HJ (1995) Mathematical programming models for environmental quality control. Oper Res 43(4):578–622

Hall DK, Dowdle A, Gonzalez J, Trollinger L, Thalheimer W (2018)Assessment of a boundary layer ingesting turboelectric aircraft configuration using signomial programming. In: 2018 Aviation technology, integration, and operations conference, p 3973

Hoburg W, Abbeel P (2013) Fast wind turbine design via geometric programming. In: 54th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference

Hoburg W, Abbeel P (2014) Geometric programming for aircraft design optimization. AIAA J 52(11):2414–2426

Jabr RA (2005) Application of geometric programming to transformer design. IEEE Trans Magn 41(11):4261–4269

Jian J, Liu P, Yin J, Zhang C, Chao M (2021) A qcqp-based splitting sqp algorithm for two-block nonconvex constrained optimization problems with application. J Comput Appl Math 390:113368

Kandukuri S, Boyd S (2002) Optimal power control in interference-limited fading wireless channels with outage-probability specifications. IEEE Trans Wireless Commun 1(1):46–55

Karcher C (2021) Logspace sequential quadratic programming for design optimization. arXiv preprint arXiv:2105.14441

Kirschen PG, Burnell EE, Hoburg WW (2016) Signomial programming models for aircraft design. In: 54th AIAA aerospace sciences meeting

Kirschen PG, Hoburg WW (2018) The power of log transformation: a comparison of geometric and signomial programming with general nonlinear programming techniques for aircraft design optimization. In: 2018 AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference

Kirschen PG, York MA, Ozturk B, Hoburg WW (2018) Application of signomial programming to aircraft design. J Aircr 55(3):965–987

Kraft D (1988) A software package for sequential quadratic programming

Li X, Gopalakrishnan P, Xu Y, Pileggi T (2004) Robust analog/RF circuit design with projection-based posynomial modeling. In: IEEE/ACM international conference on computer aided design, 2004. ICCAD-2004, pp 855–862. IEEE

Lin B, Carpenter M, de Weck O (2020) Simultaneous vehicle and trajectory design using convex optimization. In: AIAA Scitech 2020 Forum, p 0160

Liu M, Jian J, Tang C (2020) A method combining norm-relaxed QCQP subproblems with active set identification for inequality constrained optimization. Optimization pp 1–31

Marin-Sanguino A, Voit EO, Gonzalez-Alcon C, Torres NV (2007) Optimization of biotechnological systems through geometric programming. Theoret Biol Med Modell 4(1):38

Martins J, Ning A (2021) Engineering design optimization

Martins JR, Lambe AB (2013) Multidisciplinary design optimization: a survey of architectures. AIAA J 51(9):2049–2075

Misra S, Fisher MW, Backhaus S, Bent R, Chertkov M, Pan F (2014) Optimal compression in natural gas networks: a geometric programming approach. IEEE Trans Control Netw Syst 2(1):47–56

Nocedal J, Wright S (2006) Numerical optimization. Springer Science & Business Media

Preciado VM, Zargham M, Enyioha C, Jadbabaie A, Pappas G (2014) Optimal resource allocation for network protection: a geometric programming approach. IEEE Trans Control Netw Syst 1(1):99–108

Reuther A, Kepner J, Byun C, Samsi S, Arcand W, Bestor D, Bergeron B, Gadepally V, Houle M, Hubbell M, et al (2018) Interactive supercomputing on 40,000 cores for machine learning and data analysis. In: 2018 IEEE high performance extreme computing conference (HPEC), pp.1–6. IEEE

Saab A, Burnell E, Hoburg WW (018) Robust designs via geometric programming

Sela Perelman L, Amin S (2015) Control of tree water networks: a geometric programming approach. Water Resour Res 51(10):8409–8430

Tang CM, Jian JB (2008) A sequential quadratically constrained quadratic programming method with an augmented lagrangian line search function. J Comput Appl Math 220(1–2):525–547

Torenbeek E (2013) Advanced aircraft design: conceptual design, analysis and optimization of subsonic civil airplanes, 2nd edn. John Wiley & Sons Ltd, Chichester

Vera J, González-Alcón C, Marín-Sanguino A, Torres N (2010) Optimization of biochemical systems through mathematical programming: methods and applications. Comput Oper Res 37(8):1427–1438

Xu Y, Pileggi LT, Boyd SP 2004) Oracle: optimization with recourse of analog circuits including layout extraction. In: Proceedings of the 41st annual design automation conference, pp 151–154
York MA, Hoburg WW, Drela M (2018) Turbofan engine sizing and tradeoff analysis via signomial programming. J Aircr 55(3):988–1003
York MA, Öztürk B, Burnell E, Hoburg WW (2018) Efficient aircraft multidisciplinary design optimization and sensitivity analysis via signomial programming. AIAA J 56(11):4546–4561