Exploration of Nonconvex Solution Spaces via Operator Splitting Sequential Convex Programming

Justin Ganiban

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2026

Committee:

Behçet Açıkmeşe

Mehran Mesbahi

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Exploration of Nonconvex Solution Spaces via Operator Splitting Sequential Convex Programming

Justin Ganiban

Chair of the Supervisory Committee:
Behçet Açıkmeşe
Aeronautics & Astronautics

Sequential convexification methods provide an efficient and reliable means of computing feasible trajectories in nonconvex solution spaces. However, a well-known limitation of these algorithms is that they are inherently local in nature, and typically converge to a solution in the neighborhood of their initial guess. This paper presents a sequential operator-splitting framework, based on the alternating direction method of multipliers (ADMM), aimed at promoting exploration within the sequential convex programming (SCP) framework. In particular, diverse initial solutions are modeled as agents within the consensus ADMM framework. Driving these agents toward consensus facilitates exploration of the nonconvex optimization landscape. Numerical simulations demonstrate that the proposed method consistently yields equivalent or lower-cost solutions compared to the standard SCP approach, with the same or fewer agents.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1
INTRODUCTION

Nonconvex optimization problems arise in a wide range of applications, including control, robotics, machine learning, economics, structures. While convex optimization methods offer strong theoretical guarantees and efficient numerical algorithms, many problems of practical interest are inherently nonconvex.

Sequential convex programming (SCP) is one approach to solving these problems, in which a nonconvex problem is solved through a sequence of convex subproblems obtained via local linearization. These methods are typically guaranteed to converge to a stationary point of the original problem. However, because they rely on local approximations, their performance is highly dependent on the initialization [2, 3]. As a result, SCP-based methods often converge to suboptimal solutions within the basin of attraction of the initial guess.

Several approaches have been proposed to address the challenge of local minima in nonconvex optimization. Sampling-based methods explore the solution space by generating candidate solutions from a distribution and incrementally refining them [4, 5, 6, 7]. While these methods can provide broad exploration, they often scale poorly with problem dimension and may lack guarantees of optimality [8].

Multi-start optimization provides a simple alternative by solving multiple optimization problems from different initializations [9, 10]. This approach promotes exploration across different regions of the solution space, but each instance evolves independently. Consequently, information about the structure of the solution landscape is not shared, often resulting in redundant convergence to the same local minima and increased computational cost [11].

Another class of methods introduces stochastic perturbations to escape local minima, such as injecting noise into gradient-based updates [12]. While such techniques can help traverse saddle points and explore the landscape, they may compromise feasibility and

introduce significant computational overhead due to the need for repeated sampling [12].

In convex settings, augmented Lagrangian methods provide a principled framework for handling constrained optimization problems. The alternating direction method of multipliers (ADMM), in particular, has gained widespread use in convex optimization due to its ability to decompose problems with separable structure into smaller subproblems [13]. This decomposition is formalized through the concept of *operator splitting*, which enables complex optimization problems to be solved via iterative coordination of simpler components.

ADMM has been extensively used to exploit structure in large-scale optimization problems, including distributed optimization and model predictive control [14, 15, 16]. In particular, ADMM was employed to temporally split a trajectory optimization problem so that agents could solve trajectory optimization problems with fewer discrete samples in parallel. A similar strategy was used to simultaneously compute optimal trajectories and time-varying linear feedback control policies in [17]. In these contexts, operator splitting is primarily used to improve computational efficiency by decomposing a large problem into smaller, parallelizable subproblems. However, its potential for promoting exploration in nonconvex optimization has received comparatively less attention.

This work leverages operator splitting to promote exploration within the sequential convex programming framework. Specifically, we construct a population of agents with diverse initializations, each solving a locally convexified subproblem. These agents are coupled through a consensus ADMM formulation, enabling information sharing across trajectories and promoting coordinated exploration of the nonconvex solution space.

By combining local refinement (via SCP) with global coordination (via ADMM), the proposed method enables agents to escape stationary points that would otherwise trap standard local methods. Unlike multi-start approaches, the agents do not evolve independently, but instead are guided toward consensus, allowing them to traverse between basins of attraction in a structured manner.

We demonstrate that this operator-splitting framework leads to improved exploration of nonconvex solution landscapes and enables convergence to solutions that are not accessible from any single initialization alone.

Notation. The following notation is used throughout this work. We denote the set of

functions that are $n-$times differentiable by $\mathcal{C}^n$. A vector of ones of length n is denoted $\mathbf{1}_n$, and a vector of zeros of length n is denoted $\mathbf{0}_n$. We use the shorthand $[K]$ to denote the set of integers $\{0, \ldots, K\}$. The sets of real numbers are denoted by $\mathbb{R}$ and positive reals by $\mathbb{R}_+$, and the sets of integers and positive integers are denoted $\mathbb{Z}$, and $\mathbb{Z}_+$, respectively.

Chapter 2
BACKGROUND

This section reviews the foundational concepts underlying the proposed Operator-Splitting Sequential Convex Programming (OS-SCP) framework. The motivating application for the proposed method is optimal control and trajectory optimization, so some background will pose these problems. However, the OS-SCP method can be used for general nonconvex optimization, which is demonstrated in 3.5.

2.1 Sequential Convex Programming

We begin the background section with sequential convex programming (SCP), a widely used approach for solving nonconvex optimal control problems, and then develop the alternating direction method of multipliers (ADMM) starting from dual ascent. This progression highlights the complementary roles of local convexification and distributed coordination, which together motivate the proposed method.

Consider a discrete-time optimal control problem of the form

$$\min_{z} \quad J_K(z_K) + \sum_{k=0}^{K-1} J_k(z_k) \tag{2.1}$$

$$\text{s.t.} \quad x_{k+1} = f_k(z_k), \quad k \in [K-1], \tag{2.2}$$

$$g(z_k) \leq 0, \quad k \in [K], \tag{2.3}$$

$$h(z_k) = 0, \quad k \in [K], \tag{2.4}$$

$$z_k \in \mathcal{Z}_k^c, \quad k \in [K], \tag{2.5}$$

where $J_k : \mathbb{R}^{n_z} \to \mathbb{R}$ is the running cost function, $J_K : \mathbb{R}^{n_z} \to \mathbb{R}$ is the terminal cost function, and $z_k = [x_k^\top \; u_k^\top]^\top$ denotes the stacked state-control vector. The cost functions J_K, J_k, dynamics f_k and constraints g, h may be nonlinear and nonconvex.

We apply the prox-linear method [18, 19, 20] to solve (2.1). For a general function

$\Xi : \mathbb{R}^{n_z} \to \mathbb{R}^{n_\Xi}$, we denote the linearized function

$$\tilde{\Xi}(\bar{z}, z) = \Xi(\bar{z}) + \nabla\Xi^\top(\bar{z})(z - \bar{z}),$$

where $\bar{z}$ is the state about which the function is linearized. We can formulate (2.1) as an unconstrained minimization problem by penalizing the nonconvex constraints [21]. The convex constraints are enforced through an indicator function. At the $(j+1)^{\text{th}}$ iteration, the nonconvex cost and constraints are linearized about the solution to the j^{th} subproblem. We define the linear function

$$\begin{aligned}\Theta(z^j, z) \coloneqq \tilde{J}_K(z_K^j, z_K) + \sum_{k=0}^{K-1} \tilde{J}_k(z_k^j, z_k) + \mathbb{1}_{\mathcal{Z}^c}(z) \\ + w^1 \sum_{k=0}^{K-1} \left\| \tilde{f}_k(z_k^j, z_k) \right\|_1 + w^2 \mathbf{1}_{n_g}^\top \left| \tilde{g}(z^j, z) \right| + w^3 \left\| \tilde{h}(z^j, z) \right\|_1, \quad (2.6)\end{aligned}$$

where $w^1, w^2, w^3 \in \mathbb{R}_+$ are user-selected weights, and $\mathcal{Z}^c = \bigcup_{k\in[K]} \mathcal{Z}_k^c$. Note that if the running cost and terminal cost functions are convex, they need not be linearized, and $\tilde{J}_K(z_K^j, z_K)$ and $\tilde{J}_k(z_K^j, z_K)$ in (2.6) are replaced by $J_K(z_K)$ and $J_k(z_K)$, respectively. Since the linearizations of the cost and constraints are only accurate in the neighborhood of the trajectory about which they are performed, deviation from that trajectory is penalized with a trust region, resulting in the function

$$\Gamma(z^j, z) \coloneqq \Theta(z^j, z) + \frac{w^{\text{p}}}{2} \sum_{k=0}^{K} \left\| z_k - z_k^j \right\|_2^2, \quad (2.7)$$

where $w^{\text{p}} \in \mathbb{R}_+$ is a user-selected proximal weight. The SCP framework solves (2.1) by successively solving the *convex subproblem*

$$z^{j+1} = \arg\min_z \Gamma(z^j, z). \quad (2.8)$$

Between SCP iterations, the trajectory about which the problem is linearized is updated with the solution of the previous iterate. This process is repeated until a convergence criterion is satisfied. Algorithm 1 outlines a multi-start version of the standard SCP algorithm, in which n_a initial trajectories are used to initialize the algorithm.

This procedure defines a fixed-point iteration in which each step refines the trajectory using a convex model constructed from local information. Under suitable regularity conditions, SCP converges to a stationary point of the original nonconvex problem.

Algorithm 1 Multi-Start Standard SCP

Require: $\epsilon_c \in \mathbb{R}_+,\ n^{\max} \in \mathbb{Z}_+,\ z_i^0$ for $i \in [n_a]$

for $i = 1, \ldots, n_a$ **do**

while $j \leq j^{\max},\ \left\|\Theta(z_i^{j-1}, z_i^j)\right\| > \epsilon_c$ **do**

$z_i^{j+1} \leftarrow \arg\min_z \Gamma(z_i^j, z)$

$j \leftarrow j + 1$

end while

end for

Limitations of SCP Despite its effectiveness, SCP is inherently local. Because each subproblem is built from a linearization about the current iterate, convergence is typically restricted to a basin of attraction determined by the initial guess. Consequently, different initializations may converge to distinct local minima, and SCP lacks a mechanism for systematically exploring the broader nonconvex solution landscape.

2.2 Dual Ascent and Augmented Lagrangian Methods

To develop ADMM, consider the equality-constrained optimization problem

$$\min_x\ f(x) \quad \text{s.t.} \quad Ax = b. \tag{2.9}$$

The associated Lagrangian is

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^\top (Ax - b), \tag{2.10}$$

where λ denotes the dual variable. Dual ascent methods iteratively minimize the Lagrangian with respect to x and update the dual variable using the constraint residual [13]:

$$x^{k+1} = \arg\min_x \mathcal{L}(x, \lambda^k), \tag{2.11}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} - b). \tag{2.12}$$

While conceptually simple, dual ascent can exhibit poor convergence due to ill-conditioning of the Lagrangian. The augmented Lagrangian method addresses this issue by introducing

a quadratic penalty on constraint violation:

$$\mathcal{L}_\rho(x, \lambda) = f(x) + \lambda^\top (Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2. \tag{2.13}$$

This additional curvature improves numerical stability and leads to the method of multipliers, which alternates between minimizing $\mathcal{L}_\rho$ and updating λ.

2.3 Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) extends the augmented Lagrangian framework by introducing variable splitting, enabling decomposition of complex problems into simpler subproblems [13]. Consider

$$\min_{x,z} \; f(x) + g(z) \quad \text{s.t.} \quad Ax + Bz = c. \tag{2.14}$$

ADMM proceeds by alternating updates over x and z:

$$x^{k+1} = \arg\min_x \mathcal{L}_\rho(x, z^k, \lambda^k), \tag{2.15}$$

$$z^{k+1} = \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, \lambda^k), \tag{2.16}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \tag{2.17}$$

By decoupling the optimization variables, ADMM enables efficient optimization of structured problems. The iterations enforce the KKT conditions by approximating the stationarity conditions for x and z iteratively while driving the primal feasibility residual

$$r^{k+1} = Ax^{k+1} + Bz^{k+1} - c$$

to zero through the dual update. The dual variable λ accumulates constraint violation, so persistent disagreement between the coupled variables is penalized more strongly in subsequent iterations. At convergence, the primal residual vanishes, the dual variable no longer changes, and the first-order optimality conditions of the x- and z-subproblems recover the stationarity conditions of the original constrained problem. Thus, ADMM can be interpreted as iteratively enforcing primal feasibility, dual feasibility, and stationarity, which together form the KKT conditions.

2.4 Consensus ADMM

A special case of ADMM that is particularly relevant to this work is the consensus formulation:

$$\min_{\{z_i\},\bar{z}} \sum_{i=1}^{n_a} p_i(z_i) + q(\bar{z}) \quad \text{s.t.} \quad z_i = \bar{z}, \ \ \forall i. \tag{2.18}$$

Here, each z_i corresponds to a local agent, while $\bar{z}$ represents a global consensus variable. The constraint on each z_i ensures that over iterations of the algorithm, all agents will converge to the same consensus state. Thus, a single optimum is achieved among all split operators. The ADMM iterations take the form

$$z_i^{k+1} = \arg\min_z \ p_i(z) + \frac{\rho}{2}\|z - \bar{z}^k + \xi_i^k\|_2^2 \tag{2.19a}$$

$$\bar{z}^{k+1} = \arg\min_{\bar{z}} \ q(\bar{z}) + \frac{\rho}{2}\sum_i \|z_i^{k+1} - \bar{z} + \xi_i^k\|_2^2 \tag{2.19b}$$

$$\xi_i^{k+1} = \xi_i^k + \left(z_i^{k+1} - \bar{z}^{k+1}\right) \tag{2.19c}$$

alternating between a primal update, consensus update, and dual update. This formulation coordinates multiple optimization processes through a shared consensus variable. Each agent solves a local problem, while the consensus step aggregates information across agents and enforces agreement. Similar to the standard ADMM algorithm, consensus ADMM enforces the KKT conditions iteratively.

2.5 Connection to Exploration of Nonconvex Solution Spaces

The preceding developments reveal a complementary relationship between SCP and ADMM. SCP provides a method for locally minimizing possibly nonconvex objectives subject to possibly nonconvex constraints through convex approximations, but does not facilitate exploration beyond the basin of attraction of a given initialization. In contrast, ADMM provides a framework for coordinating multiple optimization processes through consensus constraints, enabling information sharing across otherwise independent solves.

This observation motivates the operator-splitting SCP framework proposed in this work. As demonstrated in [22], this coupling promotes exploration of the nonconvex solution space

and enables convergence to solutions that are not accessible from any single initialization alone. This OS-SCP method is the focus of the work contained in this thesis.

Chapter 3

OPERATOR SPLITTING SEQUENTIAL CONVEX PROGRAMMING

The solution obtained using standard SCP corresponds to a stationary point of (2.1) in the neighborhood of the initial trajectory, z^0 [23, 18]. As a consequence, SCP exhibits a strong dependence on initialization, and may converge to suboptimal solutions when initialized in unfavorable regions of the solution space. This motivates the development of methods that enable more effective exploration of nonconvex landscapes.

In this section, we introduce a modified SCP framework, referred to as operator-splitting sequential convex programming (OS-SCP), which is designed to promote exploration while preserving the structure and feasibility properties of SCP.

3.1 Operator Splitting SCP (OS-SCP)

3.1.1 Primal update

We introduce n_a virtual agents, each initialized with a distinct initial guess, to explore different regions of the solution space. At each iteration, these agents independently solve convex subproblems derived from local linearizations, while being softly coupled through a shared consensus variable.

Specifically, the OS-SCP algorithm begins by solving (2.19a) with $p_i(z_i) = \Theta(z_i^j, z_i)$ for each agent. This corresponds to a standard SCP update with a soft trust region and an additional proximal consensus penalty term that encourages agreement between agents. The resulting subproblem is given by

$$z_i^{j+1} = \arg\min_{z} \ \Theta(z_i^j, z) + \frac{w_p}{2} \sum_{k=0}^{K} \left\| z_k - z_{i,k}^j \right\|_2^2 + \frac{\rho}{2} \left\| z - \bar{z}^j + \xi_i^j \right\|_2^2, \qquad i \in [n_a]. \tag{3.1}$$

Here, z_i denotes the state of the i^{th} agent, and z_i^j is the reference used to construct the linearized model at iteration j. ξ_i denotes the dual variable for the i^{th} agent at iteration

j. The parameter w_p is the trust region weight, and $\rho \in \mathbb{R}_+$ controls the strength of the coupling between agents.

3.1.2 Consensus update

After each agent computes an updated trajectory, a consensus step is performed to aggregate information across agents and enforce approximate feasibility with respect to the original problem.

We begin by defining the convexified constraint set

$$\mathcal{Z} = \mathcal{Z}^c \cap \mathcal{Z}^n, \tag{3.2}$$

where $\mathcal{Z}^c$ denotes the set of convex constraints, and $\mathcal{Z}^n$ is the set of convexified nonconvex constraints obtained by linearizing about the reference. The convexified constraint set is given by

$$\mathcal{Z}^n = \{z \mid \tilde{g}(\hat{z}_k^j, z_k) \leq 0,\ \tilde{h}(\hat{z}_k^j, z_k) = 0,\ k \in [K],\ z_{k+1} = \tilde{f}_k(\hat{z}_k^j, z_k),\ k \in [K-1]\}. \tag{3.3}$$

The consensus trajectory is then obtained by solving (2.19b) with $q(\bar{z})$ defined as the indicator function of $\mathcal{Z}$. This yields

$$\bar{z}^{j+1} = \arg\min_{\bar{z}} \mathbb{1}_{\mathcal{Z}}(\bar{z}) + \frac{\rho}{2} \sum_{i=1}^{n_a} \left\| z_i^{j+1} - \bar{z} + \xi_i^j \right\|_2^2, \tag{3.4}$$

$$\text{where } \mathbb{1}_{\mathcal{Z}}(\bar{z}) = \begin{cases} 0 & \text{if } \bar{z} \in \mathcal{Z} \\ +\infty & \text{if } \bar{z} \notin \mathcal{Z} \end{cases}.$$

This problem admits a closed-form solution given by the projection of the average agent trajectory onto the convexified constraint set:

$$\bar{z}^{j+1} = \frac{\rho}{2}\, \Pi_{\mathcal{Z}} \left(\frac{1}{n_a} \sum_{i=1}^{n_a} (z_i^{j+1} + \xi_i^j) \right), \tag{3.5}$$

where $\Pi_{\mathcal{Z}}(\cdot)$ denotes the Euclidean projection operator,

$$\Pi_{\mathcal{Z}}(v) := \arg\min_{\bar{z} \in \mathcal{Z}} \|\bar{z} - v\|_2^2 . \tag{3.6}$$

This projection step enforces approximate feasibility while combining information from multiple agents. As the agents are pulled towards the consensus through the proximal penalty at each update, the agents explore the solution space between the initializations.

3.1.3 Dual update

Finally, the dual variables are updated by solving (2.19c), which accumulates the disagreement between each agent and the consensus trajectory. These dual variables act as memory terms that guide future iterations toward agreement.

3.1.4 Primal and Dual Residuals

To monitor convergence of the OS-SCP iterations, we define primal and dual residuals analogous to those used in standard ADMM. The primal residual measures disagreement between each agent trajectory and the consensus trajectory:

$$r_i^{j+1} = z_i^{j+1} - \bar{z}^{j+1}, \qquad i \in [n_a]. \tag{3.7}$$

As the primal residual approaches zero, the agents converge toward consensus and the equality constraints $z_i = \bar{z}$ are satisfied.

The dual residual measures the change in the consensus trajectory between iterations:

$$s^{j+1} = \rho \left(\bar{z}^{j+1} - \bar{z}^{j} \right). \tag{3.8}$$

This residual reflects the degree to which the consensus variable has stabilized. Large dual residuals indicate that the consensus trajectory is still changing significantly between iterations, while small dual residuals indicate approximate stationarity of the consensus update.

The OS-SCP iterations are considered converged when both the primal and dual residual norms fall below prescribed tolerances:

$$\left\| r^j \right\|_2 \leq \varepsilon_r, \qquad \left\| s^j \right\|_2 \leq \varepsilon_s. \tag{3.9}$$

At convergence, the agents approximately agree on a common trajectory and the consensus updates have stabilized, corresponding to approximate satisfaction of the KKT conditions for the consensus optimization problem.

3.1.5 OS-SCP Algorithm

The complete algorithm is summarized in Algorithm 2. The agents are initialized with distinct trajectories, and at each iteration, the primal, consensus, and dual updates are

performed sequentially. This process is repeated until both primal and dual residuals fall below specified tolerances.

Algorithm 2 Operator-Splitting SCP

Require: $\epsilon_r,\ \epsilon_s,\ \epsilon_c \in \mathbb{R}_+,\ j^{\max} \in \mathbb{Z}_+,\ z_i^0$ for $i \in [n_a]$

while $\left(\left\|\delta_r^j\right\| > \epsilon_r,\ \delta_s^j > \epsilon_s,\ j \leq j^{\max},\ \left\|\Theta(\bar{z}^{j-1}, \bar{z}^j)\right\| > \epsilon_c\right)$ **do**
  for $i = 1, \ldots, n_a$ **do**
    $z_i^{j+1} \leftarrow \arg\min_z \Gamma^c(z_i^j, \bar{z}_i^j, \xi_i^j, z)$
  end for
  $\bar{z}^{j+1} \leftarrow \frac{\rho}{2}\ \Pi_{\mathcal{Z}}\left(\frac{1}{n_a}\sum_{i=1}^{n_a}\left(z_i^{j+1} + \xi_i^j\right)\right)$
  for $i = 1, \ldots, n_a$ **do**
    $\xi_i^{j+1} \leftarrow \xi_i^j + \left(z_i^{j+1} - \bar{z}^{j+1}\right)$
  end for
  $j \leftarrow j + 1$
end while

While the formulation above enforces consensus across all components of z, this is not required in general. In practice, consensus may be restricted to a subset of the variables or constraints, which can be advantageous when only certain components of the state space exhibit significant nonconvexity.

3.2 Convergence Considerations

The convergence properties of the proposed OS-SCP framework are influenced by both its constituent components: sequential convex programming (SCP) and the alternating direction method of multipliers (ADMM).

For convex problems, ADMM is known to converge under mild assumptions, with both primal and dual residuals vanishing as the number of iterations increases. However, these guarantees do not generally extend to nonconvex settings. While ADMM has been observed to converge to stationary points for certain classes of nonconvex problems, such behavior typically requires additional assumptions such as smoothness, boundedness, or regularity conditions.

Similarly, SCP methods are known to converge to a stationary point of the original nonconvex problem under appropriate conditions on the linearization accuracy and trust-region mechanism. However, this convergence is inherently local, and the final solution depends strongly on the initialization.

The OS-SCP algorithm combines these two methodologies by embedding SCP subproblems within an ADMM consensus framework. As a result, the overall algorithm does not admit general convergence guarantees to a global or even local optimum of the original nonconvex problem. In particular, the presence of successive linearizations implies that the optimization landscape changes at each iteration, while the consensus coupling introduces additional interactions between agents.

Despite the lack of formal guarantees, several properties can be observed in practice. First, the primal and dual residuals associated with the ADMM iterations typically decrease over time, indicating convergence to a consensus among agents. Second, the use of local convex approximations ensures that each subproblem remains well-posed and computationally tractable. Although formal convergence guarantees remain an open question, the numerical examples in this work suggest that OS-SCP retains the practical convergence properties of standard SCP

From an operator-splitting perspective, the OS-SCP method can be viewed as an iterative fixed-point scheme applied to a sequence of locally convexified problems. While this perspective provides intuition for the observed convergence behavior, it does not yield formal guarantees due to the nonconvex and time-varying nature of the underlying operators.

In summary, the proposed method should be viewed as a heuristic optimization framework that leverages structure from both SCP and ADMM. Analytic convergence properties is an avenue for future work.

3.3 Practical Enhancements

The OS-SCP algorithm as described in 3.1 generally performs well in low dimension nonconvex optimization problems. For higher dimensional systems or more difficult to deal with constraints, there are some practical enhancements that can be made to the algorithm which improve performance.

3.3.1 Effect of Relaxation on Consensus and Exploration

Relaxation is a commonly used modification in ADMM and operator-splitting methods [13], in which the consensus update is performed using a weighted combination of current and previous iterates. In convex settings, over-relaxation is often used to accelerate convergence. However, in the OS-SCP framework, relaxation plays a more nuanced role due to the nonconvex and multi-agent nature of the algorithm.

We introduce a relaxation parameter $\alpha \in (0, 2)$ in the consensus update. In particular, the relaxed consensus variable is defined as

$$\tilde{z}^{j+1} = \alpha \bar{z}^{j+1} + (1 - \alpha)\bar{z}^{j}.$$

When $\alpha = 1$, the standard OS-SCP update is recovered. Values $\alpha > 1$ correspond to over-relaxation, while $\alpha < 1$ correspond to under-relaxation.

In convex ADMM, literature notes that over-relaxation can cause accelerated convergence [13]. It does so by more aggressively aggregating agents toward the consensus. However, this contradicts the goal of OS-SCP as an exploration focused method. If agents are drawn to the consensus too quickly, they might jump over local minima which are more optimal than the local minima nearest to the current consensus. Thus, in OS-SCP we implement under-relaxation by choosing $\alpha \in [0.2, 0.8]$. This stabilizes iterations, but more importantly dampens the convergence of the agents towards the consensus. This promotes exploration of agents while they are more slowly pulled towards congruence. The downside to this approach is that it increases number of iterations to converge in some cases.

3.3.2 Weighted Consensus

In the standard consensus ADMM formulation, the consensus update is based on an unweighted average of the agent trajectories. This treats all agents equally, regardless of the quality of their current solutions. In the context of OS-SCP, however, different agents may converge to trajectories with significantly different objective values. This suggests that some agents provide more informative guidance than others, especially as agents are exploring the solution space while being iteratively pulled towards the consensus.

To account for this, we introduce a weighted consensus mechanism in which the consensus update is biased toward higher-quality agents. Let $w_i^j \geq 0$ denote the weight associated with agent i at iteration j, with $\sum_{i=1}^{n_a} w_i^j = 1$. The weighted consensus update is given by

$$\bar{z}^{j+1} = \frac{\rho}{2}\, \Pi_{\mathcal{Z}} \left(\frac{\sum_{i=1}^{n_a} w_i^j \big(z_i^{j+1} + \xi_i^j\big)}{\sum_{i=1}^{n_a} w_i^j} \right). \tag{3.10}$$

The weights w_i^j are chosen to favor agents with lower objective values. In practice, this can be achieved by defining weights based on the relative cost of each agent or by prioritizing feasible agents, for example. This biases the consensus trajectory toward regions of the solution space associated with user priority. The consensus update then promotes promising directions while exploring.

While this departs from the classical ADMM formulation, it remains consistent with the broader operator-splitting framework, in which different proximal or projection operators may be used to encode problem-specific structure.

Empirically, we observe that weighted consensus can significantly improve convergence to lower-cost solutions in highly nonconvex problems. In particular, it helps mitigate the tendency of the algorithm to converge to suboptimal, average-based consensus trajectories when a subset of agents have already identified a better region of the solution space.

However, this modification also introduces additional considerations. If the weights are too heavily concentrated on a small subset of agents, the algorithm may lose its exploratory behavior and prematurely collapse to a single agent. In practice, this results in some manual tuning of hyperparameters to balance these effects.

3.3.3 Clique Consensus

In some cases, problems are structured such that natural groupings of initializations might be intuitive. In these cases, it might be advantageous to explore this structure without pulling agents out of their groups to a single consensus.

To address this limitation, we introduce a clustered consensus strategy in which agents are partitioned into subsets, or cliques, and consensus is enforced locally within each subset rather than globally across all agents.

Let $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$ denote a partition of the agent set $[n_a]$, where each $\mathcal{C}_m \subseteq [n_a]$ represents a clique of agents. For each clique $\mathcal{C}_m$, we define a local consensus variable $\bar{z}_{\mathcal{C}_m}$, and the consensus update is performed independently within each clique. Specifically, the consensus update for clique $\mathcal{C}_m$ is given by

$$\bar{z}_{\mathcal{C}_m}^{j+1} = \frac{\rho}{2}\, \Pi_{\mathcal{Z}} \left(\frac{1}{|\mathcal{C}_m|} \sum_{i \in \mathcal{C}_m} \left(z_i^{j+1} + \xi_i^j \right) \right). \tag{3.11}$$

Each agent $i \in \mathcal{C}_m$ is then coupled only to the consensus variable associated with its clique, rather than to a global consensus across all agents. Clique consensus allows multiple candidate solutions to evolve in parallel, with each clique exploring a different region of the solution space.

The clustering structure may be fixed or adaptive. In the simplest case, agents are partitioned into clusters at initialization and remain in those clusters throughout the optimization. Alternatively, clusters may be updated dynamically based on similarity metrics, allowing agents to reorganize as the optimization progresses.

Clique consensus can also be interpreted within the operator-splitting framework as modifying the communication graph between agents. Rather than a fully connected consensus network, the algorithm operates over a set of smaller, localized networks, each performing its own consensus update. This perspective highlights the connection between clique consensus and distributed optimization over graphs.

In practice, clique consensus is particularly effective in structured nonconvex problems where the solution space contains potentially known, multiple well-separated basins of attraction. It can be useful when some intuition on the solution space exists, such as mutlimodal design spaces, problems with discrete transitions, and cases in which premature global agreement is harmful to exploration. However, this method introduces additional design choices, including the number of cliques and the assignment of agents to cliques.

3.3.4 Discussion

Relaxation combined with weighted consensus provides a framework for controlling both the rate and direction of consensus formation, enabling a balance between exploration of the

solution space and exploitation of high-quality trajectories. We find that these modifications improve the algorithm's performance in practice when applied to many problems. Clique consensus has a specific use case, but when used effectively can exploit known structures of problems.

3.4 A Discussion on Effective Agent Initialization

For effective exploration, it is desirable for the initial agents to span distinct regions of the solution space such that, as the consensus mechanism gradually couples the agents, the intermediate iterates traverse multiple local basins of attraction. In this sense, effective initializations are often located near the "edges" or "corners" of the feasible solution space, where the resulting trajectories or designs are qualitatively different from one another.

Identifying these initializations typically requires problem-specific intuition regarding the underlying dynamics, constraints, or structural characteristics of the optimization landscape. In many cases, intentionally poor or highly unconventional initial guesses can be beneficial if they encourage exploration of regions that would not normally be reached by more refined low-cost initializations. While such agents may not individually converge to desirable solutions, their interaction with the consensus mechanism can guide the population toward intermediate local minima of lower cost.

At the current stage of development, effective agent initialization remains highly problem-dependent. Consequently, each numerical example presented in this work includes a discussion of the initialization strategy used to generate diverse agent populations and encourage exploration of distinct solution families.

3.5 Numerical Results - Simple Trajectory Optimization

We now demonstrate the performance of the exploration-focused operator-splitting SCP method against standard SCP in two illustrative examples: a simple obstacle avoidance example, and an obstacle avoidance example over a landscape with a non-uniform cost field. All numerical examples are solved using the QOCO solver within CVXPY [24].

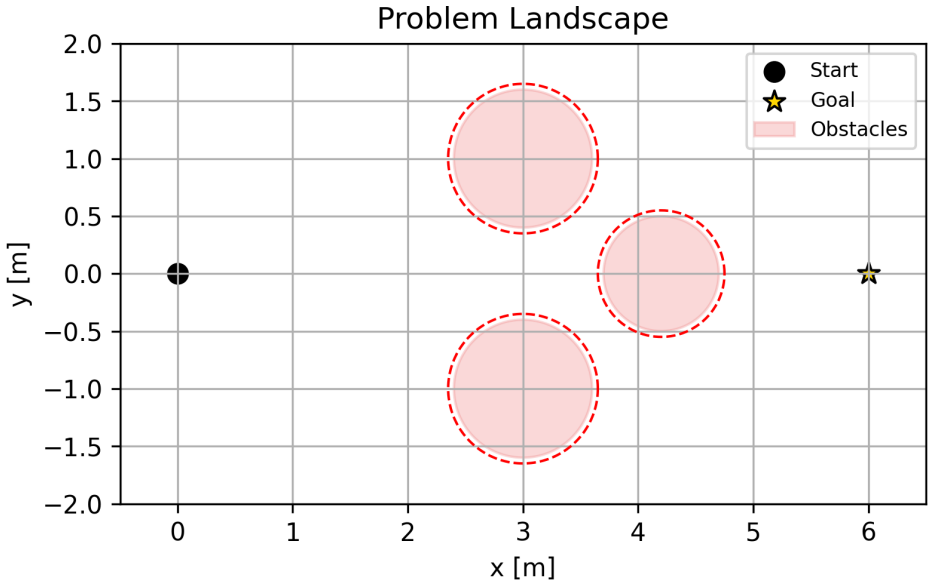


Figure 3.1: Problem setup for simple obstacle avoidance example.

3.5.1 Unicycle Trajectory Optimization

Consider the discrete-time kinematic model for a unicycle with constant velocity

$$x_{k+1} = f_k(t_k, x_k, u_k) = \begin{bmatrix} r_k^x + v\cos\theta_k \Delta t \\ r_k^y + v\sin\theta_k \Delta t \\ \theta_k + u_k \Delta t \end{bmatrix}, \tag{3.12}$$

where the state is $x_k = [r_k^x \quad r_k^y \quad \theta_k]^\top \in \mathbb{R}^3$ and control $u_k \in \mathbb{R}$ is the yaw rate. The speed is defined as a constant $v > 0$, and the dynamics are discretized with the uniform time step Δt. We aim to find an optimal trajectory and nominal control to bring the vehicle from an initial state to a desired terminal state, while avoiding obstacles. The problem setup is shown in Figure 3.1.

Let x_g be the desired terminal state. We express the problem in the form of (2.1), where the convex terminal and running cost functions are

$$J_K(z_K) = (e^x z_K - x_g)^\top Q_g (e^x z_K - x_g), \tag{3.13a}$$

$$J_k(z_k) = \|e^u z_k\|_2^2, \tag{3.13b}$$

with $Q_g \succeq 0$. The nonconvex inequality constraints are

$$g_\iota(z_k) = R_\iota - \|e_r z_k - c_\iota\| \leq 0,\ \iota \in \{1,2,3\}, \tag{3.14}$$

where $R_\iota \in \mathbb{R}_+$ is the radius of obstacle ι and $c_\iota \in \mathbb{R}^2$ is the center of obstacle ι.

Following the prox-linear methodology, at iteration j, we formulate the nonconvex optimization problem as a convex unconstrained minimization problem by penalizing the constraints, and linearizing the cost and constraints about the solution to the $(j-1)^{\text{th}}$ subproblem. We formulate the penalized cost as in (2.7) with linearized cost from (2.6). Note that since the terminal and running costs are convex, they need not be convexified, and $\tilde{J}_K(z_K^j, z_K)$ and $\tilde{J}_k(z_k^j, z_k)$ in (2.6) are replaced by $J_K(z_K)$ and $J_k(z_k)$, respectively.

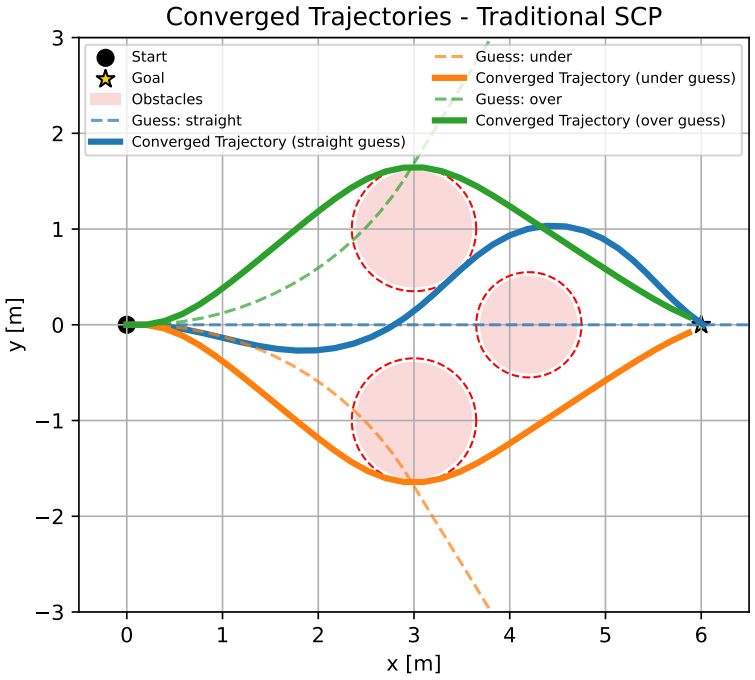
Generating Initial Guesses

For this problem, it is visually intuitive to choose good guesses and guesses which cover edges of the solution space. Because the unicycle is constant velocity and the obstacles are equidistant, minimum time and control solutions should pass through the center of the first two obstacles and then either above or below the third central obstacle. To cover edges of the solution space, OS-SCP should initialize agents whose nearby local minima are trajectories which go over or under all obstacles. Again, visually it is intuitive that these are suboptimal guesses, but this is done to demonstrate the agent initialization process for problems which do not have visually intuitive solutions.

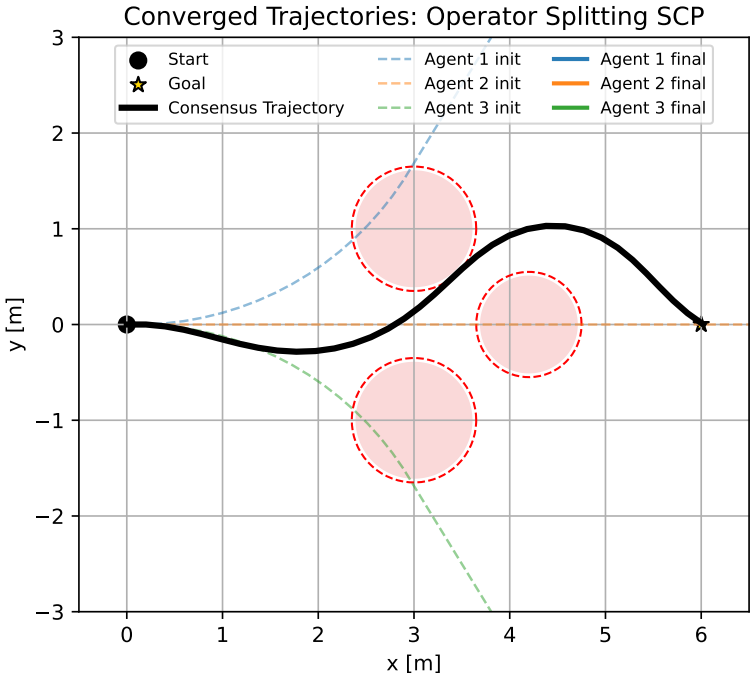
SCP solution

We first solve this problem with the standard SCP method according to Algorithm 1 with three initial guesses. The first guess is an arc veering toward the left of the vehicle, the second is a straight line from initial state to the goal, and third is an arc veering toward the right of the vehicle. The initial guesses and the converged trajectories are shown in Figure 3.2a, and the corresponding final costs and total iterations for each solve are shown in Table 3.1. From Figure 3.2a, it is evident that the SCP solution for each initialization converges to a solution in the neighborhood of the corresponding initial guess. This demonstrates the algorithm's difficulty in escaping the stationary point near its initialization. Additionally,

Table 3.1 shows that only the solution with a straight initial guess converges to a minimum cost solution. This dependence on a good initial guess is a known limitation of SCP. In this visually intuitive example, a straight line guess might be an obvious choice. However, generating good (and especially dynamically feasible) initial guesses is generally not trivial. From this example, the need for an exploration-focused, SCP-based algorithm is evident.



(a) Standard SCP algorithm.



(b) OS-SCP algorithm.

Figure 3.2: Initial (dotted) and converged (solid) trajectories.

Table 3.1: SCP Results

Guess Direction	Cost	Iterations
Over	0.230	27
Straight	0.155	36
Under	0.230	27

OS-SCP solution

We now solve the problem using the proposed OS-SCP method, summarized in Algorithm 2. At each iteration, each agent solves a convex subproblem. A consensus update is then performed, and finally a dual update is performed. This process repeats until the primal and dual residuals fall below specified tolerances $\epsilon_r \in \mathbb{R}_+$ and $\epsilon_s \in \mathbb{R}_+$, respectively. The same three initial guesses as before are used to initialize the three agents, where agent 1 is assigned the "upper" guess, agent 2 the "straight" guess, and agent 3 the "lower" guess. The converged solution is shown in Figure 3.2b, and the evolution of the primal and dual residuals are shown in Figure 3.3. The residuals converge to zero over iterations, and OS-SCP successfully forms a consensus between all three agents. The consensus converges to the same trajectory as the lowest cost solution of the standard SCP example, demonstrating the ability for the OS-SCP method to pull agents out of the local minima near their initialization. We compare the performance of both methods in Table 3.2, where OS-SCP finds an equal cost trajectory in fewer iterations than standard SCP. Since at each OS-SCP iteration three convex subproblems are solved (one per agent), and three standard SCP problems are solved (one per initial guess), the run times of the methods are roughly equal.

3.5.2 Trajectory Optimization with Gaussian Terrain Fields

The second problem extends the first example. Consider the same kinematic model and problem setup. However, we add spacial preference biases via a Gaussian terrain field. The

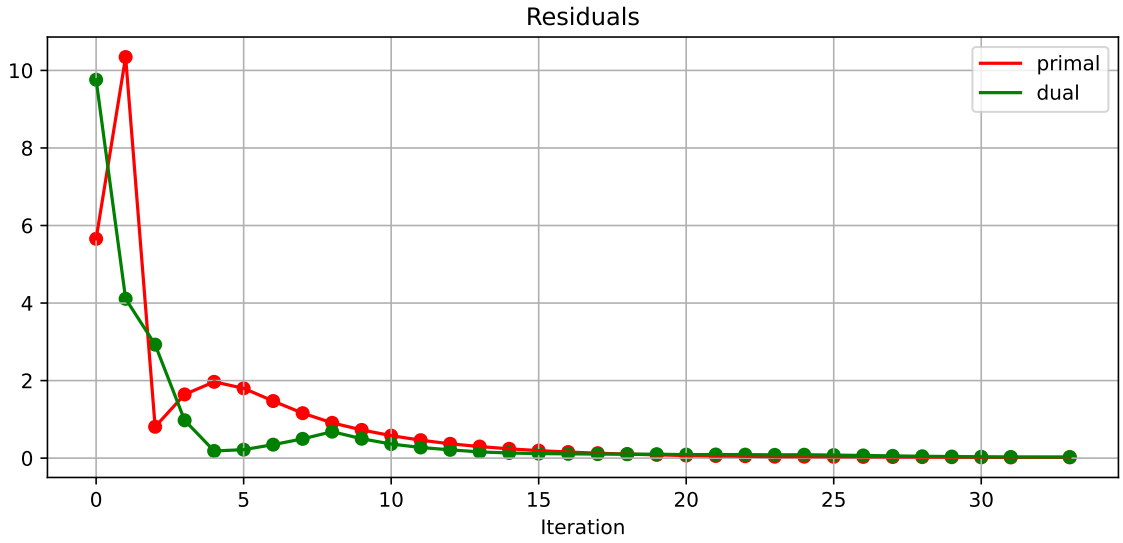


Figure 3.3: Primal and dual residuals over iterations for OS-SCP.

Table 3.2: Numerical Comparison - Unicycle Trajectory

Method	Cost	Iterations
Standard SCP	0.155	36
OS-SCP	0.155	33

cost map created by the terrain field is expressed as

$$J_{\text{map}}(r) \;=\; \sum_{\ell=1}^{L} \exp\Big(-\tfrac{1}{2}(r-\mu_\ell)^\top \Sigma_\ell^{-1}(r-\mu_\ell)\Big), \tag{3.15}$$

where $\mu_\ell \in \mathbb{R}^2$ is the position of the center of the ℓ-th Gaussian field, and $\Sigma_\ell \succeq 0$ controls the shape of the field and its amplitude. At the j^{th} iteration, the cost map in (3.15) is linearized about the solution to the $(j-1)^{\text{th}}$ subproblem, z^j, and added to the linearized cost function in (3.13). Then, SCP solves the same unconstrained convex subproblem as in (2.8), but with updated $\Gamma(z^j, z)$ to include the Gaussian terrain field cost.

The new problem setup is shown in Figure 3.4, where Gaussian cost fields are added between the upper and lower corridors of the obstacles. The terrain incentivizes traveling through the lower corridor by giving it negative cost, and penalizes the upper corridor with a higher cost terrain.

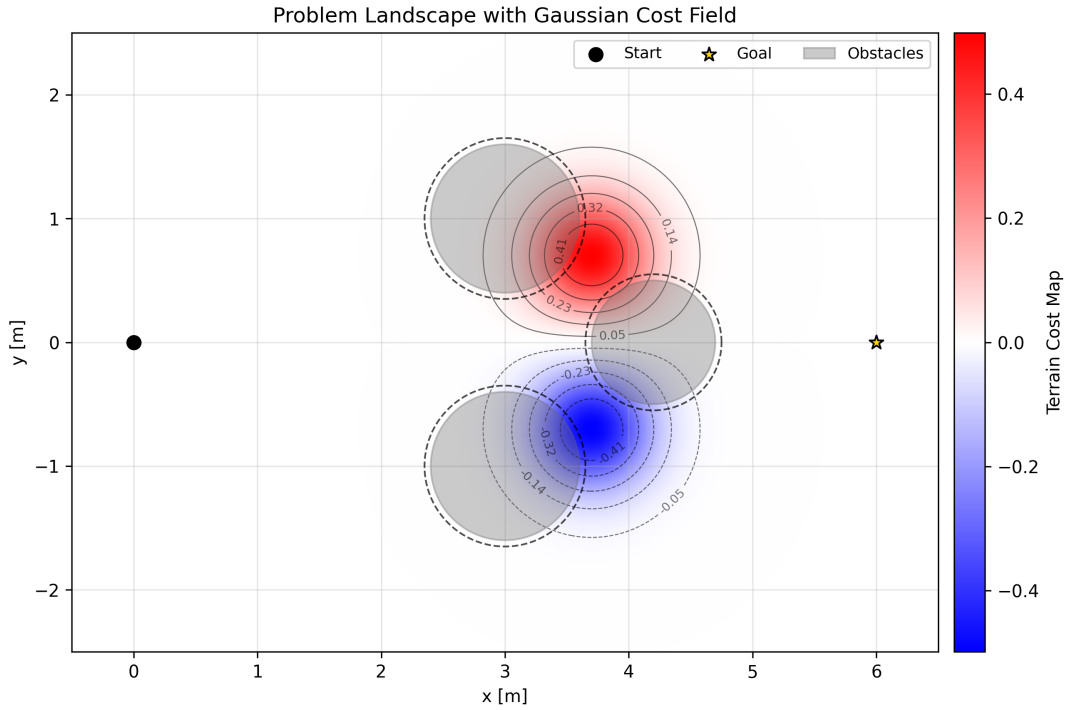


Figure 3.4: Problem setup for trajectory optimization problem with Gaussian terrain field.

SCP Solution

As with the previous example, we first solve the problem with the standard SCP method outlined in Algorithm 1. We initialize three independent standard SCP solves with the same three initial guesses as from the first example. Figure 3.5a shows that each initialization results in a converged trajectory in the neighborhood of the initial guess. Notably, the "straight" guess gets trapped in a local minimum in the upper corridor. In order for this method to find a minimum-cost solution, we need to add a new initial guess close to the global optimum which passes directly through the lower corridor, depicted in red in Figure 3.5b. The only initialization to converge to the lowest cost trajectory is the one whose initial guess passes through the optimal region. This further demonstrates the sensitivity of the standard SCP algorithm to the initial guess. Again, for this visually intuitive example choosing an initial guess near the global optimum is possible. However, for systems in higher dimensions with more complex solution spaces, an initial guess close to the global optimum is generally difficult to generate.

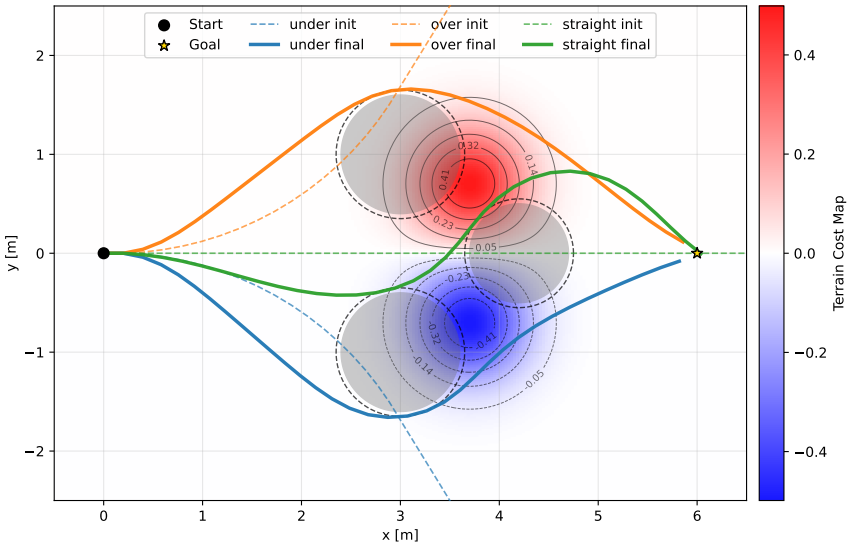
Table 3.3: Numerical Results - Unicycle Trajectory with Gaussian Terrain Field

Method	Cost	Iterations
Standard SCP (3 guesses)	-0.136	28
Standard SCP (4 guesses)	-0.715	31
OS-SCP	-0.715	33

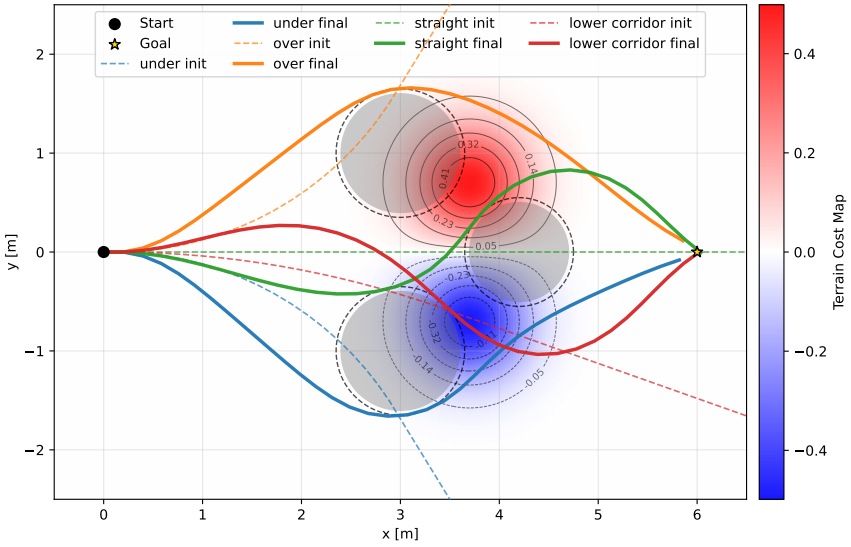
OS-SCP Solution

We then solve the problem with OS-SCP, where we use 3 agents with the same initial guesses as in the original example: "straight", "lower", and "upper". The agents explore the solution space before forming a consensus through the lower corridor, successfully finding the lowest cost solution, as shown in Figure 3.5c, without the need for an initial guess that passes through the lower corridor. Table 3.3 compares the OS-SCP method against the lowest cost

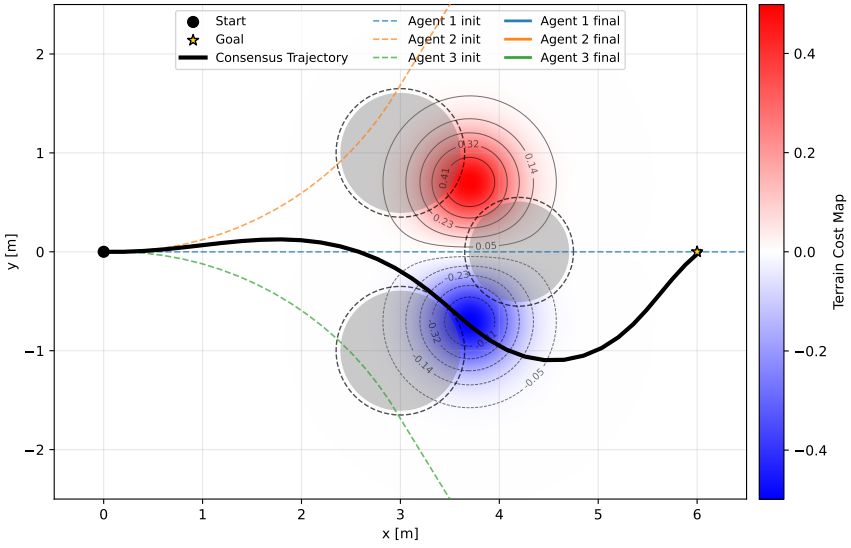
solution from standard SCP. The OS-SCP consensus converges to the same trajectory as the lowest cost solution from the standard SCP solves in near the same number of iterations, but does not have the same need for an initial guess near the global optimum. This example demonstrates the exploratory nature of the OS-SCP method and its lack of dependency on an accurate initial guess. Compared to the standard SCP approach, OS-SCP also provides the benefit of removing the human-in-the-loop requirement of selecting the best trajectory from a set of converged trajectories in the case of multiple unique trajectories with equivalent costs.



(a) Standard SCP algorithm.



(b) Standard SCP algorithm with additional "lower corridor" guess.



(c) OS-SCP algorithm.

Chapter 4

TRAJECTORY OPTIMIZATION OF A MULTI-PASS AEROBRAKING CAMPAIGN VIA OS-SCP

This chapter presents a challenging nonlinear trajectory optimization problem used to evaluate the proposed OS-SCP framework. Specifically, we consider the optimization of a multi-phase aerobraking campaign for a high-energy Earth return trajectory.

As interest grows in sending larger robotic and crewed vehicles deeper into space, innovative guidance, navigation, and control (GN&C) methods are required to overcome the severe propulsive costs associated with planetary capture and landing. Traditional fully propulsive capture strategies impose substantial fuel mass penalties, limiting payload capability and mission flexibility. In contrast, aerobraking and atmospheric capture can dramatically reduce the required ΔV, enabling higher-energy missions and increased payload capacity.

However, optimizing an aerobrake trajectory of this class is extremely challenging due to the highly nonlinear and sensitive nature of atmospheric flight dynamics. The problem contains strong nonlinearities arising from coupled orbital and aerodynamic dynamics, atmospheric uncertainty, heating and dynamic pressure constraints, and discontinuous changes in trajectory behavior across atmospheric passes and in-space orbits [25], [26], [27]. These effects produce a highly nonconvex optimization landscape containing numerous local minima.

Additionally, the sensitivity of the dynamics requires relatively tight trust regions within SCP, which often restricts the optimizer to remain close to the initial guess. As a result, standard SCP frequently converges to locally optimal trajectories strongly determined by initialization, with limited exploration of the broader solution space.

These characteristics make the aerobraking problem a particularly suitable application for the OS-SCP framework. By evolving multiple coupled trajectory hypotheses simultaneously, OS-SCP is able to explore multiple regions of the solution space while still leveraging

the computational efficiency and constraint-handling capabilities of SCP.

4.1 Problem Definition

We consider the problem of optimizing a multi-pass aerobraking trajectory for a vehicle transitioning from an initial high-apogee elliptical orbit to a desired low Earth orbit (LEO). The objective is to determine a control profile that minimizes total ΔV while satisfying nonlinear vehicle dynamics together with aerodynamic, thermal, and control constraints.

4.1.1 Dynamics Model

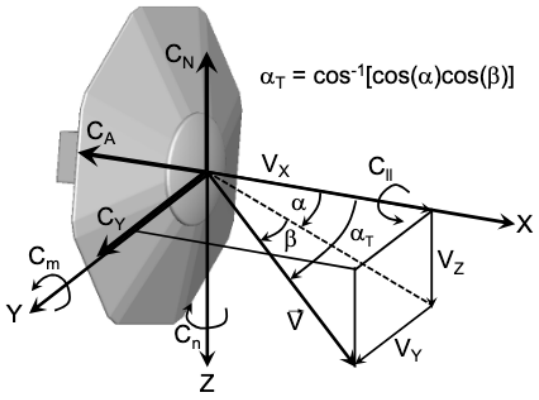


Figure 4.1: NASA ADEPT SR-1 low-lift aerodynamic decelerator

The vehicle considered in this example is a low-lift aerodynamic decelerator, similar to the NASA ADEPT vehicle [28] depicted in Figure 4.1. The vehicle dynamics are modeled using a three-degree-of-freedom (3-DoF) Cartesian formulation. The state vector is defined as

$$x = \begin{bmatrix} r_x & r_y & r_z & v_x & v_y & v_z & m \end{bmatrix}^T,$$

where $r \in \mathbb{R}^3$ is inertial position, $v \in \mathbb{R}^3$ is velocity, and m is vehicle mass.

The control input is given by

$$u = \begin{bmatrix} \Delta v_x & \Delta v_y & \Delta v_z & C_{N,n} & C_{N,b} \end{bmatrix}^T,$$

where $C_{N,n}$ and $C_{N,b}$ parameterize the commanded aerodynamic lift direction in the local normal-binormal frame.

The continuous-time dynamics are

$$\dot{r} = v, \tag{4.1}$$

$$\dot{v} = g(r) + \frac{1}{m}(L + D + T), \tag{4.2}$$

$$\dot{m} = 0, \tag{4.3}$$

where

$$g(r) = -\mu \frac{r}{|r|^3}$$

is the gravitational acceleration, and L, D, and T denote lift, drag, and thrust forces, respectively.

Aerodynamic forces are modeled as

$$q = \frac{1}{2}\rho|v|^2, \tag{4.4}$$

$$L = q\left(C_{N,n}\hat{n} + C_{N,b}\hat{b}\right), \tag{4.5}$$

$$D = -qC_A\hat{t}, \tag{4.6}$$

where ρ is atmospheric density, $\hat{t}$ is the velocity direction, and $\hat{n}$ and $\hat{b}$ define the local lift frame. Aerodynamic coefficients are obtained from lookup tables as functions of Mach number and lift coefficient.

4.1.2 Path Constraints

The trajectory must satisfy continuous-time path constraints on heating rate, dynamic pressure, and aerodynamic loading. These constraints are given by

$$g(x) = \begin{bmatrix} k_Q\rho(r)|v|^3 - \dot{Q} * \max \\ q - q * \max \end{bmatrix} \leq 0. \tag{4.7}$$

These constraints are enforced through CT-CS SCP [23]. The final trajectory must additionally satisfy terminal orbit constraints corresponding to the desired target orbit.

4.2 Sequential Convex Programming Formulation

The continuous-time optimal control problem is transcribed into a finite-dimensional optimization problem using successive convex programming (SCP).

4.2.1 Trajectory Discretization

The trajectory is discretized over N nodes. Let (x_k, u_k) denote the state and control at node k. Nodes are distributed strategically to improve numerical conditioning while preserving sufficient resolution during atmospheric flight.

Nodes are first placed at key dynamical events along the reference trajectory, including the apogee of each aerobraking pass and the atmospheric entry and exit points. Additional nodes are then linearly distributed between atmospheric entry and exit during each pass to improve aerodynamic control resolution.

This discretization concentrates nodes within atmospheric flight segments while preserving dedicated nodes for impulsive ΔV maneuvers at apogee. Through experimentation, this strategy was found to significantly improve SCP stability, conditioning, and convergence speed.

The control input is parameterized using first-order hold (FOH), producing a continuous piecewise-linear control profile between nodes. The dynamics are then exactly discretized over each trajectory segment:

$$x_{k+1} = F(x_k, u_k, u_{k+1}) := x_k + \int_0^1 f\left(x, (1-\tau)u_k + \tau u_{k+1}\right) d\tau. \tag{4.8}$$

Exact discretization preserves continuous-time dynamic fidelity between nodes, which is particularly important due to the strong sensitivity of atmospheric flight dynamics.

4.2.2 Linearization and Convexification

At SCP iteration j, the nonlinear dynamics are linearized about the reference trajectory (x_k^j, u_k^j):

$$x_{k+1} \approx F(x_k^j, u_k^j) + A_k(x_k - x_k^j) + B_k(u_k - u_k^j), \tag{4.9}$$

where A_k and B_k are Jacobians of the discretized dynamics.

The convex SCP subproblem is then formulated as

$$\min_{x^-,x^+,u,u^{\text{imp}}} \quad \Gamma + \frac{1}{2}p^T W p + \lambda^T p + \frac{1}{2s_x}||\delta x^+||^2 + \frac{1}{2s_u}||\delta u||^2 \tag{4.10a}$$

$$\text{s.t.} \quad x_k^+ = x_k^- + B_k^{\text{imp}} u_k^{\text{imp}}, \qquad k = 0, \ldots, N \tag{4.10b}$$

$$x_{k+1}^- = F(x_k^+, u_k) + A_k \delta x_k^+ + B_k \delta u_k + C_k \delta u_{k+1}, \quad k = 0, \ldots, N-1 \tag{4.10c}$$

$$z_k \geq 0, \qquad k = 0, \ldots, N \tag{4.10d}$$

$$||\Delta V_k|| \leq \Gamma_k, \qquad k = 0, \ldots, N \tag{4.10e}$$

$$||\ell_k|| \leq C_{L,\max}, \qquad k = 0, \ldots, N \tag{4.10f}$$

$$h(x_N^+) + \nabla h(x_N^+)^T \delta x_N^+ = p \tag{4.10g}$$

Here, p is a virtual buffer applied only to the terminal orbit constraint, while the quadratic proximal penalties enforce trust-region behavior to maintain validity of the local linearization.

The terminal orbit constraint function $h(x_N)$ enforces agreement with the target orbital angular momentum and eccentricity vectors:

$$h(x) = \left[r \times v - h_{\text{target}} \; v \times (r \times v) - \frac{r}{|r|} - e_{\text{target}} \right]. \tag{4.11}$$

Unlike many SCP formulations, no virtual control is introduced into the dynamics. Due to the sensitivity of atmospheric trajectories, maintaining first-order dynamic feasibility throughout optimization was found to significantly improve robustness.

4.2.3 Auto-Tuned Feasibility Management

The terminal buffer p is penalized using the Auto-SCvx primal-dual penalty update method [29]:

$$\lambda \leftarrow \lambda + s_\lambda p^*, \tag{4.12}$$

$$W \leftarrow \frac{((W) \odot |p^*|)}{\varepsilon}, \tag{4.13}$$

where ε is the target feasibility tolerance.

However, aggressive feasibility enforcement can cause the quadratic penalty matrix W to grow excessively before the optimizer is able to sufficiently reduce the constraint violation. To mitigate this instability, we introduce a ramped feasibility tolerance $\varepsilon_j = \max(\gamma^j \varepsilon_0, \varepsilon)$ which begins with a loose tolerance ε_0 and gradually tightens toward the desired feasibility tolerance ε throughout successive convexification iterations.

This ramping strategy allows the optimizer to progressively reduce terminal constraint violation while avoiding instability caused by excessively rapid penalty growth.

4.3 OS-SCP For Aerobraking

To improve exploration of the nonconvex solution space, the SCP formulation is extended using the OS-SCP framework.

4.3.1 OS-SCP Implementation

Multiple agents are initialized with distinct trajectory guesses. Generating initial guesses is a consideration on its own which will be addressed in 4.3.2. At each iteration, each agent solves an SCP subproblem of the form

$$z_i^{j+1} = \arg\min_z \Gamma^c(z_i^j, \bar{z}^j, \xi_i^j, z), \tag{4.14}$$

where Γ^c includes both trust-region penalties and consensus coupling.

A consensus trajectory is computed via weighted projection:

$$\bar{z}^{j+1} = \Pi_{\mathcal{Z}} \left(\frac{\sum_{i=1}^{n_a} w_i (z_i^{j+1} + \xi_i^j)}{\sum_{i=1}^{n_a} w_i} \right). \tag{4.15}$$

The dual variables are updated as

$$\xi_i^{j+1} = \xi_i^j + (z_i^{j+1} - \bar{z}^{j+1}). \tag{4.16}$$

This process is repeated until convergence.

In the aerobraking setting, OS-SCP enables multiple candidate trajectory families to evolve simultaneously, allowing the optimizer to explore distinct atmospheric entry profiles and intermediate orbital passes before converging toward consensus.

4.3.2 Initial Guess Generation and Agent Initialization

To employ the OS-SCP algorithm effectively, each agent must have a unique initial guess which ideally lie within the basin of attraction of distinct local minima. As discussed in Section 3.4, effective exploration requires the initial agent trajectories to span diverse regions of the solution space.

To achieve this for the aerobraking problem, we consider two strong channels in the solution space that are inherently driven by the dynamics of the problem: the number of aerobrake passes and the time spent in the atmosphere per pass. Both of these factors are determined by the direction and magnitude of ΔV at apogee (which determines flight path angle at entry and perigee reduction of the initial orbit) and the commanded aerodynamic control through each pass.

Initial guesses are then generated through shooting the nonlinear dynamics over a grid of commanded lift and ΔV direction and magnitude. Shooting the dynamics over this grid produces the trajectories shown in Figure 4.2. Through experimentation, we found that taking one minimum and maximum lift and perigee target guesses across three, four, and five pass guesses provided a wide range of initializations to properly explore the solution space. These are automatically selected from the set of trajectories shown in Figure 4.2. However, as shown in the numerical example to follow, even small intuitive changes to initialization of agents without the need for shooting can make drastic improvements to the quality of the computed trajectories.

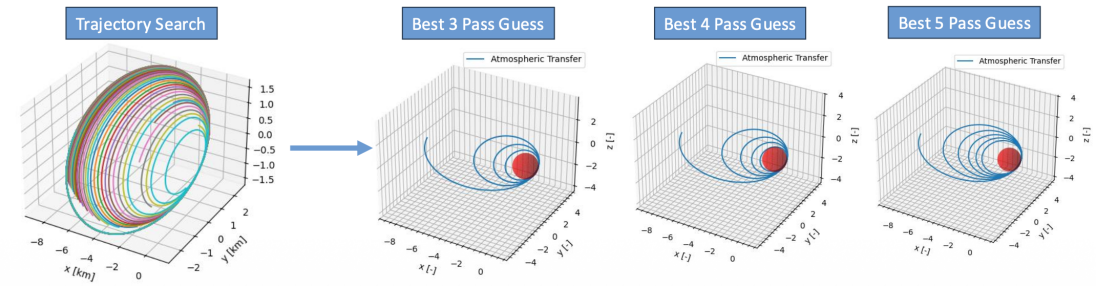


Figure 4.2: All Simulated Trajectories Over Grid of ΔV and Commanded Lift

4.4 Numerical Example

This section evaluates the performance of the proposed OS-SCP framework on an example aerobraking trajectory optimization problem and compares it against standard SCP.

4.4.1 Problem Definition

Consider the example problem, where a spacecraft in an initial orbit shown in Table 4.1 must reach the target orbit shown in Table 4.1 with minimal ΔV. Note that the final position of the spacecraft on the target orbit is free.

Orbital Elements	Initial Orbit	Target Orbit
Semi-major axis (km)	42,164	500 + 6378 = 6878
Eccentricity	0	0
Inclination (deg)	0	28.5
Longitude of the ascending node (deg)	0	0
Argument of perigee (deg)	0	0
Mean anomaly (deg)	0	Free

Table 4.1: Example orbital transfer problem parameters.

4.4.2 Standard SCP Solution

For the initial guess, we use a 3-pass trajectory with a perigee altitude of 90 km, an inclination of 20°, and a 90° bank angle at maximum angle of attack to maximize plane change during atmospheric flight. After three atmospheric passes, the guess trajectory lowers the apoapsis altitude to approximately 15 600 km.

Given this initial guess, we can express and solve this problem using the CVXPY package with the QOCO convex solver [24]. Each SCP iteration takes about 100 ms, so hundreds of iterations can be performed in under a minute. This speed comes from the small problem size of 13 nodes, which is possible due to exact discretization, and the fast

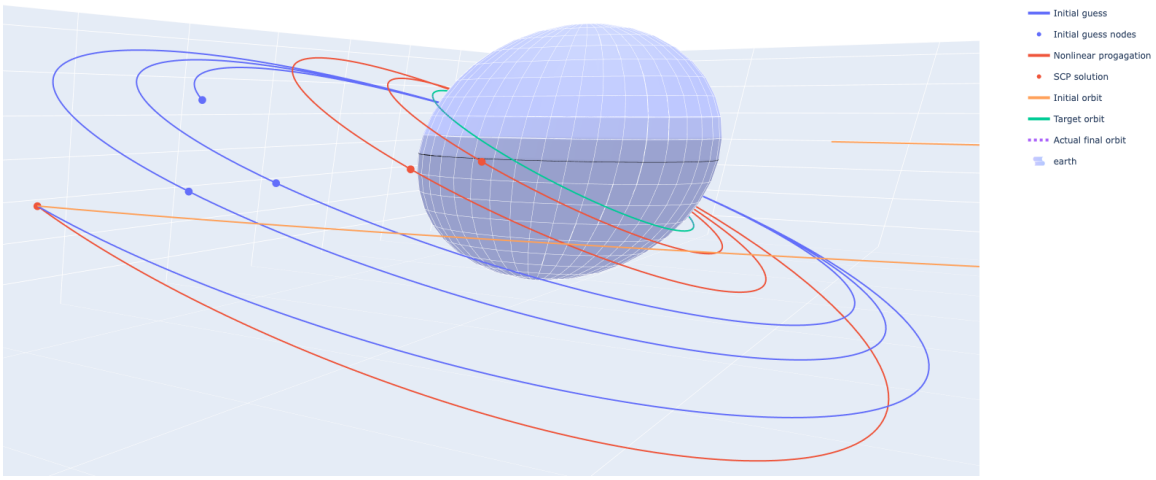


Figure 4.3: Standard SCP solution trajectory vs. initial guess

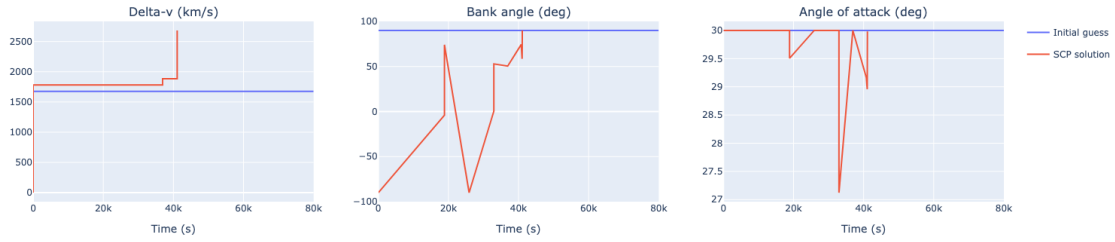


Figure 4.4: Standard SCP optimal control profile vs. initial guess

discretize-then-linearize method of discretization [30]. The optimizer reaches the maximum number of iterations at 1000 iterations without converging, returning the trajectory at the 1000^{th}. In three atmospheric passes, the optimizer achieves a $503 \times 497\,\text{km}$ orbit at an inclination of $28.50°$. The final ΔV is $2684\,\text{m/s}$, and this solution notably gets trapped in a local minimum which cannot eliminate intermediate burns and requires a $104\,\text{m/s}$ burn at the apogee of the second pass. The initial guess and optimized trajectories are shown in Figure 4.3, and the control time history is shown in Figure 4.4.

4.4.3 OS-SCP Solution

For the OS-SCP solution, we initialize three agents. Similar to the standard SCP solution, each agent has an initial guess of a 3-pass trajectory with an inclination of 20°, and a 90° bank angle, but each agent differs in targeted perigee altitudes. By targeting perigee altitudes of 85, 88, and 90 km, the agents are effectively initialized into nearby but distinct local basins of attraction.

After 197 iterations, the agents converge to a consensus trajectory that has reached a stable local minimum. The consensus trajectory reaches the target orbit exactly at $500 \times 500\,\text{km}$ at an inclination of 28.5°. The final ΔV is $1937\,\text{m/s}$, reducing the required ΔV from the standard SCP solution by about 27%. Additionally, note in Figure 4.6 that the OS-SCP consensus solution requires no intermediate burns; the solution only requires ΔV maneuvers at the first node to initialize the aerobrake maneuver and at the final node to circularize the orbit. The OS-SCP solution is more optimal than the standard SCP solution in all considered aspects.

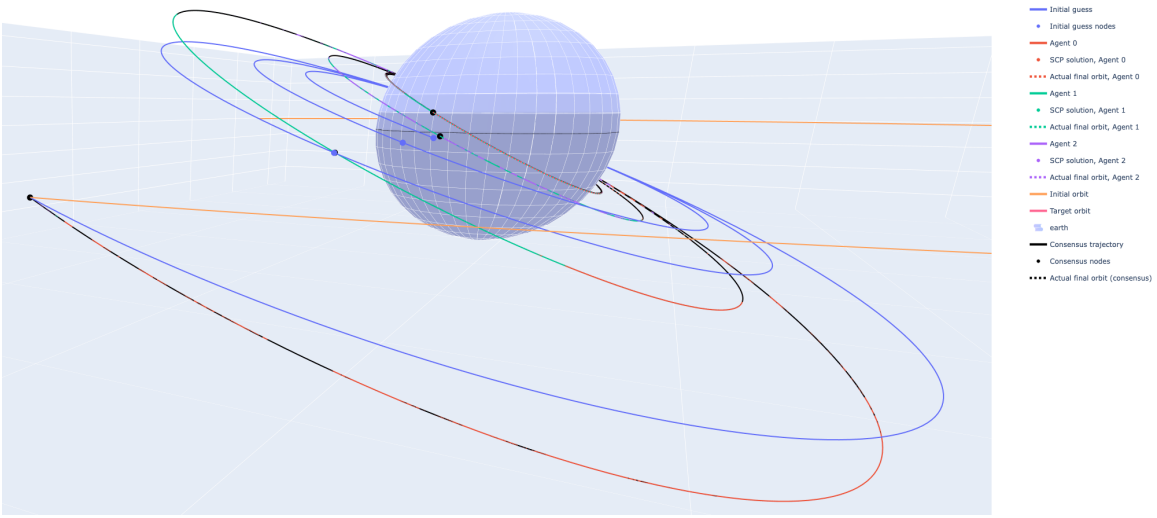


Figure 4.5: OS-SCP solution trajectory

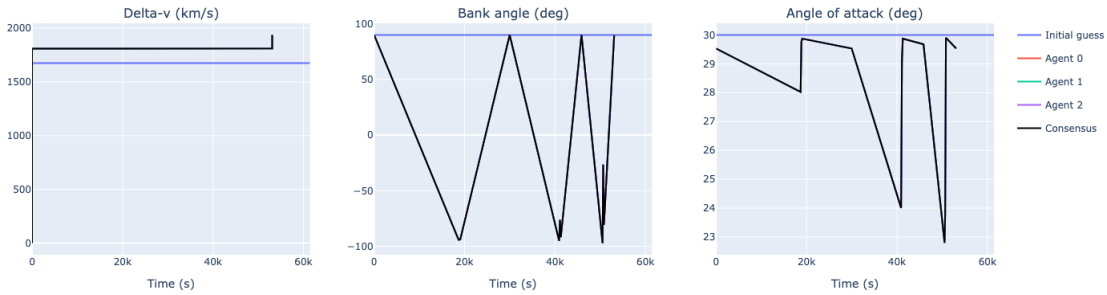


Figure 4.6: OS-SCP optimal control profile

4.5 Conclusions

This chapter presented the application of the proposed operator-splitting successive convex programming (OS-SCP) framework to a challenging aerobraking trajectory optimization problem. The problem exhibits highly nonlinear, sensitive, and multi-regime dynamics arising from coupled orbital and atmospheric flight, making it particularly susceptible to local minima and initialization sensitivity when solved using standard SCP methods.

To address these challenges, the chapter introduced a discretization and initialization strategy tailored to multi-pass aerobraking trajectories, together with an OS-SCP implementation capable of evolving multiple coupled trajectory hypotheses simultaneously. By initializing agents in neighboring but distinct regions of the solution space, OS-SCP was able to explore multiple nearby local basins of attraction while still leveraging the computational efficiency and constraint-handling capabilities of SCP.

The numerical results demonstrated that standard SCP became trapped in a suboptimal local minimum containing unnecessary intermediate burns. In contrast, OS-SCP converged to a substantially lower-cost solution corresponding to an approximately 27% reduction in total ΔV. Importantly, OS-SCP achieved this improved solution while converging in significantly fewer iterations than standard SCP.

These results highlight the effectiveness of OS-SCP as a practical framework for exploring highly nonconvex trajectory optimization landscapes that are difficult to solve using

conventional SCP alone. In particular, the ability of OS-SCP to coordinate multiple nearby trajectory hypotheses appears well-suited for atmospheric guidance problems where small perturbations can produce qualitatively different trajectory families.

Future work will investigate deorbit and entry, descent and landing (EDL) flight planning. Additional future directions include robustness to atmospheric uncertainty, real-time onboard guidance applications, and extension to fully integrated aerocapture and powered descent guidance problems.

Chapter 5

STRUCTURAL OPTIMIZATION NUMERICAL EXAMPLES

While the primary inspiration of this work is the application of OS-SCP to nonlinear trajectory optimization problems, the proposed framework is not limited to dynamical systems or optimal control. OS-SCP is intended as a general-purpose methodology for exploring nonconvex optimization landscapes through coupled local optimization processes.

To demonstrate this broader applicability, this chapter considers two classical structural optimization benchmarks: the 10-bar planar truss problem and the 72-bar space truss problem [1]. These problems differ significantly from the aerobraking trajectory optimization problems considered previously. Rather than involving nonlinear dynamics and path constraints, the structural optimization problems considered here are governed by finite element equilibrium equations and structural performance constraints, including stress, displacement, and natural frequency limits.

These benchmarks are particularly well-suited for evaluating OS-SCP because they exhibit many of the characteristics that challenge local optimization methods. The design spaces contain numerous local minima associated with different structural load paths and sparsity patterns, while the constraints introduce strong nonlinear coupling between design variables. As a result, standard SCP methods often converge to locally optimal structures strongly influenced by initialization.

Thus, there are two main objectives of this chapter. First, we demonstrate that the proposed OS-SCP framework extends naturally beyond trajectory optimization into structural optimization. Second, we investigate whether the exploration mechanisms introduced by OS-SCP enable convergence to competitive or improved structural designs relative to other structural optimization methods from literature.

5.1 10-Bar Truss Optimization

5.1.1 Problem Definition

The first structural benchmark considered is the classical 10-bar planar truss optimization problem. The objective is to minimize structural mass while satisfying stress and displacement constraints under a prescribed loading condition.

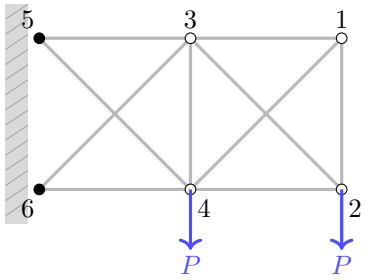


Figure 5.1: Nominal 10-bar truss topology used for sparse-structure initialization.

The truss consists of 10 members connecting 6 nodes in a planar configuration. A depiction of the truss is shown in Figure 5.1. The truss members have a Young's modulus of 10 Msi and a density of 0.1 lb/in^3. Two load cases are considered: Case 1 has loads $P_1 = 100$ kips and $P_2 = 0$ kips, and Case 2 has loads $P_1 = 150$ kips and $P_2 = 50$ kips. The design variables are the cross-sectional areas of each member,

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{10} \end{bmatrix}^T,$$

subject to box constraints

$$A_{\min} \leq A_i \leq A_{\max}$$

where the minimum and maximum allowable areas are 0.1 and 35 in^2 respectively. This problem includes 22 nonlinear constraints on element stresses and nodal displacements [1].

The optimization problem is formulated as

$$\min_{A} \quad \sum_{i=1}^{10} \rho L_i A_i \tag{5.1}$$

$$\text{s.t.} \quad |\sigma_i(A)| \leq \sigma_{\max}, \tag{5.2}$$

$$|u_j(A)| \leq u_{\max}, \tag{5.3}$$

$$A_{\min} \leq A_i \leq A_{\max}, \tag{5.4}$$

where ρ is the material density, L_i is the length of member i, σ_i is the axial stress in member i, and u_j denotes nodal displacements.

5.1.2 Finite Element Formulation

The structural response is computed using a standard linear finite element truss model. For each member, the local stiffness matrix is given by

$$k_e = \frac{A_i E}{L_i} \begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix}, \tag{5.5}$$

where c and s are the directional cosines of the element.

The global stiffness matrix is assembled as

$$K(A) = \sum_{i=1}^{10} k_e^{(i)}.$$

The equilibrium equations are then solved as

$$K(A)u = f,$$

where u denotes nodal displacements and f is the applied load vector.

5.1.3 Sequential Convex Programming Formulation

The 10-bar truss optimization problem is nonconvex due to the nonlinear dependence of the stress and displacement constraints on the structural design variables through the finite

element equilibrium equations. To solve this problem using SCP, the nonlinear constraints are successively linearized about a reference design iterate.

Let

$$A^j = \begin{bmatrix} A_1^j & A_2^j & \cdots & A_{10}^j \end{bmatrix}^T$$

denote the design vector at SCP iteration j. The nonlinear constraint vector is written compactly as

$$g(A) \leq 0,$$

where $g(A)$ contains all stress and displacement constraints.

At each iteration, the constraints are linearized using a first-order Taylor expansion about the current reference design:

$$g(A) \approx g(A^j) + \nabla g(A^j)(A - A^j). \tag{5.6}$$

The resulting convex SCP subproblem is then formulated as

$$\min_{A,s} \quad \sum_{i=1}^{10} \rho L_i A_i + \lambda_s \sum_{k=1}^{n_g} s_k + \frac{w_p}{2} \left\| A - A^j \right\|_2^2 \tag{5.7}$$

$$\text{s.t.} \quad g(A^j) + \nabla g(A^j)(A - A^j) \leq s, \tag{5.8}$$

$$s \geq 0, \tag{5.9}$$

$$A_{\min} \leq A_i \leq A_{\max}, \tag{5.10}$$

$$\left\| A - A^j \right\|_\infty \leq \Delta^j, \tag{5.11}$$

where s are nonnegative slack variables used to preserve feasibility of the linearized constraints, λ_s is a penalty weight on constraint violation, w_p is the proximal trust-region weight, Δ^j is the trust-region radius at iteration j.

The proximal penalty and trust-region constraint serve to limit deviation from the current linearization point, thereby maintaining validity of the local convex approximation. After solving the convex subproblem, the reference design is updated according to

$$A^{j+1} \leftarrow A^\star,$$

where $A^\star$ denotes the solution of the SCP subproblem.

This process is repeated until convergence of both the design variables and constraint residuals.

5.1.4 OS-SCP Implementation

The 10-bar truss problem presents a nonconvex design landscape containing multiple feasible load paths and sparse structural configurations. In particular, many locally optimal solutions differ primarily in which members become effectively inactive by converging toward the minimum allowable area.

To explore these distinct structural topologies, multiple OS-SCP agents are initialized using diverse structural templates which target specific load paths. This is achieved by selectively initializing specific members to be sparse to force specific structural load paths. Examples of this sparse member initialization technique is shown in Figure 5.2.

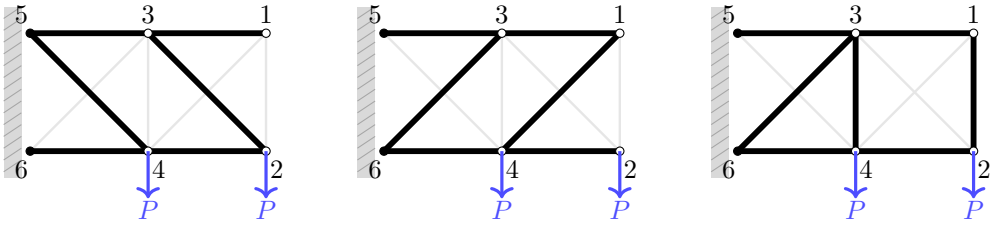


Figure 5.2: Example sparse initializations for OS-SCP agents. Dark members have larger cross-sectional areas, while faint members have the minimum allowable area.

Weighted consensus is used to bias the consensus trajectory toward lower-mass feasible structures. We found a benefit of assigning consensus weight through a combination of total structure weight and constraint violation. At iteration j, each agent is assigned a scalar score

$$S_i^j = W(A_i^j) + \mu \max\left(0, \max_{\ell} g_\ell(A_i^j)\right), \tag{5.12}$$

where $W(A_i^j)$ is the structural weight of agent i, $g_\ell(A_i^j) \leq 0$ are the stress and displacement constraints, and $\mu = 10^4$ is the constraint-violation penalty used in the merit function to balance the magnitude of the terms.

The scores are normalized according to

$$\tilde{S}_i^j = \frac{S_i^j - \min_r S_r^j}{\max_r S_r^j - \min_r S_r^j}, \tag{5.13}$$

with $\tilde{S}_i^j = 0$ if all agents have identical scores. The consensus weights are then computed using a softmin rule,

$$w_i^j = \frac{\exp(-5\tilde{S}_i^j)}{\sum_{r=1}^{n_a} \exp(-5\tilde{S}_r^j)}. \tag{5.14}$$

These weights are used to define an agent-dependent consensus penalty,

$$\rho_i^j = \rho_{\text{eff}}^j(\epsilon_w + w_i^j) + 10^{-12}, \tag{5.15}$$

where $\epsilon_w = 10^{-2}$ prevents any agent from receiving zero weight.

The updates are performed in normalized log-area coordinates. Defining

$$x_i^j = \frac{\log(A_i^j) - \log(A_{\min})}{\log(A_{\max}) - \log(A_{\min})}, \tag{5.16}$$

the weighted average is computed as

$$\bar{x}^{j+1} = \frac{\sum_{i=1}^{n_a} \rho_i^j x_i^{j+1}}{\sum_{i=1}^{n_a} \rho_i^j}. \tag{5.17}$$

The consensus design is then mapped back to physical area variables using

$$\bar{A}^{j+1} = \exp\left(\log(A_{\min}) + \bar{x}^{j+1}\left[\log(A_{\max}) - \log(A_{\min})\right]\right). \tag{5.18}$$

Then, we use the relaxation technique for the consensus step. Let $i^\star$ denote the feasible agent with the lowest score. Then the final consensus update is

$$\bar{A}^{j+1} = \alpha^j A_{i^\star}^{j+1} + (1 - \alpha^j)\bar{A}_{\text{avg}}^{j+1}, \tag{5.19}$$

where α^j is the relaxation parameter weight. As noted in section 3.3.1, under-relaxation is advantageous here to purposefully delay consensus formation to promote exploration of the solution space.

5.1.5 Results

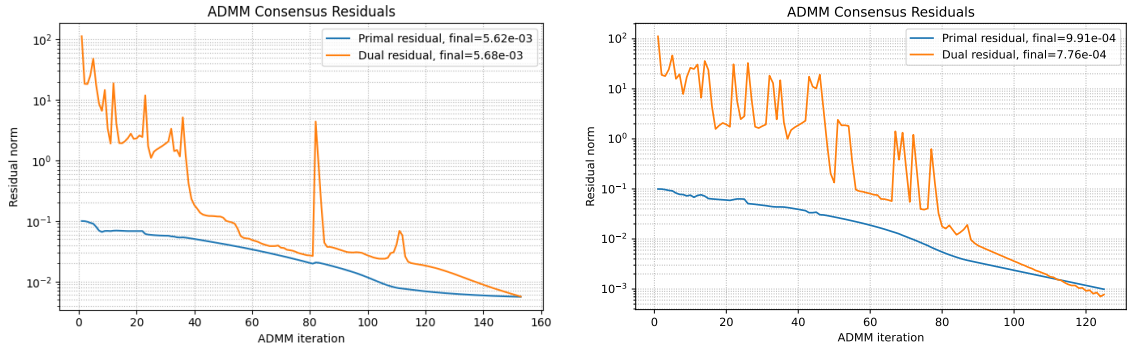
The 10-bar truss optimization problem has been studied extensively in optimization literature, with many algorithms using this problem as a benchmark. Some prominent examples of solutions to this problem are tabulated in Tables 5.1 and 5.2. Note that some of these methods claim to solve the problem with lower mass, but use penalty methods to enforce constraints which tolerate small violations of the constraints and are actually infeasible. All solutions were re-evaluated using the finite element model and constraint definitions used in this work, and methods which violate constraints are noted in the table. The OS-SCP method performs well in this problem, achieving the same or lower weight as the minimum feasible solution found in literature across both load cases. It achieves this optimal value for Cases 1 and 2 in 153 and 124 iterations respectively. All agents reach converge to the consensus, with the primal and dual residuals shown in Figures 5.3b.

Table 5.1: Comparison of optimization results with literature: Case 1 of 10-bar truss problem. Recreated from Kim and Byun [1].

	HS	PSOPC	HPSO	PSO	HPSACO	IHS	EHS	SAHS	ABC-AP	TLBO	CNNT-PSO	OS-SCP
Citation	[31]	[32]	[32]	[33]	[34]	[35]	[36]	[36]	[37]	[38]	[1]	
A_1	30.15	30.569	30.704	33.500	30.307	30.5222	30.208	30.394	30.548	30.4286	30.5171	30.5557
A_2	0.102	0.100	0.100	0.100	0.100	0.1000	0.100	0.100	0.100	0.1000	0.1000	0.100
A_3	22.71	22.974	23.167	22.766	23.434	23.2005	22.698	23.098	23.180	23.2436	23.1999	23.1904
A_4	15.27	15.148	15.183	14.417	15.505	15.2232	15.275	15.491	15.218	15.3677	15.2198	15.2136
A_5	0.102	0.100	0.100	0.100	0.100	0.1000	0.100	0.100	0.100	0.1000	0.1000	0.100
A_6	0.544	0.547	0.551	0.100	0.5241	0.5513	0.529	0.529	0.551	0.5751	0.5512	0.549
A_7	7.541	7.493	7.460	7.534	7.4365	7.4572	7.558	7.488	7.463	7.4404	7.4574	7.4579
A_8	21.56	21.159	20.978	20.467	21.079	21.0367	21.559	21.189	21.058	20.9665	21.03931	21.025
A_9	21.45	21.156	21.508	20.392	21.229	21.5288	21.491	21.342	21.501	21.5330	21.5310	21.5301
A_{10}	0.100	0.100	0.100	0.100	0.100	0.1000	0.100	0.100	0.100	0.1000	0.1000	0.100
Weight (lb)	5058.336	5040.669	5060.906	5024.248	5056.59	5060.93	5062.39	5061.275	5060.888	5060.956	5060.8540	5060.8461
Feasibility	Infeasible	Infeasible	Feasible	Infeasible	Infeasible	Feasible	Feasible	Feasible	Feasible	Feasible	Feasible	Feasible
Infeasible node disp. (in)	−2.0018	−2.0085	None	−2.0389	−2.002	None	None	None	None	None	None	None
Infeasible bar stress (ksi)	None	25.00118	None	25.01712	None	None	None	None	None	None	None	None

Table 5.2: Comparison of optimization results with literature: Case 2 of 10-bar truss problem. Recreated from Kim and Byun [1].

Citation	HS [31]	PSOPC [32]	HPSO [32]	HPSACO [34]	EHS [36]	SAHS [36]	ABC-AP [37]	TLBO [38]	CNNT-PSO [1]	OS-SCP
A_1	23.250	23.473	23.353	23.194	23.589	23.525	23.4692	23.524	23.5186	23.5002
A_2	0.1020	0.101	0.100	0.100	0.101	0.100	0.1005	0.1000	0.1000	0.100
A_3	25.730	25.287	25.502	24.585	25.422	25.429	25.2393	25.441	25.2897	25.2543
A_4	14.510	14.413	14.250	14.221	14.488	14.488	14.354	14.479	14.3685	14.3512
A_5	0.1000	0.100	0.100	0.100	0.100	0.100	0.1001	0.1000	0.1000	0.100
A_6	1.9770	1.969	1.972	1.969	1.975	1.992	1.9701	1.995	1.9697	1.9697
A_7	12.210	12.362	12.363	12.489	12.362	12.352	12.4128	12.334	12.3894	12.3997
A_8	12.610	12.694	12.984	12.925	12.682	12.698	12.8925	12.689	12.8299	12.8504
A_9	20.360	20.323	20.356	20.952	20.322	20.341	20.3343	20.354	20.3371	20.3505
A_{10}	0.100	0.103	0.1010	0.101	0.100	0.100	0.1000	0.1000	0.1000	0.100
Weight (lb)	4669.365	4667.76	4681.93	4675.797	4679.0148	4678.8476	4677.0754	4678.3149	4676.9239	4676.6146
Feasibility	Infeasible	Infeasible	Infeasible	Infeasible	Feasible	Feasible	Feasible	Feasible	Feasible	Feasible
Infeasible node disp. (in)	−2.0039	−2.005	None	−2.00158	None	None	None	None	None	None
Infeasible bar stress (ksi)	25.04062	25.00876	25.07655	25.00189	None	None	None	None	None	None



(a) Case 1 residuals

(b) Case 2 residuals

Figure 5.3: OS-SCP primal and dual residuals for the 10-bar truss optimization problem, load cases 1 and 2.

5.2 72-Bar Space Truss Optimization

5.2.1 Problem Definition

The second structural benchmark considered is the classical 72-bar space truss optimization problem. Unlike the 10-bar planar truss, this problem is a three-dimensional tower structure

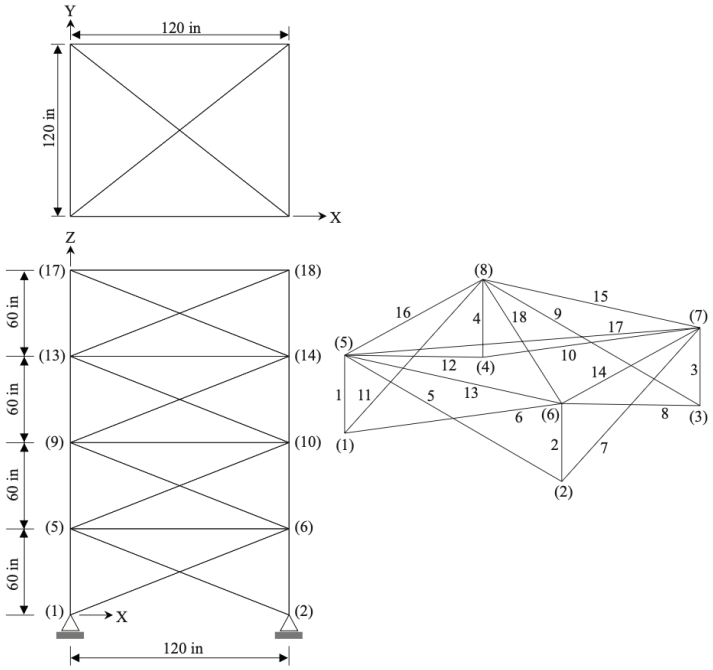


Figure 5.4: 72 bar Truss [1]

whose objective is dynamic rather than static. The goal is to minimize structural mass while satisfying lower bounds on selected natural frequencies.

The truss consists of 20 nodes and 72 members arranged as a four-story space frame. Each story has height $L = 60$ in, and each floor has a square footprint of side length $2L$. The base nodes are fixed, while all nodes above the base are free to translate in three dimensions. The node numbering follows a top-to-bottom convention: nodes 1–4 are located at the top story, nodes 5–8 at the third story, nodes 9–12 at the second story, nodes 13–16 at the first story, and nodes 17–20 at the base. The base nodes 17–20 are fully pinned in all three translational degrees of freedom. A visual depiction of the 72 bar truss is shown in Figure 5.4.

The material properties used in this study are

$$E = 10 \text{ Msi}, \qquad \rho = 0.1 \text{ lb/in}^3.$$

The 72 members are partitioned into 16 design-variable groups. All members within a

group share a common cross-sectional area. The design vector is therefore

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{16} \end{bmatrix}^T,$$

where A_g denotes the cross-sectional area assigned to all members in group g. The group areas are subject to the bound constraints.

$$A_{\min} \leq A_g \leq A_{\max},$$

with

$$A_{\min} = 0.1 \text{ in}^2, \qquad A_{\max} = 4.0 \text{ in}^2.$$

The member grouping used in the optimization is summarized in Table 5.3. The grouping exploits the symmetry and repeated structural patterns of the tower, reducing the design dimension from 72 independent member areas to 16 grouped design variables.

The structure is optimized under two independent loading conditions. In Load Case 1, node 17 is subjected to

$$P_x = 5 \text{ kips}, \qquad P_y = 5 \text{ kips}, \qquad P_z = -5 \text{ kips}.$$

In Load Case 2, nodes 17–20 are each subjected to

$$P_z = -5 \text{ kips}.$$

The optimization problem includes stress constraints on all 72 members under both load cases, with allowable tensile and compressive stress

$$|\sigma_i| \leq 25 \text{ ksi}, \qquad i = 1, \ldots, 72.$$

In addition, the displacements of the top nodes 17–20 are constrained in both the X and Y directions:

$$|u_j| \leq 0.25 \text{ in}.$$

Table 5.3: Design-variable groups for the 72-bar space truss.

Group	Member type
1	Vertical members between nodes 1–4 and 5–8
2	Cross-diagonal members between nodes 1–4 and 5–8
3	Horizontal perimeter members at the top story
4	Horizontal diagonal members at the top story
5	Vertical members between nodes 5–8 and 9–12
6	Cross-diagonal members between nodes 5–8 and 9–12
7	Horizontal perimeter members at story 3
8	Horizontal diagonal members at story 3
9	Vertical members between nodes 9–12 and 13–16
10	Cross-diagonal members between nodes 9–12 and 13–16
11	Horizontal perimeter members at story 2
12	Horizontal diagonal members at story 2
13	Vertical members between nodes 13–16 and 17–20
14	Cross-diagonal members between nodes 13–16 and 17–20
15	Horizontal perimeter members at story 1
16	Horizontal diagonal members at story 1

The optimization problem is formulated as

$$\min_{A} \quad \sum_{m=1}^{72} \rho L_m A_{g(m)} \tag{5.20}$$

$$\text{s.t.} \quad |\sigma_m^{(q)}(A)| \le \sigma_{\max}, \qquad m = 1, \ldots, 72, \quad q = 1, 2, \tag{5.21}$$

$$|u_j^{(q)}(A)| \le u_{\max}, \qquad j \in \mathcal{D}_{\text{top}}, \quad q = 1, 2, \tag{5.22}$$

$$A_{\min} \le A_g \le A_{\max}, \qquad g = 1, \ldots, 16, \tag{5.23}$$

$$f_1(A) \ge 4.0 \text{ Hz}, \tag{5.24}$$

$$f_3(A) \ge 6.0 \text{ Hz}, \tag{5.25}$$

$$\tag{5.26}$$

where q indexes the load case, $g(m)$ maps member m to its design-variable group, $\mathcal{D}_{\text{top}}$ denotes the constrained X- and Y-direction displacement degrees of freedom at nodes 17–20, $\sigma_{\max} = 25$ ksi, and $u_{\max} = 0.25$ in. L_m is the length of member m, and $f_1(A)$ and $f_3(A)$ denote the first and third natural frequencies of the structure. The constraint vector used in the numerical implementation is written as

$$g(A) = \begin{bmatrix} 4.0 - f_1(A) \\ 6.0 - f_3(A) \end{bmatrix} \leq 0.$$

Thus, the 72-bar problem considered here contains two nonlinear frequency constraints. This problem contains 16 design variables and 264 nonlinear constraints.

5.2.2 Finite Element Formulation

The structural response is computed using a standard three-dimensional truss finite element model. Each node has three translational degrees of freedom. For member m, let

$$\ell_x, \ell_y, \ell_z$$

denote the direction cosines of the member axis, and define

$$c_m = \begin{bmatrix} \ell_x & \ell_y & \ell_z \end{bmatrix}^T.$$

The element stiffness matrix is

$$k_m = \frac{A_{g(m)} E}{L_m} \begin{bmatrix} c_m c_m^T & -c_m c_m^T \\ -c_m c_m^T & c_m c_m^T \end{bmatrix}. \tag{5.27}$$

The global stiffness matrix is assembled as

$$K(A) = \sum_{m=1}^{72} k_m.$$

For each load case q, the nodal displacement vector is obtained by solving

$$K(A) u^{(q)} = f^{(q)},$$

with the fixed degrees of freedom removed. Member axial stresses are then computed from the corresponding element displacement vectors.

5.2.3 Sequential Convex Programming Formulation

As in the 10-bar case, the nonlinear constraints are handled using sequential convex programming.

Let

$$A^j = \begin{bmatrix} A_1^j & A_2^j & \cdots & A_{16}^j \end{bmatrix}^T$$

denote the group-area design vector at SCP iteration j. The nonlinear frequency constraints are written compactly as

$$g(A) \leq 0,$$

where

$$g(A) = \begin{bmatrix} 4.0 - f_1(A) \\ 6.0 - f_3(A) \end{bmatrix}.$$

At each SCP iteration, the constraints are linearized about the current design:

$$g(A) \approx g(A^j) + \nabla g(A^j)(A - A^j). \tag{5.28}$$

The Jacobian $\nabla g(A^j)$ is computed using finite differences with respect to the 16 group-area variables.

The convex SCP subproblem is then

$$\min_{A,s} \quad \sum_{m=1}^{72} \rho L_m A_{g(m)} + \lambda_s \sum_{k=1}^{2} s_k + \frac{w_p}{2} \left\| A - A^j \right\|_2^2 \tag{5.29}$$

$$\text{s.t.} \quad g(A^j) + \nabla g(A^j)(A - A^j) \leq s, \tag{5.30}$$

$$s \geq 0, \tag{5.31}$$

$$A_{\min} \leq A_g \leq A_{\max}, \qquad g = 1, \ldots, 16, \tag{5.32}$$

$$\left\| A - A^j \right\|_\infty \leq \Delta^j, \tag{5.33}$$

where slack variables are used to preserve feasibility of the convex subproblem, and the trust-region constraint limits the step size to maintain validity of the linearization.

5.2.4 OS-SCP Implementation

Each OS-SCP agent optimizes the 16 grouped cross-sectional areas. The initial population is generated using structured templates in normalized log-area space. The structured tem-

plates represent qualitatively different tower designs, including uniform-area designs and designs which target specific load paths using sparsity of members, similar to the 10-bar truss problem.

As in the 10-bar case, the normalized log-area coordinates are defined by

$$x_g = \frac{\log(A_g) - \log(A_{\min})}{\log(A_{\max}) - \log(A_{\min})}. \tag{5.34}$$

Consensus is then computed in this normalized design space and mapped back to physical cross-sectional areas. This prevents the consensus operation from being dominated by the largest area variables and provides a more balanced interpolation between sparse and dense structural designs.

Each agent is scored using the merit function from the 10-bar truss problem described in Equations 5.12 - 5.15.

The consensus design is computed as a weighted average in log-area coordinates:

$$\bar{x}^{j+1} = \frac{\sum_{i=1}^{n_a} \rho_i^j x_i^{j+1}}{\sum_{i=1}^{n_a} \rho_i^j}. \tag{5.35}$$

The physical consensus areas are then recovered by

$$\bar{A}^{j+1} = \exp\left(\log(A_{\min}) + \bar{x}^{j+1}\left[\log(A_{\max}) - \log(A_{\min})\right]\right).$$

This log-space representation is useful because the admissible areas span more than an order of magnitude, and averaging in log space gives a more balanced interpolation between small and large member groups than direct averaging in physical area coordinates.

5.2.5 Results

Similarly to the 10-bar truss, the 72-bar truss optimization problem has been studied extensively in optimization literature. Solutions to this problem from literature in both load cases are tabulated in Tables 5.1 and 5.2. Like the 10-bar, some of these tabulated methods claim to solve the problem with lower mass but use penalty methods to enforce constraints which tolerate small violations of the constraints. Again, methods which violate constraints are noted in the table. The OS-SCP method performs well in this problem, achieving the same or lower weight as the minimum feasible solution found in literature across both load

cases. It achieves this optimal value for Cases 1 and 2 in 394 and 387 iterations respectively. All agents reach converge to the consensus, with the primal and dual residuals shown in Figure 5.5.

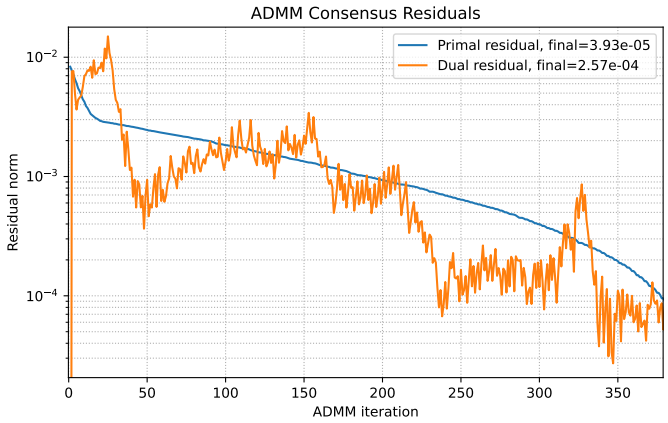


Figure 5.5: Case 1 residuals

5.3 Structural Optimization Remarks

The structural optimization benchmarks considered in this chapter demonstrate that OS-SCP extends naturally beyond trajectory optimization problems. Across both benchmarks, the multi-agent exploration mechanisms introduced by OS-SCP were observed to improve the ability of the optimizer to discover lower-cost feasible solutions relative to standard SCP. Additionally, OS-SCP performed as well as, or better than, the best results from literature across objective value, constraint satisfaction, and iterations to convergence. These examples support the usage of OS-SCP as a general framework for exploration-driven nonconvex optimization.

Table 5.4: Comparison of optimization results with literature: Case 1 of 72-bar truss problem. Recreated from Kim and Byun [1].

	HS	PSO	PSOPC	HPSO	BB-BC	HBB-BC	EHS	SAHS	TLBO	CNNT-PSO	OS-SCP
Citation	[31]	[33]	[32]	[32]	[39]	[34]	[36]	[36]	[38]	[1]	
$A_1 - A_4$	1.790	1.7427	1.855	1.857	1.8577	1.9042	1.967	1.860	1.9064	1.8861	1.8861
$A_5 - A_{12}$	0.521	0.5185	0.504	0.505	0.5059	0.5162	0.510	0.521	0.50612	0.5123	0.5123
$A_{13} - A_{16}$	0.100	0.1000	0.100	0.100	0.1000	0.1000	0.100	0.100	0.100	0.1000	0.1000
$A_{17} - A_{18}$	0.100	0.1000	0.100	0.100	0.1000	0.1000	0.100	0.100	0.100	0.1000	0.1000
$A_{19} - A_{22}$	1.229	1.3079	1.253	1.255	1.2476	1.2582	1.293	1.271	1.2617	1.2685	1.2685
$A_{23} - A_{30}$	0.522	0.5193	0.505	0.503	0.5269	0.5035	0.511	0.509	0.5111	0.5117	0.5117
$A_{31} - A_{34}$	0.100	0.1000	0.100	0.100	0.1000	0.1000	0.100	0.100	0.100	0.1000	0.1000
$A_{35} - A_{36}$	0.100	0.1000	0.100	0.100	0.1012	0.1000	0.100	0.100	0.100	0.1000	0.1000
$A_{37} - A_{40}$	0.517	0.5142	0.497	0.496	0.5209	0.5178	0.499	0.485	0.5317	0.5236	0.5236
$A_{41} - A_{48}$	0.504	0.5464	0.508	0.506	0.5172	0.5214	0.501	0.501	0.51591	0.5171	0.5171
$A_{49} - A_{52}$	0.100	0.1000	0.100	0.100	0.1004	0.1000	0.100	0.100	0.100	0.1000	0.1000
$A_{53} - A_{54}$	0.101	0.1095	0.100	0.100	0.1005	0.1007	0.100	0.100	0.100	0.1000	0.1000
$A_{55} - A_{58}$	0.156	0.1615	0.100	0.100	0.1565	0.1566	0.160	0.168	0.5162	0.1565	0.1565
$A_{59} - A_{66}$	0.547	0.5092	0.525	0.524	0.5507	0.5421	0.522	0.584	0.54927	0.5456	0.5456
$A_{67} - A_{70}$	0.442	0.4967	0.394	0.400	0.3922	0.4132	0.478	0.433	0.40966	0.4103	0.4103
$A_{71} - A_{72}$	0.590	0.5619	0.535	0.534	0.5922	0.5756	0.591	0.520	0.56976	0.5697	0.5697
Weight (lb)	379.217	381.936	369.65	369.64	379.64	379.65	381.026	380.837	379.63494	379.6148	379.6140
Feasibility	Infeasible	Feasible	Infeasible	Infeasible	Feasible	Feasible	Feasible	Feasible	Feasible	Feasible	Feasible
Infeasible node disp. (in)	0.2505456	None	0.3114043	0.3115011	None	None	None	None	None	None	None
Infeasible bar stress (ksi)	25.01883	None	34.762759	34.768775	None	None	None	None	None	None	None

Chapter 6

CONCLUSIONS

This thesis presented Operator Splitting Sequential Convex Programming (OS-SCP), a framework designed to improve exploration of nonconvex solution spaces within sequential convex programming. While SCP methods have demonstrated strong performance across a wide range of aerospace and engineering trajectory optimization problems, they remain fundamentally local methods whose convergence behavior is strongly dependent on initialization. In highly nonconvex problems, this often causes standard SCP to converge to suboptimal local minima.

To address this limitation, this work proposed embedding SCP within a consensus-based operator splitting framework inspired by the alternating direction method of multipliers (ADMM). In the proposed method, multiple SCP trajectories evolve simultaneously as coupled agents, with consensus updates enabling information exchange between them. This formulation allows intermediate trajectories to emerge between the initializations, enabling exploration of neighboring local minima that may not be reachable using standard SCP from a single initial guess.

Several extensions to the baseline consensus formulation were introduced to improve robustness and exploration capability, including weighted consensus, relaxation, and clique consensus updates. These modifications were shown to improve convergence and exploration behavior while preserving the computational advantages of SCP-based subproblems.

The proposed framework was evaluated on several challenging nonlinear optimization problems. In structural optimization examples involving the 10-bar and 72-bar truss problems, OS-SCP consistently achieved solutions competitive with other nonlinear optimization techniques. More significantly, the framework was applied to a highly nonlinear aerobraking trajectory optimization problem involving coupled orbital and atmospheric dynamics. In this setting, standard SCP became trapped in a suboptimal local minimum which missed the

target orbit and required unnecessary intermediate maneuvers. OS-SCP successfully converged to a substantially lower-cost trajectory and achieved the target orbit exactly without any intermediate burns. These results demonstrated the ability of OS-SCP to explore multiple trajectory families simultaneously and identify improved solutions in extremely sensitive and nonconvex optimization landscapes.

Several limitations remain. Theoretical convergence guarantees for OS-SCP applied to nonconvex problems remain limited, especially when incorporating practical modifications such as weighted consensus and clustering. Additionally, the quality of exploration remains dependent on the diversity and placement of initial agent trajectories. This takes the user's intuition to construct meaningful initializations for agents to effectively explore the solution space. For extremely high-dimensional problems, computational scaling with agent count may also become significant.

Future work will focus on improving both the theoretical and practical aspects of the framework. Potential directions include theoretical convergence properties, parallelization of agents' primal subproblem solves, and incorporation of uncertainty-aware formulations. Additional opportunities include extending the method to higher-fidelity six-degree-of-freedom trajectory optimization problems, onboard guidance applications, and robust optimal control under atmospheric and environmental uncertainty.

Overall, this thesis demonstrates that operator splitting provides a promising avenue for improving exploration within sequential convex programming. By combining the strengths of SCP with coordinated multi-agent optimization, OS-SCP offers a practical and flexible framework for solving challenging nonconvex optimization problems that are otherwise difficult to address using conventional local methods alone.

BIBLIOGRAPHY

[1] T.-H. Kim and I. Maruta, "Truss sizing optimization with a diversity-enhanced cyclic neighborhood network topology particle swarm optimizer," *Mathematics*, vol. 8, no. 7, p. 1087, 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/7/1087

[2] P. Elango, D. Luo, A. G. Kamath, S. Uzun, T. Kim, and B. Açıkmeşe, "Successive convexification for trajectory optimization with continuous-time constraint satisfaction," *arXiv preprint arXiv:2404.16826*, 2024.

[3] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. Routledge, 2018.

[4] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 2002.

[7] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[8] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.

[9] R. Martí, M. G. Resende, and C. C. Ribeiro, "Multi-start methods for combinatorial optimization," *European Journal of Operational Research*, vol. 226, no. 1, p. 1–8, Apr 2013.

[10] M. G. Resende and C. C. Ribeiro, *Optimization by GRASP*. Springer, 2016.

[11] J. de Nobel, D. Vermetten, A. V. Kononova, O. M. Shir, and T. Bäck, "Avoiding redundant restarts in multimodal global optimization," in *Parallel Problem Solving from Nature – PPSN XVIII*. Cham: Springer Nature Switzerland, 2024, pp. 268–283.

[12] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," *2011 IEEE ICRA*, p. 4569–4574, May 2011.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] J. Kang, A. U. Raghunathan, and S. Di Cairano, "Decomposition via ADMM for scenario-based model predictive control," in *2015 IEEE ACC*. IEEE, 2015, pp. 1246–1251.

[15] F. Rey, P. Hokayem, and J. Lygeros, "Ask not what ADMM can do for you, ask what you can do for ADMM—virtual subsystems in MPC," in *2017 IEEE CDC*. IEEE, 2017, pp. 4357–4362.

[16] F. Rey, P. Hokayem, and J. Lygeros, "ADMM for exploiting structure in MPC problems," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2076–2086, 2020.

[17] V. Sindhwani, R. Roelofs, and M. Kalakrishnan, "Sequential operator splitting for constrained nonlinear optimal control," in *2017 IEEE ACC*, 2017, pp. 4864–4871.

[18] D. Drusvyatskiy and A. S. Lewis, "Error bounds, quadratic growth, and linear convergence of proximal methods," *Mathematics of operations research*, vol. 43, no. 3, pp. 919–948, 2018.

[19] M. Szmuk, T. P. Reynolds, and B. Açıkmeşe, "Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1399–1413, 2020.

[20] T. Reynolds, D. Malyuta, M. Mesbahi, B. Acikmese, and J. M. Carson, "A real-time algorithm for non-convex powered descent guidance," in *AIAA Scitech 2020 Forum*, 2020, p. 0844.

[21] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[22] J. Ganiban, N. Pavlasek, and B. Acikmese, "A sequential operator-splitting framework for exploration of nonconvex trajectory optimization solution spaces," 2025. [Online]. Available: https://arxiv.org/abs/2511.14752

[23] P. Elango, D. Luo, A. G. Kamath, S. Uzun, T. Kim, and B. Açıkmeşe, "Continuous-time successive convexification for constrained trajectory optimization," *Automatica*, vol. 180, p. 112464, 2025.

[24] G. M. Chari and B. Açıkmeşe, "QOCO: a quadratic objective conic optimizer with custom solver generation," *Mathematical Programming Computation*, Mar. 2026. [Online]. Available: http://dx.doi.org/10.1007/s12532-026-00311-8

[25] A. P. Girija, "Aerocapture: A historical review and bibliometric data analysis from 1980 to 2023," *Journal of Aeronautics, Astronautics and Aviation*, 2023, preprint available on arXiv.

[26] N. X. Vinh, W. R. Wyatt, and J. M. Longuski, "Mars aerocapture using bank modulation," in *AIAA/AAS Astrodynamics Specialist Conference*. Denver, Colorado: American Institute of Aeronautics and Astronautics, 2000.

[27] NASA Langley Research Center, "Aerocapture for discovery missions," https://discovery.larc.nasa.gov/PDF$_F$$ILES$/$Aerocapture_std_briefing_June10_v2.pdf$, 2010.

[28] A. M. Korzun, S. Dutta, R. D. McDaniel, C. D. Karlgaard, and J. A. Tynis, "Aerodynamics for the adept sr-1 flight experiment," in *AIAA Scitech 2020 Forum*, 2020.

[29] S. Mceowen, D. J. Calderone, A. Tiwary, J. S. K. Zhou, T. Kim, P. Elango, and B. Açıkmeşe, "Autotuned primal–dual successive convexification for reentry guidance," *Journal of Guidance, Control, and Dynamics*, vol. 48, no. 9, pp. 2020–2040, 2025.

[30] A. Mittal, J. P. Ganiban, F. Spada, S. Mceowen, C. J. Morales, and B. Açıkmeşe, "Real-time multi-pass aerobrake optimization via successive convexification," 2027, submitted for publication.

[31] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Computers & Structures*, vol. 82, no. 9–10, pp. 781–798, 2004.

[32] L. J. Li, Z. B. Huang, F. Liu, and Q. H. Wu, "A heuristic particle swarm optimizer for optimization of pin connected structures," *Computers & Structures*, vol. 85, no. 7–8, pp. 340–349, 2007.

[33] R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Computers & Structures*, vol. 85, no. 19–20, pp. 1579–1588, 2007.

[34] A. Kaveh and S. Talatahari, "Size optimization of space trusses using big–bang big–crunch algorithm," *Computers & Structures*, vol. 87, no. 17–18, pp. 1129–1140, 2009.

[35] L. Lamberti and C. Pappalettere, "An improved harmony-search algorithm for truss structure optimization," in *Proceedings of the Twelfth International Conference on Civil, Structural and Environmental Engineering Computing*. Stirlingshire, UK: Civil-Comp Press, 2009.

[36] S. O. Degertekin, "Improved harmony search algorithms for sizing optimization of truss structures," *Computers & Structures*, vol. 92–93, pp. 229–241, 2012.

[37] M. Sonmez, "Artificial bee colony algorithm for optimization of truss structures," *Applied Soft Computing*, vol. 11, no. 2, pp. 2406–2418, 2011.

[38] S. O. Degertekin and M. S. Hayalioglu, "Sizing truss structures using teaching-learning-based optimization," *Computers & Structures*, vol. 119, pp. 177–188, 2013.

[39] C. V. Camp, "Design of space trusses using big bang–big crunch optimization," *Journal of Structural Engineering*, vol. 133, no. 7, pp. 999–1008, 2007.