Semantic Constraint Synthesis for Adaptive Trajectory Optimization via Large Language Models

Anonymous CVPR submission

Paper ID *****

Abstract

Trajectory optimization is a critical component for enabling safe and reliable autonomous operations in space exploration. As space missions increase in frequency, complexity, and scope, there is a growing need to rapidly formulate mathematically sound trajectory optimization problems that accurately reflect mission objectives and operational constraints. However, translating mission intent into tractable analytical formulations for trajectory optimization requires substantial domain expertise. This paper presents a framework that leverages large language models (LLMs) to translate natural language descriptions of mission requirements and constraints into executable trajectory optimization code and corresponding mathematical formulations. Experiments in spacecraft rendezvous scenarios demonstrate a high success rate in reconditioning a convex trajectory optimization problem from semantic mission requirements. Ultimately, this work highlights the potential of LLMs to bridge high-level intent and formal optimization models, enabling more flexible and efficient trajectory design of spacecraft.

1. Introduction

The formulation of optimal control problems (OCPs) plays a critical role in spacecraft trajectory generation, enabling both high-level mission planning and low-level maneuver execution. Although numerous case-specific solution methods have been developed by exploiting particular vehicle dynamics [3, 7], there is an increasing demand for general-purpose solvers capable of handling constrained trajectory optimization in complex scenarios such as rocket landing and space rendezvous. Convex optimization, in particular, provides a powerful framework due to its strong theoretical foundations [1, 16] and the availability of efficient embedded solvers [6], making it well-suited for real-time and safety-critical applications. However, its practical implementation and iterative design are often constrained by the need for substantial domain expertise and labor-intensive problem reformulation. Moreover, in operational settings where environmental conditions or mission objectives may change, reliance on a fixed, parameterized OCP significantly limits system adaptability.

In parallel, recent advances in large language models (LLMs), capable of scenario interpretation, mathematical reasoning, and code generation, have enabled the translation of natural-language specifications into formal, solver-verifiable constraints and symbolic representations [17, 18]. Despite the existing theoretical and algorithmic support for embedding complex constraints within a convex-based OCP framework[15, 16, 19], current spacecraft guidance methodologies have not yet capitalized on these advances. Leveraging these algorithms would provide a principled and systematically extensible foundation for language-driven problem formulation.

This paper proposes an LLM-based semantic interface that compiles natural-language mission requirements into solver-ready trajectory constraints for OCPs. Given a baseline problem specification and a newly documented mission requirement, the proposed pipeline generates both a formal mathematical formulation of the corresponding constraints and executable Python code that augments the underlying OCP. We situate this contribution within the broader context of aerospace vehicle guidance, which traditionally demands substantial expertise in vehicle dynamics and optimal control. Incorporating language-driven problem formulation offers practical advantages by enabling more intuitive specification of guidance objectives while streamlining iterative design and refinement.

The contributions of this work are threefold:

- We introduce a semantic interface that translates natural-language mission requirements into formal mathematical constraints and executable code compatible with numerical optimal control solvers using LLMs
- We conduct initial ablation studies to analyze the impact of different LLM prompting and generation strategies, providing insight into the contributions of individual components within the semantic compilation pipeline.

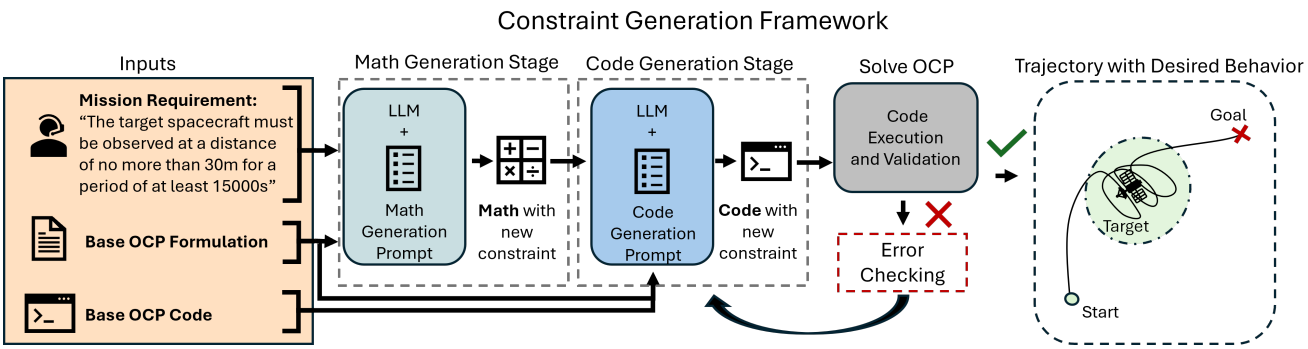


Figure 1. **Automated constraint synthesis from natural-language mission requirements.** A structured input consisting of (i) the baseline LaTeX formulation of the OCP, (ii) the corresponding baseline code, and (iii) a text-based mission requirement is provided as input to the proposed framework to initialize a structured generation process. First, a pretrained LLM translates the requirement into a formal mathematical expression that augments the original problem formulation. Next, a subsequent query generates the corresponding executable code snippet. Finally, the updated code is solved by the numerical OCP solver, producing an adaptive and safety-compliant spacecraft trajectory.

- We validate the proposed framework in a preliminary spacecraft rendezvous case study, demonstrating its ability to reformulate and augment a convex optimal control problem based on semantic constraint descriptions.

2. Related Work

Semantics-based Adaptive Trajectory Generation: The majority of learning-based spacecraft guidance methods target efficient solutions of parameterized OCPs [10, 11, 20]. Recent efforts toward semantic conditioning include LLM-based control generation [9, 12] and transformer models that map text to trajectories with safety enforced via SCP projection [21]. Yet these approaches remain tied to predefined trajectory parameterizations and waypoints, limiting their ability to encode complex mission requirements. Here, we instead synthesize formal mathematical constraints from text and integrate them directly into the OCP formulation.

LLMs for Optimization Code Generation: Recent works in optimization and operations research show that LLMs are effective for extracting a structured constraint formulation from natural-language descriptions [18]. Specifically, natural-language interfaces for translation from natural language to temporal logics [2, 23] or automated nonconvex-to-convex transformation [17] are explored, with particular applications in LLM-driven safe robotic control [8, 24]. However, applications of such LLM-based constraint generation to vehicular motion planning remain limited [22, 23], and are particularly scarce in space vehicle guidance in relation to OCPs. We address this research gap through the use of LLM–based automated constraint generation for spacecraft autonomy applications.

3. Methodology

The goal of this paper is to translate natural-language mission requirements into a formal mathematical expression and then into an executable code snippet for a numerical solver, specifically a convex optimization-based algorithm. Formally, for a discrete-time OCP with fixed final time expressed as:

$$\min_{\{x_k,u_k\}_{k=1}^N} \quad \sum_{k=1}^{N} \mathcal{J}_k \tag{1a}$$

$$\text{subject to} \quad x_{k+1} = f_k(x_k, u_k), \quad \forall k \in \mathbb{Z}_{1:N-1}, \tag{1b}$$

$$x_1 = x_0, \quad x_N = x_f, \tag{1c}$$

$$\boldsymbol{g}(x_k, u_k) \leq 0, \qquad \forall k \in \mathbb{Z}_{1:N-1}. \tag{1d}$$

$$\boldsymbol{h}(x_k, u_k) = 0, \qquad \forall k \in \mathbb{Z}_{1:N-1}, \tag{1e}$$

we wish to reconstruct $\boldsymbol{g}(x, u)$ and $\boldsymbol{h}(x, u)$ based on the documented mission requirements, and integrate them into an executable script. To this end, the proposed automated architecture is structured as a three-stage process consisting of (i) mathematical expression generation, (ii) executable code generation, and (iii) execution of a convex-based numerical OCP solver with error handling. This intermediary step of mathematical expression generation is intended to facilitate the generation of complex requirements that require systematic decomposition with intermediate representations [13, 14].

The remainder of the section introduces the structured input design to the framework (Sec. 3.1) and the internal two-staged structure (Sec. 3.2), respectively.

3.1. Structured input design

Providing a structured representation of the baseline OCP formulation, together with explicit documentation of the mission requirements, is essential to enable coherent reasoning by the LLM and to support reliable code generation. Accordingly, the proposed framework ingests three inputs:

- **Base OCP code:** a Python file with the base OCP problem in code based on CVXPY library [5], with designated placeholders indicating where additional constraints or modifications can be inserted.
- **Base OCP formulation:** a LaTeX-based description of the baseline OCP that explains the general context of the problem (e.g., variable definition and base constraints).
- **Mission Requirement:** a text describing a feasible Scenario adjustment, trajectory requirement, or path constraint.

3.2. Staged constraint synthesis

Given the structured inputs, a pre-trained LLM is queried twice with an incremental instruction towards the full constraint generation. Specifically, the following process is taken within the framework.

- **Mathematical Reasoning:** Instructs the LLM to augment the provided base OCP formulation in LaTeX-format with the mathematical expression translated from the mission requirements.
- **Code Generation:** Instructs LLM to read the added constraint from the generated LaTeX-file, convert to an executable code snippet, and insert into the base OCP code at an appropriate location.

Once the executable code is generated, the numerical OCP solver is called to perform trajectory optimization, yielding either a feasible trajectory or an error report.

4. Experiments

A spacecraft proximity operations scenario, in which a chaser spacecraft executes a complex maneuver around a chief spacecraft, is formulated as a baseline convex OCP for the case study. The (relative) state representation is in Relative Orbital Elements (ROE) [4] scaled by the semi-major axis of the target orbit: $x_{\text{qns}} = a[\delta a, \delta\lambda, \delta e_x, \delta e_y, \delta i_x, \delta i_y]$. Across all test cases, the initial and terminal states are fixed to $x_0 = [0, -100, 0, 0, 0, 0]$ m and $x_f = [0, 100, 0, 0, 0, 0]$ m with the chief located at a Low Earth Orbit, with a time horizon of $t_f = 7.645$ hours with $N = 200$ timesteps. This corresponds to a relative reconfiguration in which the chaser initially shares the chief's orbit with a negative along-track separation (trailing) and is required to transition to a positive along-track separation (leading).

To benchmark the performance of the proposed pipeline, three other alternative frameworks are considered:

A) **Single-shot, No base OCP formulation**: The pretrained model performs one-shot executable code generation using only the base OCP code and the text-based mission requirement. The base OCP formulation document is not provided.

B) **Single-shot, Full inputs**: All structured inputs (base OCP formulation, base OCP code, and mission requirement) are provided to the pretrained model for one-shot executable code generation.

C) **Two-staged, No base OCP formulation in the second stage**: The framework employs two-stage constraint synthesis via a mathematical intermediate representation. The first model (constraint translation) has access to the base OCP formulation. However, the second model (code generation) receives only (i) the base OCP code and (ii) the generated mathematical constraints.

D) **Two-staged, Full inputs (proposed)**: The framework employs two-stage constraint synthesis via a mathematical intermediate representation. In the second stage, the code generation model is provided with (i) the base OCP code, (ii) the generated mathematical constraints, and (iii) the original base OCP formulation.

Table 1 summarizes the documented mission requirements that are used as a part of the input, the ground truth mathematical representation, and example implementation in CVXPY [5]. All three requirements can be formulated as convex constraints, requiring no additional information outside what is given in the inputs in order to implement.

4.1. Tests and results

The experiment comprises 100 trials for each fixed input and mission requirement, evaluated across all four frameworks. In this experiment, GPT-5.2 was adopted as a code generation model, and GPT-4.1-mini was used as a mathematical reasoning model. A temperature of 0.0 was used for all experiments in order to maintain predictability across output structures and to facilitate correct code syntax.

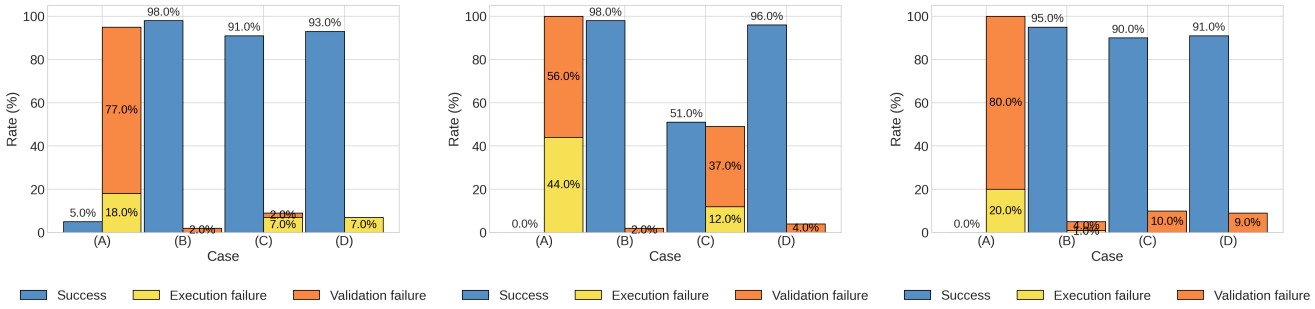
The success rates for each prompt are summarized in Table 2. In this paper, *success* of a trial is defined as a case that generates (i) executable code and (ii) a trajectory that satisfies the target behavior specified in the mission requirements. Any instance in which the generated code failed to execute is classified as a failure. Furthermore, the resulting trajectory behavior must behave as expected within a margin. Fig. 2 summarizes the types of failures encountered with the two categories of failures documented; code execution failures and validation failures.

From these results, the importance of a structured input that provides the problem context is clear. In cases where there is no initial base OCP formulation, the model almost always fails to get the desired trajectory behavior, due to a mix of code-based failures and validation failures (cf. Fig. 2). These results underscore the importance of providing a structured input that provides sufficient problem context. In contrast, for scenarios where the OCP formulation is given, the success rate of the single-shot code generation pipeline is markedly improved, with success rates exceeding 95% across all tested prompts.

When comparing the two-stage frameworks and the effect of including the full inputs at each stage, the success rate was approximately maintained for Prompt 1 and

Prompt	Mission Requirements	Mathematical Constraints	Example Code
1	The target spacecraft must be observed at a distance of no more than 30m for a period of at least 15000s in the middle of the trajectory.	$\|(\Psi_k x_k)_{1:3}\|_2 \leq 30, \quad T_{\text{start}} \leq k \leq T_{\text{end}}$	`[cp.norm(Psi[k][0:3]@x[k]) <= 30 for k in range(t_start, t_end)]`
2	At a point in the middle of the trajectory, the relative RTN position between the two spacecraft must be [20, 0, 0] m.	$(\Psi_k x_k)_{1:3} = [20, 0, 0] \quad k = T_{\text{mid}}$	`(Psi[T_mid]@x[T_mid])[0:3] == cp.Constant([20,0,0])`
3	The total Delta-V used in the first half of the trajectory must be no more than 2 mm/s. The spacecraft must remain behind the target spacecraft in the orbit for the first 75% of the trajectory.	$\|a\|_2 \leq 2, \quad 0 \leq k \leq T_{\text{mid}}$ $(\Psi_k x_k)_1 \leq 0, \quad 0 \leq k \leq \frac{3}{4} T_{\text{final}}$	`[cp.norm(a[k]) <= 2 for k in range(0, T_mid)]` `[Psi[T_mid]@x[T_mid][1] <= 0 for k in range(0, int(.75*T_final))]`

Table 1. Mission requirements used for semantic constraint synthesis. `cp` refers to a CVXPY library [5].



(a) Prompt 1 (b) Prompt 2 (c) Prompt 3

Figure 2. Distributions of failure modes in the three prompt inputs across four frameworks: (A) Single-Shot, No OCP Formulation, (B) Single-Shot, Full Inputs, (C) Two-staged, No base OCP formulation in the second stage, and (D) Two-staged, Full inputs.

	Prompt		
	1	2	3
(A) Single-shot, No base OCP formulation (%)	5.0	0.0	0.0
(B) Single-shot, Full inputs (%)	98.0	98.0	95.0
(C) Two-staged, No base OCP formulation in the second stage (%)	91.0	51.0	90.0
(D) Two-staged, Full inputs (%)	93.0	96.0	91.0

Table 2. Success statistics across three prompts (100 trials each).

3 while Prompt 3 suffered a significant drop. This suggests that the OCP formulation provides essential contextual information, and passing it to the second stage helps prevent failures caused by subtle but crucial missing details that may be omitted from the generated math formulation.

While all frameworks that incorporate a base OCP formulation at each stage achieve success rates exceeding 90% across the evaluated prompts, the single-shot approach with full inputs demonstrates the strongest performance under the selected mission requirements. This result may be attributed to the relatively simple nature of the tested requirements, all of which have direct convex implementations. If this interpretation is correct, the proposed Two-Stage method—despite maintaining a high success rate—offers limited advantage over a more direct formulation strategy in these cases. However, as requirement complexity increases and the prompt-to-code pathway requires additional intermediate reasoning steps, we expect the performance of the direct single-shot method to degrade. Given that the long-term objective of this framework is to address such complex requirements and generate their corresponding mathematical formulations, the Two-Stage architecture remains the more promising approach for scalable and flexible automated constraint generation.

5. Conclusion and Future Work

This paper proposes an automated framework that translates semantic mission requirements into well-defined mathematical constraints and executable code for trajectory optimization problems. The results indicate that consistent generation of executable code from semantic mission requirements is achievable with appropriate input structuring and prompt design. In the evaluated prompts, an intermediate mathematical formulation step was not strictly necessary; however, incorporating this step did not significantly degrade performance. Future work will extend the framework to (i) an automatic convexification with formal verification to accommodate nonconvex constraints, (ii) ablation studies across different pretrained LLMs, and (iii) supervised fine-tuning of the models and associated scalable dataset curation tailored for trajectory optimization.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 1

[2] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. Nl2tl: Transforming natural languages to temporal logics using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15880–15903, 2023. 2

[3] M. Chernick. *Optimal Impulsive Control of Spacecraft Relative Motion*. PhD thesis, Stanford University, 2021. 1

[4] S. D'Amico. *Autonomous Formation Flying in Low Earth Orbit*. PhD Thesis, Delft University, 2010. 3

[5] Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016. 3, 4

[6] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European control conference (ECC)*, pages 3071–3076. IEEE, 2013. 1

[7] Christopher D'Souza and Christopher D'Souza. An optimal guidance law for planetary landing. In *Guidance, Navigation, and Control Conference*, page 3709, 1997. 1

[8] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, 2025. 2

[9] Matthew Foutter, Daniele Gammelli, Justin Kruger, Ethan Foss, Praneet Bhoj, Tommaso Guffanti, Simone D'Amico, and Marco Pavone. Space-llava: A vision-language model adapted to extraterrestrial applications. In *2025 IEEE Aerospace Conference*, pages 1–23, 2025. 2

[10] Tommaso Guffanti, Daniele Gammelli, Simone D'Amico, and Marco Pavone. Transformers for trajectory optimization with application to spacecraft rendezvous. In *IEEE Aerospace Conference*, pages 1–13, 2024. 2

[11] Kirk Hovell and Steve Ulrich. Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets*, 58(2):254–264, 2021. 2

[12] Amit Jain and Richard Linares. Autonomous reasoning for spacecraft control: A large language model framework with group relative policy optimization. *arXiv preprint arXiv:2601.04334*, 2026. 2

[13] Zhongyuan Lyu, Shuoyu Hu, Lujie Liu, Hongxia Yang, and Ming LI. Canonical intermediate representation for llm-based optimization problem formulation and code generation. *arXiv preprint arXiv:2602.02029*, 2026. 2

[14] Zhi Ma, Cheng Wen, Zhexin Su, Xiao Liang, Cong Tian, Shengchao Qin, and Mengfei Yang. Bridging natural language and formal specification–automated translation of software requirements to ltl via hierarchical semantics decomposition using llms. *arXiv preprint arXiv:2512.17334*, 2025. 2

[15] Danylo Malyuta and Behçet Açıkmeşe. Fast homotopy for spacecraft rendezvous trajectory optimization with discrete logic. *Journal of Guidance, Control, and Dynamics*, 46(7): 1262–1279, 2023. 1

[16] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe. Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently. *IEEE Control Systems Magazine*, 42(5):40–113, 2022. 1

[17] Xinyue Peng, Yanming Liu, Yihan Cang, Yuwei Zhang, Xinyi Wang, Songhang Deng, and Jiannan Cao. Nc2c: Automated convexification of generic non-convex optimization problems. *arXiv preprint arXiv:2601.04789*, 2026. 1, 2

[18] Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, et al. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In *NeurIPS 2022 competition track*, pages 189–203. PMLR, 2023. 1, 2

[19] Michael Szmuk, Taylor P Reynolds, and Behçet Açıkmeşe. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(8):1399–1413, 2020. 1

[20] Yuji Takubo, Tommaso Guffanti, Daniele Gammelli, Marco Pavone, and Simone D'Amico. Towards robust spacecraft trajectory optimization via transformers. In *IEEE Aerospace Conference*, 2025. 2

[21] Yuji Takubo, Arpit Dwivedi, Sukeerth Ramkumar, Luis A Pabon, Daniele Gammelli, Marco Pavone, and Simone D'Amico. Semantic trajectory generation for goal-oriented spacecraft rendezvous. In *AIAA SCITECH 2026 Forum*, 2026. 2

[22] Teun van de Laar, Zengjie Zhang, Shuhao Qi, Sofie Haesaert, and Zhiyong Sun. Vernacopter: Disambiguated natural-language-driven robot via formal specifications. *arXiv preprint arXiv:2409.09536*, 2024. 2

[23] Yujin Wang, Zhaoyan Huang, Shiying Dong, Hongqing Chu, Xiang Yin, and Bingzhao Gao. Chatstl: A framework of translation from natural language to signal temporal logic specifications for autonomous vehicle navigation out of blocked scenarios. In *2024 16th International Conference on Computer and Automation Engineering (ICCAE)*, pages 483–487. IEEE, 2024. 2

[24] Ziyi Yang, Shreyas S Raman, Ankit Shah, and Stefanie Tellex. Plug in the safety chip: Enforcing constraints for llm-driven robot agents. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14435–14442. IEEE, 2024. 2

Semantic Constraint Synthesis for Adaptive Trajectory Optimization via Large Language Models

Supplementary Material

6. Generation Prompts

LaTeX Generation Prompt Example

```
You are an expert in optimal control problem
     formulation and convex optimization using cvxpy.

You will take in 2 files:

1. Problem Spec: LATEX file that lays out a given
     optimal control problem
2. Problem Prompt: A description of a change to the
     given control problem in that will result in a
     new constraint or set of constraints for the
     problem

Your job is to output a new Latex document with a
     section for each new constraint that includes:
-The minimal mathematical formulation of the
     constraint in terms of the variables and
     parameters defined in the original problem spec.
- A single line mathematical equation for the
     constraint in terms of the variables and
     parameters defined in the original problem spec
     and any values that you have also defined in
     the mathematical formulation of the constraint.

Rules for the constraint addition section:
- Do not use any explanatory text, only include the
     mathematical formulation and the single line
     mathematical equation.
-  Ensure the Dimensions of all variables is
     explicitly defined in the mathematical
     formulation of the constraint.
- This description will be used to write code so
     ensure that the mathematical formulation is
     clear and unambiguous.


Output nothing other than the LATEX document with
     just the constraints. Do not include any
     explanations or commentary.
```

Code Generation Prompt Example

```
You are an expert in optimal control problem
     formulation and convex optimization using cvxpy.
You are editing an existing Python file.
Based on the provided constraint addition latex file
     and the latex spec file you will add the
     constraints in the constraint addition latex
     file.

Rules:
- You will keep all existing code.
- ONLY replace code between:
  # BEGIN AUTO-GENERATED CONSTRAINTS
  and
  # END AUTO-GENERATED CONSTRAINTS
- Do NOT change anything else.
- Return the FULL updated file.
- No markdown. No explanation.



Generate constraint code that:
- Only uses the information explicitly provided in
     the provided specification and does not require
     further user input to define the constraint.
- Ensures that the constraint is satisfied when
     discretized.
- Is mindful of dimensionality and ensures all
     variables are defined with explicit dimensions
     in the code.
- Does not overly complicate the constraint and uses
     the simplest form that satisfies the
     requirement of the constraint.
- Uses cvxpy syntax
- Is DCP compliant
```

7. Example Output

7.1. Generated LaTeX Output for Prompt 1

Observation Distance Constraint

Let $d_{\max} = 30$ m be the maximum allowable relative position norm for observation.

Let $T_{\text{obs}} = 15000$ s be the required observation duration.

Let Δt be the time step duration in seconds.

Define the number of discrete time steps for observation as

$$M = \left\lceil \frac{T_{\text{obs}}}{\Delta t} \right\rceil, \quad M \in \mathbb{N}.$$

Let $t_{\text{start}} = \left\lfloor \frac{N-M}{2} \right\rfloor$ be the starting time index of the observation window.

For all $t = t_{\text{start}}, t_{\text{start}} + 1, \ldots, t_{\text{start}} + M - 1$, the relative position vector in RTN frame is

$$\mathbf{p}_t = \left(\mathbf{x}_t^{\text{RTN}}\right)_{1:3} = \left(\Psi_t \mathbf{s}_t\right)_{1:3} \in \mathbb{R}^3.$$

The constraint is

$$\|\mathbf{p}_t\|_2 \leq d_{\max}, \quad \forall t = t_{\text{start}}, \ldots, t_{\text{start}} + M - 1.$$

Single line constraint:

$$\| \left(\Psi_t \mathbf{s}_t\right)_{1:3} \|_2 \leq 30$$

$$t = \left\lfloor \frac{N - \lceil T_{obs}/\Delta t \rceil}{2} \right\rfloor, \ldots, \left\lfloor \frac{N - \lceil T_{obs}/\Delta t \rceil}{2} \right\rfloor$$

$$+ \lceil T_{obs}/\Delta t \rceil - 1$$

7.2. Generated Code Output Snippet For Prompt 1

```
#BEGIN AUTO-GENERATED CONSTRAINTS####################

# Observation Distance Constraint:
# For an observation window of duration T_obs=15000s,
     enforce ||p_t||_2 <= d_max=30m
# where p_t = (Psi_t s_t)_{1:3}.
d_max = 30.0
T_obs = 15000.0

# Conservative discretization: ensure at least one step
     if dt is large; cap to horizon length.
M = int(np.ceil(T_obs / float(self.dt))) if self.dt is
     not None else self.n_time
M = max(1, M)
M = min(M, self.n_time)

t_start = int(np.floor((self.n_time - M) / 2.0))
t_start = max(0, min(t_start, self.n_time - M))

for t in range(t_start, t_start + M):
    # psi[t] is expected shape (6,6); s[t] is (6,)
    p_t = (psi[t] @ s[t])[0:3]
    con += [cp.norm(p_t, 2) <= d_max]
#END AUTO-GENERATED CONSTRAINTS####################
```