Back to Basics for CAE: Demystifying Input File for and with Generative AI

Dr. Anthony Favaloro,
Hexagon Manufacturing Intelligence, USA,
anthony.favaloro@hexagon.com

Brody Kendall
Northwestern University, USA

Daniel Paff
Hexagon Manufacturing Intelligence, Germany

Subham Sett
Hexagon Manufacturing Intelligence, USA

Abstract

Finite element analysis (FEA) has been a pillar of computer-aided engineering (CAE) since the 1960s. The longevity of this simulation technology presents both opportunities as well as unique challenges for leveraging generative AI for both new and experienced users. While there are mushrooming large language model (LLM) based assistants available for such users, many of the assistants operate at a superficial level based on training documents and manuals; they are not sufficient for supporting a user intent on understanding, debugging and quickly resolving issues that lie at the input file level, the gateway to the FEA solver. In this context, there are three unique challenges to leveraging generative AI. First, the inputs to FEA solvers are text-based with syntaxes, definitions and descriptions exposed through keywords that follow a non-intuitive, unique taxonomy and can be documented in manuals that can have thousands of pages. Next, these input file formats never adapted beyond the original implementation intended for punch cards which allow input to be unsorted but create the burden of ID management. This approach is in direct conflict with modern input decks that are geared towards HPC and can easily be gigabytes in file size. The sorting of such a file could be specific to a pre-processor, company best practices or simply the historical build-up of a model. The unsorted nature of this directly contracts with the natural flow of human language. In this work, we present a unique approach to solving the problem by establishing a graph representation of the input file. This approach enables input-file specific document retrieval and provides users of every skill-level the ability to obtain prompts and responses at a higher level (pure documentation) and deeper level (input file). Future ongoing work explores optimizing the developed algorithms that reduce token consumptions and help users leverage their compute resources in a more cost-effective manner. In this proof-of-concept work, we combine a custom parser and documentation retrieval to

deliver an in-context Generative AI experience relative to the exact features of an input file a user is interrogating. In ongoing trials, we expect reduced onboarding times, reduced debugging times, and reduced touches of traditional documentation.

1. Introduction

Since the advent of the finite element method and the development of CAE tools, users have been struggling against formatting errors, syntax errors, and incomplete definitions to activate necessary capabilities, or even over-defined segments of models. As finite element capabilities have increased, so too has this challenge. While pre-processors, standardized scripts, custom plugins, and similar all seek to simplify the user experience, it is still often the expert users first step to open the finite element input file in a simple text editor and proceed through their mental map of potential issues. While this process feels quite standard to the expert, it often leaves the junior engineer or new CAE user mystified simply due to the volume of information and methodology learned through countless experiences by the expert user.

In recent years, large language models (LLMs) or even more advanced multi-modal foundation models have shown great success in assisting users in collating and synthesizing large amounts of data via prompts into reasonable summaries, helpful study guides, or even generated podcasts (e.g., NotebookLM by Google). Many products exist built using LLMs to assist with daily tasks such as catching up on emails, generating first drafts of written works, or summarizing meetings via transcripts (e.g., Copilot for Office 365 by Microsoft, Google Workspaces with Gemini, etc. Similarly, products exist to enhance the productivity of programmers with code generation, code review, code commenting, and code explanation (e.g., Github Copilot, Codeium, Gemini Code Assist, Amazon CodeWhisperer, etc). Such tools are being shown to provide increased user productivity and also increased user satisfaction levels [1].

In this proof of concept work, we demonstrate that LLMs can be successfully leveraged to make sense of finite element input files with realized features that are beneficial to onboarding new users, continued and specialized training, or simply enhancing an expert users productivity including overall model summarization, model gap analysis, detailed explanations of specific input file segments, and enhancement of a particular entry or feature. In this way, the language model behaves as a live demonstration guide, capable of providing insight into a user's current model and tasks, rather than generic problems included with CAE tool documentation.

2. Methods

In this proof of concept specifically, MSC Nastran is chosen as the CAE tool, and the input files (bulk data files or BDFs) are considered. When approaching, Nastran-style input files, there are a few noteworthy conflicts with the standard approach of LLMs. LLMs rudimentarily function by predicting the next token in a sequence of tokens, where tokens are simply translations of words or sections of words into a numerical equivalent. In this way, LLMs are functionally conditioned to deal with sequential data such as typical human language. In contrast, Nastran-style input format is a legacy format and was developed in the era of punch cards, and this led to a format that is completely unsorted. So, while an LLM may treat relevance of any token to another token via proximity (e.g, In the text sequence "ABC", B follows A), a Nastran input file tracks relevance through an often user-overwhelming number of cross-referencing IDs (e.g., CTRIA3 2 may be defined between CTRIA3 1 and CTRIA3 3 but reference GRIDs defined elsewhere in the input file). Additionally, the compute cost of LLMs is directly tied to the length of the prompt. So, it is unfeasible to provide an entire finite element model or even all relevant documentation to the LLM through the prompt. Rather, we must develop an economical system that gathers the required context for the desired tasks. Therefore, to generate appropriate prompts for LLMs from a Nastran input file, a custom retrieval augmented generation (RAG) [2] system is implemented which makes use of the user provided BDF file, MSC Nastran Documentation, and a custom MSC Nastran BDF parser.

In the following, we demonstrate the summary task, gap analysis task, entry explanation task, and entry enhancement task as well as layering user directives onto the tasks. Due to the everchanging availability and capability of LLMs, it is noted that the results in this paper are obtained using Claude 3.5 Sonnet in January 2025.

3. Results

As a first example result, we consider the summarization task using the model ha144a.dat from the Aeroelastic Analysis Users Guide and distributed in the MSC Nastran Test Problems Library. This model is described in the manual as "a half span symmetric model of an idealized forward swept wing configuration." [3] The response to the summary prompt from the LLM is shown in Figure 1. Here, although the model contains all the typical required entries in a finite element model: nodes, elements, properties sections, materials, etc. The LLM has successfully highlighted the key items related to the aeroelastic analysis as well as providing a study guide for how one might begin their investigation through the model.

Key Entry Importance

1. **TRIM** (3 instances, 9 references)
   - Central to analysis purpose with multiple subcases
   - Heavily connected to AESTAT entries
2. **AEROS** (1 instance, 2 references)
   - Defines fundamental aeroelastic parameters
   - Connected to coordinate systems
3. **AESURF** (1 instance, 2 references)
   - Defines control surfaces
   - Links aerodynamic and structural models
4. **CAERO1** (2 instances, 4 references)
   - Defines aerodynamic panels
   - Connected to PAERO1 and SPLINE2

Model Summary

This appears to be a static aeroelastic analysis (Solution 144) with these key characteristics:

1. **Structure**
   - Simple beam-based structure (CBAR, PBAR)
   - Rigid connections (RBAR)
   - Mass properties (CONM2)
2. **Aerodynamics**
   - Doublet-Lattice method panels (CAERO1)
   - Aero-structure coupling (SPLINE2)
   - Control surface definition (AESURF)
3. **Analysis**
   - Three trim conditions (3 TRIM entries, 3 subcases)
   - Multiple rigid body states (4 AESTAT)
   - Matrix input present (DMI)

Learning Focus

When working with this model, concentrate on:

1. Understanding the trim conditions in each subcase
2. Checking aero-structure coupling through splines
3. Verifying control surface definitions
4. Reviewing coordinate system setups
5. Understanding constraint conditions (SPC1, SUPORT)

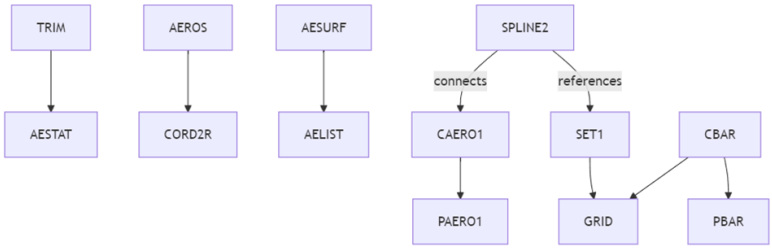


Figure 1 LLM Response to Summary Prompt for ha144a.dat

One immediate, secondary benefit of this approach is that many LLMs have been trained to be multilingual. By simply adding an additional directive to the prompt such as “Respond in Japanese,” responses can be localized, greatly improving accessibility as seen in Figure 2.

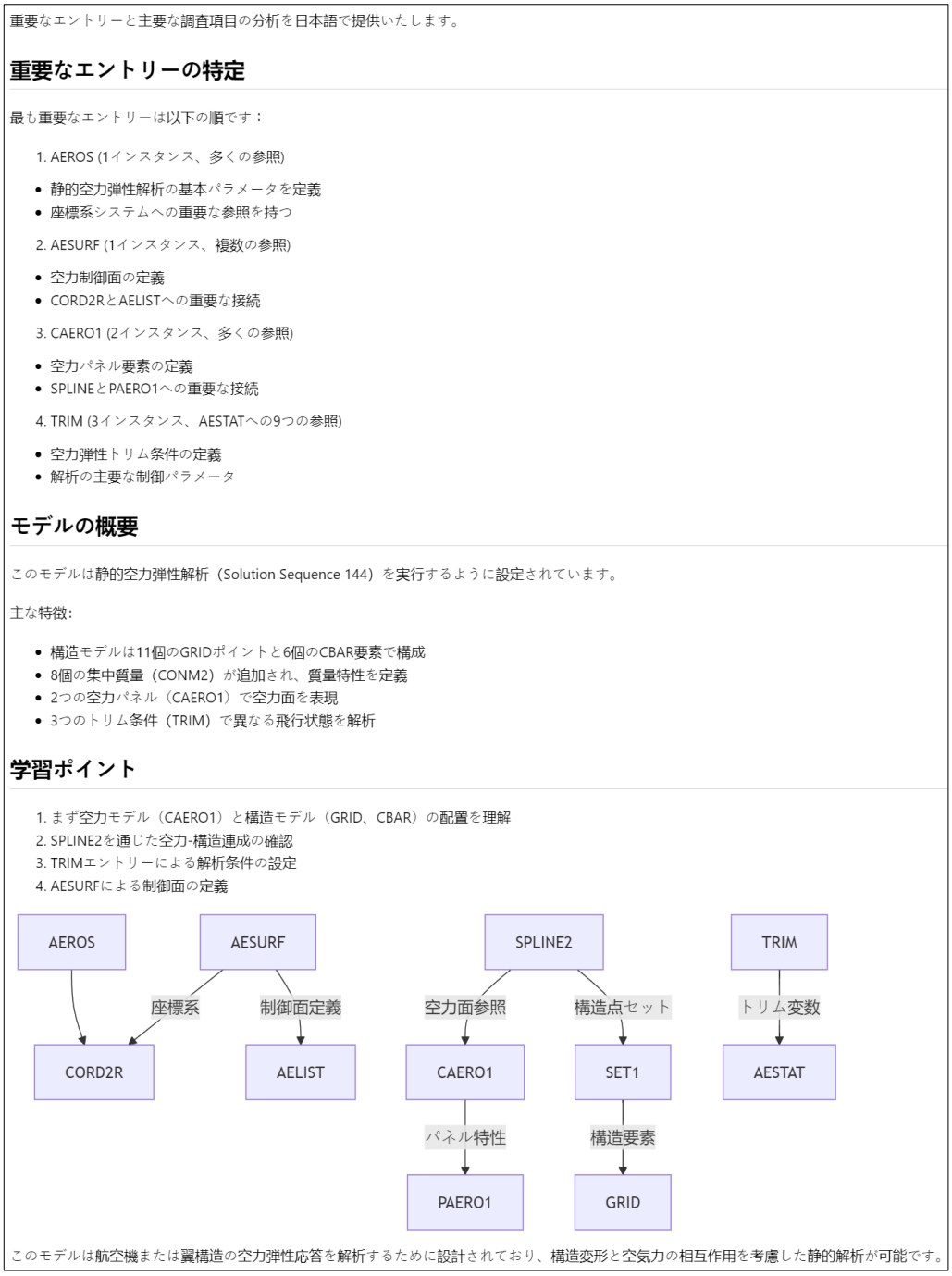
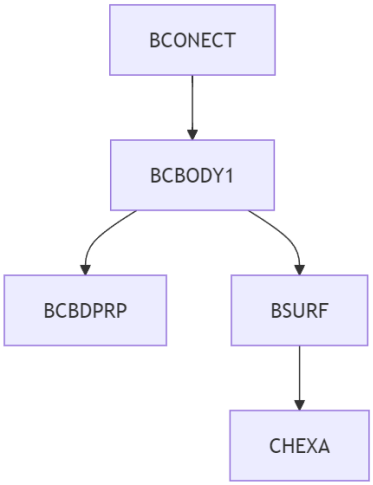


Figure 2 LLM Response to Summary Prompt with “Respond in Japanese” Directive

Key Observations

1. This is a SOL 400 nonlinear analysis with contact bodies and bolts
2. There are 13 BCBODY1 entries defined but no BCONECT entries to create contact pairs
3. All contact body related entries (BCBODY1, BCBDPRP, BSURF) are properly cross-referenced to each other
4. The BOLT1 entries are already referencing CHEXA and GRID entries correctly

Contact Body Hierarchy



Next Steps Priority

1. **Highest Priority**: Create BCONECT entries to define contact pairs between the 13 BCBODY1 entries
2. **Secondary**: Create SET3 entries to group the RBE2 elements if needed for output or post-processing
3. **Optional**: Review if BCMOVE or BCHANGE entries are needed for contact body modifications

Rationale

The most critical gap is the lack of BCONECT entries, which are essential to define how the contact bodies interact with each other. Without these, the contact analysis portion of the solution will not function properly.

The BCPARA entry is already present which will control the contact behavior once the BCONECT entries are created.

The remaining unreferenced entries (PSLDN1, BOLT1, RBE2, PARAM, HDF5OUT, MDLPRM, NLMOPTS) are all standalone configuration entries that don't require explicit referencing to function in the solution.

Figure 3 LLM Response to Gap Analysis Prompt

As a next example, the leaf spring model from Chapter 58 of the Demonstration Problems Manual – Implicit Nonlinear is used [4]. This model is a nonlinear contact model with a mixture of glued and general contact. For this example, we have deleted the BCONTACT case control commands and the BCTABL1, BCONECT, BCONPRG, and BCONPRP bulk data entries leaving the contact setup in a state directly following contact body identification and definition.

We now consider the gap analysis task. For this task, the gap analysis prompt is generated using relevant entries from MSC Nastran that could be added to reference existing bulk data entries. In this way, the LLM is enabled to identify areas for improvement. The initial response to the gap analysis prompt is shown in Figure 3. Here, the LLM has correctly identified that although many entries exist which could be referenced for some goals, the highest priority seems to be to finish the contact setup by specifying the BCONECT entries to pair the defined BCBODY1 entries. Adding the BCONECT cards back to the model, the resulting gap analysis response is shown in Figure 4. Here the LLM has noticed the next step in the contact setup as the primary recommendation, namely, to define BCTABL1 entries and BCONTACT case control commands. Notably, it also notices that there are unreferenced RBE2 entries. Though it would be atypical to directly specify RBEs in case control rather than simply have all active, the LLM response still identifies this as a possible option.

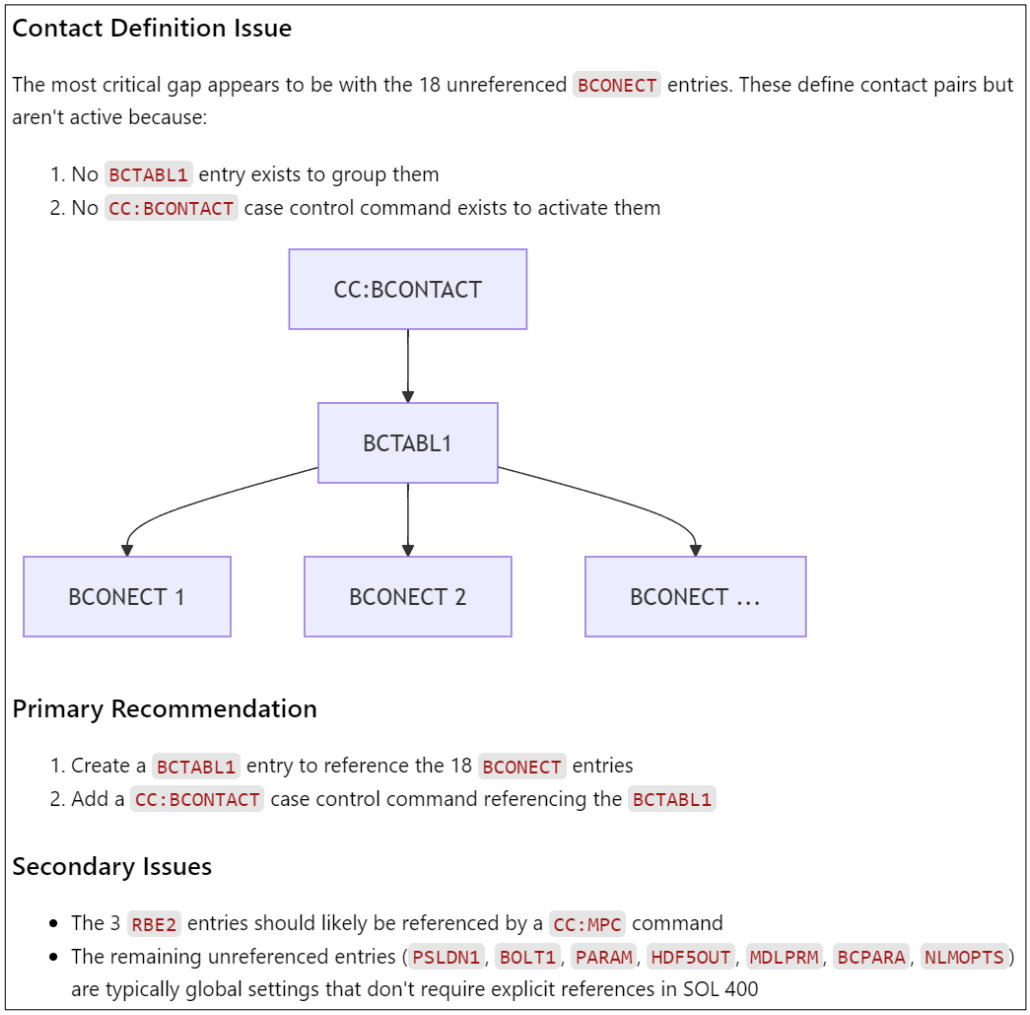


Figure 4 LLM Response to Follow-Up Gap Analysis Prompt

Continuing with the leaf spring example model, we demonstrate entry explanation prompt. Here, we will use one of MSC Nastran's more cumbersome entries present in the model, NLSTEP. The NLSTEP in the leaf spring model is given as:

```
NLSTEP    1
          GENERAL 25       2
          ADAPT   .1                        10
          MECH    UPV      .01      .01
```

The NLSTEP entry ID=1 defines nonlinear solution control parameters and is referenced by the Case Control command CC:NLSTEP=1, indicating this controls the nonlinear solution steps.

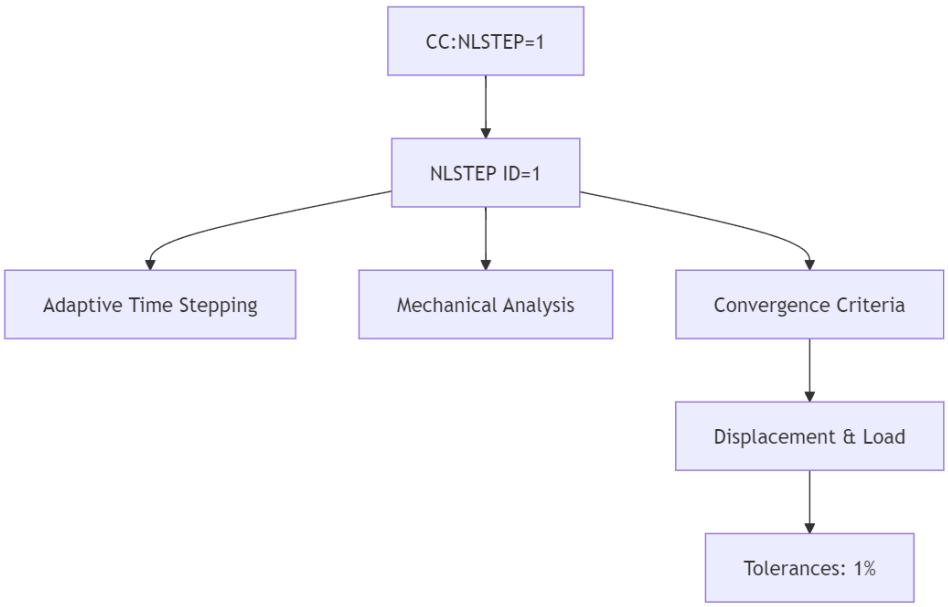
Key parameters specified:

- Uses adaptive time stepping (ADAPT) with initial time step 0.1 (DTINITF=.1)
- Maximum iterations per increment is 25 (MAXITER=25)
- Minimum iterations per increment is 2 (MINITER=2)
- Desired iterations per increment is 10 (NDESIR=10)
- For mechanical analysis (MECH), uses displacement and load convergence criteria (CONV=UPV)
- Convergence tolerances of 1% for both displacement and load (EPSU=.01, EPSP=.01)

In the model hierarchy, NLSTEP sits between:

- Case Control (CC:NLSTEP) which selects which NLSTEP to use
- The solution controls that affect individual nonlinear iterations

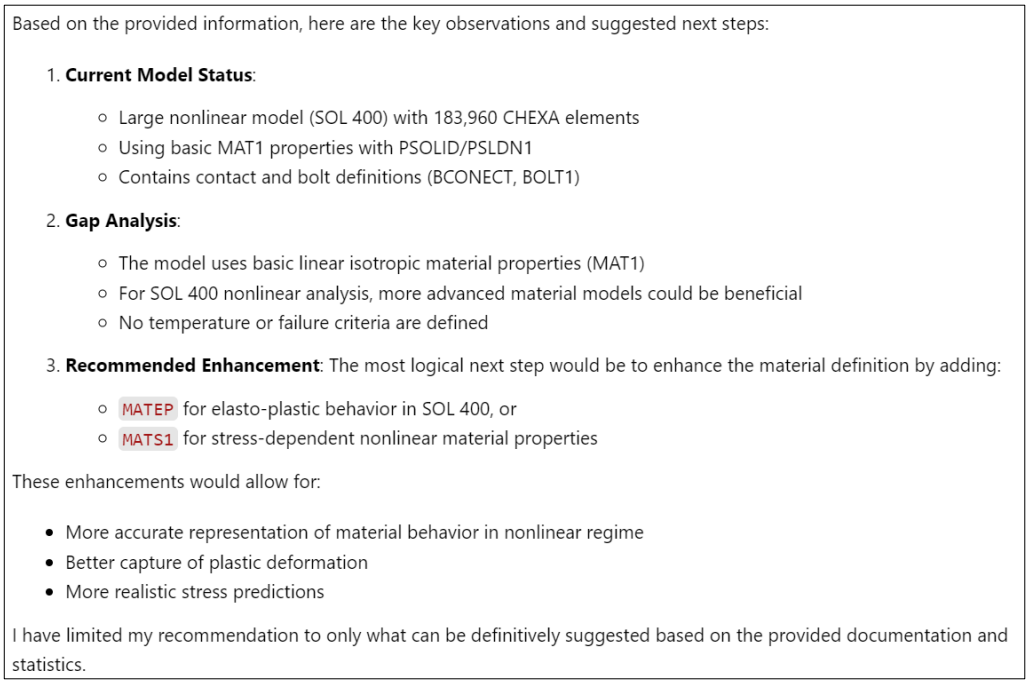
The most important connections can be visualized as:



The NLSTEP entry plays a crucial role in controlling the nonlinear solution process through adaptive time stepping, iteration limits, and convergence criteria. It ensures the solution progresses stably while maintaining accuracy through the specified tolerances.

Figure 5 LLM Response to Entry Specific Prompt for NLSTEP in Leaf Spring Model

Here, the NLSTEP entry may read perfectly well to an expert familiar with it, but to demonstrate the specific entry investigation prompt, the LLM response is shown in Figure 5. Here, without having to investigate the documentation directly to match field to explanations, a human readable description of each field is nearly immediately available. By adding user directives such as "I want to switch to fixed time stepping" additional advice can be obtained in the LLM response.


Based on the provided information, here are the key observations and suggested next steps:

1. **Current Model Status**:
   - Large nonlinear model (SOL 400) with 183,960 CHEXA elements
   - Using basic MAT1 properties with PSOLID/PSLDN1
   - Contains contact and bolt definitions (BCONECT, BOLT1)
2. **Gap Analysis**:
   - The model uses basic linear isotropic material properties (MAT1)
   - For SOL 400 nonlinear analysis, more advanced material models could be beneficial
   - No temperature or failure criteria are defined
3. **Recommended Enhancement**: The most logical next step would be to enhance the material definition by adding:
   - `MATEP` for elasto-plastic behavior in SOL 400, or
   - `MATS1` for stress-dependent nonlinear material properties

These enhancements would allow for:

- More accurate representation of material behavior in nonlinear regime
- Better capture of plastic deformation
- More realistic stress predictions

I have limited my recommendation to only what can be definitively suggested based on the provided documentation and statistics.

Figure 6 LLM Response to Enhancement Prompt

Finally, continuing with the leaf spring example, only simple isotropic material properties have been defined. To demonstrate the enhancement prompt, we choose one of the MAT1 entries and request enhancement response. With no guiding user directive, the LLM response is shown in Figure 6. Here, the LLM has determined that due to the other features of the model, the likely next step would be to define plastic behavior via MATEP or MATS1. However, we can guide the LLM response as well if a different type of enhancement is desired. For example, if we add "I want to model temperature dependency, not plasticity." as a user directive, the LLM response changes to what is shown in Figure 7.

For the prompts and responses shown in this article, the input token count ranges from ~5,000 tokens to ~20,000 tokens, while the output is only about ~1,000 tokens. With the current Claude 3.5 Sonnet pricing of $3/million input tokens and $15/million output tokens, each task costs between $0.02 and $0.09 representing a cost-effective method of performing CAE tool training, onboarding to a specific modeling method or even model on a specific project,

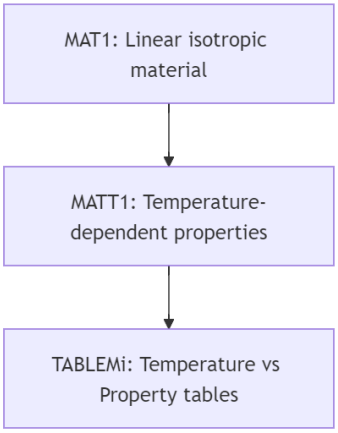
or simply day to day assistance with more cumbersome aspects of finite element input files.

Based on the provided documentation and the overriding directive to model temperature dependency, the clear next step would be to add a MATT1 entry to define temperature-dependent material properties for the existing MAT1 entries.

This is supported by:

1. The documentation explicitly states: "MAT1 materials may be made temperature-dependent by use of the MATT1 entry"
2. The model statistics show:
    - 22 instances of MAT1 entries
    - Currently no MATT1 entries exist
    - This is a SOL 400 nonlinear analysis where temperature effects could be important

The relationship between MAT1 and MATT1 can be shown as:



To implement this:

1. Create TABLEMi entries to define the temperature-property relationships
2. Add MATT1 entries referencing the existing MAT1 entries (MID must match)
3. Reference the TABLEMi entries in the MATT1 entries for the properties that vary with temperature

Note: The documentation also mentions that in SOL 400, TREF on the MAT1 entry would be used for calculating temperature-dependent thermal expansion coefficients, so this should be considered in conjunction with the MATT1 implementation.

Figure 7 LLM Response to Enhancement Prompt with User Directive

4. Conclusion and Outlook

In this work, we have demonstrated the current LLMs are capable, when provided the right context from a CAE system, to provide insightful responses to user tasks. Specifically, we have enabled a summarization task, gap analysis task, entry explanation task, and entry enhancement task. These tasks can also be overloaded with user directives providing more direction by informing the type of modeling being performed or desired to be performed or accessibility by requesting the LLM respond in a certain language or to a certain sophistication level.

It is important to note that there are at least two modes of assistance from LLMs demonstrated in this collection of tasks. One mode requires no user intervention, such as a summarization task or gap analysis task. These tasks can be completed for a variety of training purposes or reporting purposes without specific user queries, much like a summary of a meeting transcript can be automatically produced without a user request. Alternately, as is the case for the entry specific tasks or when layering user directives into the prompt, users can directly engage with the system for a more guided experience with the LLM acting as an advisor.

While future modes of interacting with LLMs are already emerging, to date, they still rely on underlying context systems such as those developed for this proof of concept to enable the LLMs with domain specific knowledge and relevant tasks.

5. References

[1] G. Bakal, A. Dasdan, Y. Katz, M. Kaufman, and G. Levin, “Experience with GitHub Copilot for Developer Productivity at Zoominfo,” arXiv.org, 2025. https://arxiv.org/abs/2501.13282v1 (accessed Jan. 31, 2025).

[2] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Apr. 2021. Available: https://arxiv.org/pdf/2005.11401

[3] *MSC Nastran 2024.1 Aeroelastic Analysis User’s Guide*. 2024.

[4] “Chapter 58: Leaf Spring Analysis using NLPERF,” in *MSC Nastran 2024.1 Demonstration Problems Manual - Implicit Nonlinear* , 2024.