

A systematic assessment of Large Language Models for constructing two-level fractional factorial designs

Alan R. Vazquez ^{*1,†}, Kilian M. Rother², and Marco V. Charles-Gonzalez¹

¹School of Engineering and Sciences, Tecnológico de Monterrey, Mexico

²Faculty of Mechanical Engineering, University of Applied Sciences Kiel, Germany.

[†]Corresponding author. Email: alanrvazquez@tec.mx;

Contributing authors: kilrother@googlemail.com,
Marcocharlesgzz@gmail.com

December 22, 2025

Abstract

Two-level fractional factorial designs permit the study multiple factors using a limited number of runs. Traditionally, these designs are obtained from catalogs available in standard textbooks or statistical software. However, modern Large Language Models (LLMs) can now produce two-level fractional factorial designs, but the quality of these designs has not been previously assessed. In this paper, we perform a systematic evaluation of two popular classes of LLMs, namely GPT and Gemini models, to construct two-level fractional factorial designs with 8, 16, and 32 runs, and 4 to 26 factors. To this end, we use prompting techniques to develop a high-quality set of design construction tasks for the LLMs. We compare the designs obtained by the LLMs with the best-known designs in terms of resolution and minimum aberration criteria. We show that the LLMs can effectively construct optimal 8-, 16-, and 32-run designs with up to eight factors.

Keywords: Artificial intelligence, chain-of-thought, engineering, orthogonal array, screening, zero-shot prompt.

^{*}ORCID: 0000-0002-3658-0911

1 Introduction

Design of experiments (DoE) is a subfield of statistics that involves the design and analysis of cost-effective experiments to study complex processes. Typically, a process has several input factors that can be set deliberately and a response that measures the quality characteristic of the process' product. DoE methods, such as experimental plans and statistical models, are used to elucidate the true relationship between the response and the factors.

Two-level fractional factorial designs (Wu and Hamada, 2011) are commonly used experimental plans that can study many factors using an economical number of runs. In these designs, all factors are set at two levels, allowing the study of their main effects and interactions. We denote a two-level fractional factorial design as 2^{m-p} , where m is the number of factors and p is the number of generated factors. The generated factors are obtained from the interactions of $b = m - p$ basic factors, whose 2^b level combinations are all in the design (Wu and Hamada, 2011). Design 2^{m-p} is thus a 2^p fraction of the 2^m full factorial design, with a run size that is a power of two.

The DoE literature has complete catalogs of two-level fractional factorial designs. For instance, Chen et al. (1993) provide a catalog of designs with 8, 16, 32, and 64 runs, and up to 32 factors. Ryan and Bulutoglu (2010) provide a catalog of 128-run designs with up to 64 factors. The best designs in these catalogs are available in standard DoE textbooks such as Wu and Hamada (2011) and Montgomery (2017), or statistical software such as JMP, Minitab, and the FrF2 package (Grömping, 2014) in R.

In recent years, generative artificial intelligence (GenAI) chatbots, such as ChatGPT (OpenAI, 2025) and Gemini (Google, 2025), have emerged as disruptive technology that has transformed the way we learn, write, access information, make decisions, and even conduct research. For example, GenAI chatbots have been used to solve optimization problems (Bertsimas and Margaritis, 2024), prove mathematical theorems (Frieder et al., 2023), teach statistics (Ellis and Slade, 2023), generate code for data analyses (Song et al., 2025), and build search engine software for statistical quality control (Megahed et al., 2024). The success of GenAI chatbots is due to their use of Large Language Models (LLMs; Alammari and Grootendorst, 2024) and the high computational power available these days (TOP500 Project, 2025). Technically, an LLM is a deep neural network (Goodfellow et al., 2016) with billions of parameters that are estimated through a training process that uses an

enormous amount of text data. These data are typically extracted from the Internet and may include books, blogs, research and news articles, social media, among others. The goal of the training process is to predict the next word in a sentence. To this end, the LLM uses a neural network architecture called transformer (Vaswani et al., 2017), which allows it to learn the context and relationships between words in the text during its training.

In particular, ChatGPT and Gemini use the Generative Pre-trained Transformer (GPT; OpenAI, 2025) model and the Gemini model (Comanici et al., 2025), respectively, which are sophisticated, pre-trained transformer-based LLMs. To perform a specific task, their main input (as well as of other LLMs) is a user-specified text called *prompt* (Schulhoff et al., 2024), which describes the question, instruction, or guidelines for it. At the time of writing this article, the versions of the LLMs used by default in ChatGPT’s and Gemini’s web interfaces are GPT-5.1 and Gemini 2.5 Flash, respectively. Remarkably, these LLMs can reason across audio, vision, and text in real time. Moreover, they can be accessed using application programming interfaces (APIs) for experimentation.

In this paper, we evaluate the performance of the GPT-5.1 and Gemini 2.5 Flash models for constructing two-level fractional factorial designs. Our research goal is motivated by the following interaction between us and ChatGPT with the GPT-5.1 model available at <https://chatgpt.com/>. Prompt 1 shows the instruction we gave to ChatGPT.

Prompt 1: Simple prompt to construct a two-level design.

Construct a two-level fractional factorial design with four factors and eight runs.

ChatGPT’s output 1 shows a summary of the response of the chatbot to Prompt 1. The symbol ‘:’ indicates that text has been omitted to streamline the output.

ChatGPT's Output 1: Summary of output to Prompt 1.

A **two-level fractional factorial design with four factors (A, B, C, D) and eight runs** is a 2^{4-1} design: a half fraction of the full $2^4 = 16$ runs.

Choose a generator

A common choice is:

$$D = ABC$$

This produces a **Resolution IV** design.

Construct the design

Start with the full 2^3 design for factors **A, B, C**, then set:

$$D = A \times B \times C$$

(where levels are coded as -1 and +1 or as - and +).

Design Table (- = low, + = high)

Run	A	B	C	D = ABC
1	-	-	-	-
2	+	-	-	+
3	-	+	-	+
4	+	+	-	-
5	-	-	+	+
6	+	-	+	-
7	-	+	+	-
8	+	+	+	+
			\vdots	

ChatGPT's output 1 shows a 2^{4-1} design that matches the 4-factor 8-run design in Table 8.14 of Montgomery (2017). The four factors are labeled A , B , C , and D , where the first three are basic factors and the last is the generated factor obtained as $D = ABC$. That is, the column for D is the element-wise product of the columns that accommodate A , B , and C , when their levels are coded as -1 and 1 . The design matrix is shown in the

Design Table section. Its first column shows labels for the runs and the other columns show the factors, whose levels are labeled using the symbols ‘−’ and ‘+’. This 2^{4-1} design can estimate the main effects and two-factor interactions of the four factors simultaneously. In fact, it is the best design in terms of the resolution and minimum aberration criteria introduced in Section 2. The Appendix shows a similar interaction between us and Gemini with the Gemini 2.5 Flash model, in which the same 2^{4-1} design was produced.

Our interactions with ChatGPT and Gemini show that their default LLMs have the potential to construct two-level fractional factorial designs. Therefore, practitioners may feel tempted to construct these designs from scratch using the chatbots, bypassing statistical software and textbooks. To explore the capabilities and limitations of this approach, we conducted a systematic study to evaluate the performance of the GPT-5.1 and Gemini 2.5 Flash models on 36 tasks concerning the construction of two-level fractional factorial designs. The tasks involve designs with 8, 16, and 32 runs, and 4 to 26 factors. Each task has a prompt to instruct the LLM to construct a design. To this end, we develop a prompt template using prompting techniques such as role, context, chain of thought, and output format (Kojima et al., 2022; White et al., 2023; Schulhoff et al., 2024). Our prompt template belongs to the class of *zero-shot* prompts, introduced in Section 3, and is intended to test the general knowledge of LLMs on constructing two-level fractional factorial designs without examples.

We evaluate the computational performance of the LLMs using 10 replicates of each task. We show that the LLMs have the potential to construct optimal designs with up to 15 factors in terms of the criteria in Section 2. However, the Gemini 2.5 Flash model is better than the GPT-5.1 model because it has a high consistency in obtaining optimal 8-run designs, almost all optimal 16-run designs, and the optimal 32-run 6-factor design. For designs of other sizes, the performance of both LLMs deteriorates because they cannot consistently construct the optimal design or cannot construct a design at all.

The remainder of the paper is organized as follows. In Section 2, we introduce the criteria for two-level fractional factorial designs. In Section 3, we develop our prompt template and provide the computational setup of the LLMs. In Section 4, we discuss the performance of the LLMs on our design construction tasks. In Section 5, we end the paper with remarks and directions for future research. The R and Python codes to conduct our experiments are available on GitHub at <https://github.com/alanrvazquez/LLMforDOE>.

2 Criteria for two-level fractional factorial designs

We review resolution, minimum aberration, and minimum moment aberration (Wu and Hamada, 2011; Xu, 2003) to evaluate two-level fractional factorial designs. We illustrate these criteria using a 2^{7-3} design. Its basic factors are A , B , C , and D , and the generated factors are obtained as $E = ABC$, $F = ABD$, and $G = ACD$. The defining relation of the design is

$$I = ABCE = ABDF = ACDG = CDEF = BDEG = BCFG = AEFG. \quad (1)$$

Recall that, in a defining relation, I is the identity and the other elements are called words. For the benefit of the reader, Table 1 shows the 2^{7-3} design in full.

Table 1: A two-level fractional factorial design with 16 runs and seven factors.

Run	A	B	C	D	E	F	G
1	1	-1	-1	-1	1	1	1
2	-1	-1	-1	1	-1	1	1
3	-1	-1	-1	-1	-1	-1	-1
4	-1	1	1	-1	-1	1	1
5	1	-1	1	1	-1	-1	1
6	-1	-1	1	1	1	1	-1
7	-1	1	1	1	-1	-1	-1
8	1	1	1	1	1	1	1
9	-1	1	-1	1	1	-1	1
10	1	-1	-1	1	1	-1	-1
11	1	1	-1	-1	-1	-1	1
12	1	1	1	-1	1	-1	-1
13	-1	-1	1	-1	1	-1	1
14	1	1	-1	1	-1	1	-1
15	-1	1	-1	-1	1	1	-1
16	1	-1	1	-1	-1	1	-1

2.1 Resolution and minimum aberration

The resolution is the length of the shortest word in the defining relation (Wu and Hamada, 2011). It allows us to know the properties of a 2^{m-p} design for studying main effects and two-factor interactions, which are the most important according to the effect hierarchy principle (Wu and Hamada, 2011, ch 4). For instance, resolution-3 designs provide main effects that are not aliased with each other, but at least one main effect is aliased with one or more two-factor interactions. In contrast, resolution-4 designs provide main effects that are not aliased with each other nor with two-factor interactions. However, at least one pair of two-factor interactions is aliased in these designs. Resolution-5 designs are optimal for studying main effects and two-factor interactions because they avoid aliasing among these effects. So, when comparing designs in terms of their resolutions, the design with the higher resolution is preferred. From Equation (1), we have that the 2^{7-3} design has a resolution of four.

The minimum aberration criterion allows us to differentiate between designs with the same resolution. It is based on the word length pattern (WLP; Wu and Hamada, 2011), which is a vector that collects the number of words of length 1 to m in the defining relation. Specifically, this vector is (A_1, A_2, \dots, A_m) where A_i is the number of words of length i in the defining relation. Note that the resolution of a design is the smallest value of i for which $A_i > 0$. From Equation (1), we have that the WLP of the 2^{7-3} design is $(0, 0, 0, 7, 0, 0, 0)$. So, the defining relation of this design has seven words of length four.

The minimum aberration criterion is to sequentially minimize the WLP from left to right. In this way, the criterion favors designs with less aliasing involving low-order effects of the factors than high-order ones. Chen et al. (1993) show that the 2^{7-3} design in Table 1 has minimum aberration among all two-level designs of the same size.

2.2 Minimum moment aberration

For a large number of factors, comparing designs in terms of minimum aberration can become computationally expensive. This is because the computational complexity of the WLP is $O(n2^m)$, where n is the design's run size. To overcome this issue, Xu (2003) introduced minimum moment aberration to compare two-level designs. This criterion is equivalent to minimum aberration but uses the moment aberration pattern, a vector based

on the similarity between the design’s runs. Let \mathbf{d}_i and \mathbf{d}_j be the i -th and j -th run of a 2^{m-p} design. We consider $\delta(\mathbf{d}_i, \mathbf{d}_j)$ as a function that counts the number of entries in which \mathbf{d}_i and \mathbf{d}_j coincide. The t -th moment of the 2^{m-p} design is

$$K_t = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n [\delta(\mathbf{d}_i, \mathbf{d}_j)]^t}{n(n-1)/2}.$$

In statistical terms, K_t is the t -th non-central moment of the empirical distribution of similarities between two runs of the design.

The moment aberration pattern is (K_1, K_2, \dots, K_m) , which has a computational complexity of $O(n^2 m^2)$ (Xu, 2003). This complexity is much less than that of the WLP when m is large. For two-level designs, Xu (2003) shows a lower bound K'_i for each element of the moment aberration pattern, which can be used to compute the resolution. That is, the resolution of a design is the smallest value of i for which $K_i > K'_i$.

Similar to minimum aberration, the minimum moment aberration criterion is to minimize the moment aberration pattern from left to right. The moment aberration pattern of the 2^{7-3} design in Table 1 is (3.27, 11.67, 42.47, 157.27, 591.27, 2251.67, 8666.47). Since this design has minimum aberration, it also has minimum moment aberration among all designs of the same size.

3 Methodology

Our methodology to construct two-level fractional factorial designs using LLMs has two main elements. The first one is a prompt template with precise instructions to construct designs, which we develop using prompting techniques (White et al., 2023; Schulhoff et al., 2024). The second one is a computational setup for interacting with the LLMs, which involves APIs and dedicated Python packages. We present each of these elements separately.

3.1 Prompting techniques

The input of an LLM is a prompt, the writing of which has a critical influence on the quality of the generated output (Schulhoff et al., 2024). To this end, there are several prompting techniques for crafting prompts, such as role, context, chain of thought, output format, and examples or *shots* (Kojima et al., 2022; White et al., 2023; Schulhoff et al., 2024;

Sahoo et al., 2024). We use these techniques to develop our prompt template, except for the last one for three reasons. First, our template attempts to test the general knowledge of LLMs for constructing two-level fractional factorial designs from scratch using a single well-crafted instruction. Second, writing good examples of two-level fractional factorial designs and their construction requires profound DoE knowledge and expertise, which we do not assume our practitioner has. Third, adding examples to our prompt will significantly increase its length, thereby making it more expensive to run in our APIs; see Section 3.2. We comment on the benefits and challenges of prompts with examples, called *few-shot* prompts in the prompting literature, in Section 5. Since our prompt template does not include examples, it belongs to the class of *zero-shot* prompts, which are relevant in practice (Schulhoff et al., 2024; Kojima et al., 2022).

3.1.1 Role

The performance of an LLM for a given task can be improved by assigning its *role* in the prompt (Wang et al., 2024). This allows the LLM to focus on its expertise in the given role to provide the output. The technique of assigning a role (or *persona*) to an LLM in the prompt is called role prompting (Wang et al., 2024). Ronanki et al. (2025), Kong et al. (2023) and Hadi et al. (2023) discuss applications of role prompting in engineering, mathematics, and legal practice, respectively.

To construct two-level fractional factorial designs, we assign the role of expert in DoE to the LLM. Specifically, we start our prompt template with the sentence: “You are an expert in the subfield of statistics called design of experiments.” In this role, we make explicit that DoE is a subfield of statistics, so that the LLM concentrates on the experimental design techniques discussed here. In other words, we expect the LLM to focus on two-level fractional factorial designs, resolution, and minimum aberration, which are standard topics in statistical DoE (Smucker et al., 2023; Vazquez and Xuan, 2025).

3.1.2 Context

The prompt given to an LLM must have the context needed to solve a given problem. This allows the LLM to generate a high-quality solution to the problem. If the prompt lacks context, the LLM might underachieve in the problem or generate a less informative output

(Sahoo et al., 2024; Schulhoff et al., 2024).

In our prompt template, we provide context on a design construction task in two ways. First, we state the goal in our prompt by adding the sentence: “Your goal is to construct a two-level fractional factorial design with maximum resolution and minimum aberration.” In this goal, we include the criteria for two-level designs in Section 2.1, which should be known by the LLM due to their role established in the previous section. So, we expect it to construct the best design in terms of these criteria. The goal does not involve minimum moment aberration because it is equivalent to minimum aberration; something that the LLM should know too because of its role.

Second, we state the desired number of factors, run size, and levels in the context given in our prompt template. Specifically, we add the following sentences to it: “The number of factors is m and the number of runs is n . The factors will be studied at two levels, which are coded as ‘ -1 ’ and ‘ 1 ’.” These sentences state that the coded levels of the factors must be -1 and 1 , which allow us to evaluate the resulting design using our criteria in Section 2. In the sentences, the symbols m and n must be replaced with the desired number of factors and runs, respectively.

3.1.3 Zero-shot chain of thought

A chain of thought (CoT) is a series of intermediate natural language reasoning steps that can be added to a prompt to enhance the output of an LLM (Wei et al., 2022). This technique allows the LLM to decompose the main problem into a series of smaller intermediate problems, each solved before giving the final output. Wei et al. (2022) show that adding examples written using CoT to prompts improves the performance of transformer-based LLMs in arithmetic, symbolic reasoning, and common sense tasks.

Kojima et al. (2022) show that CoT can also improve the quality of zero-shot prompts. Specifically, they show that adding the sentence “Let’s think step by step,” or a similar text to a zero-shot prompt, generally enhances the performance of transformer-based LLMs in similar tasks as Wei et al. (2022). Kojima et al. (2022) refer to the technique of adding this sentence (or one of its variants) to a zero-shot prompt as Zero-shot-CoT. Attractive features of this technique include its simplicity, task independence, and the fact that it avoids the need for good CoT-based examples, which require significant expertise and careful writing.

We use Zero-shot-CoT in our prompt template by including the sentence “You will think step by step about how to construct the design.” In this way, we expect to leverage the reasoning capabilities of the LLMs used here to produce attractive designs.

3.1.4 Output format

We can set a format for an LLM to tailor the structure of their output to our needs (White et al., 2023). For example, we can provide constraints on the length and format of its output in the prompt. An output format is particularly useful for our problem because the LLMs tend to provide two-level designs with additional text to explain its properties and construction; see ChatGPT’s output 1 and Gemini’s output A1 in the Appendix. However, for our numerical experiments, we only need a table with the design in a specific format. That is, the table must have m columns and n rows with entries equal to -1 or 1 . The table must also have a header row with the labels of the factor columns and an additional column to label the runs.

To achieve the desired format for the design table, we develop an output format with two elements that are included in our prompt template. The first one involves two sentences to avoid the explanations of the LLM and produce a design table only. These sentences are: “However, you will only generate a table containing the design. You will not generate any text explaining the table or your answer.”

The second element of our output format allows us to streamline the LLMs output because it sets the format for the table. Specifically, it instructs the LLM to generate the table using a comma-separated values (CSV) format, which is common for storing tables, matrices, and arrays. In this format, every entry in the table is separated by a comma. We specify the end of a row using two backslash symbols (`\\`). To set our second format element, we add the following sentences to our prompt: “The table must be in a comma-separated values (CSV) format. Specifically, the values in the table must be separated by ‘,’ and each row must end with ‘\\’. In the table, the first row will be used as a header row to label the factors using the letters of the English alphabet starting with ‘A’. The first column will be called “Run” and used to count the number of runs starting at ‘1’. Each design cell (excluding the header and Run columns) must contain either ‘-1’ or ‘1’.” We instruct the LLMs to label factors using the letters of the English alphabet following Montgomery

(2017), ChatGPT’s output 1, and Gemini’s output A1. Moreover, the maximum number of factors in our experiments in Section 4 is 26, which is the number of letters in this alphabet.

Our output format results in a design table suitable for evaluation using the criteria in Section 2 and our Python code. To our knowledge, this is the first output format used to obtain a two-level fractional factorial design table from an LLM.

3.1.5 Prompt template

Using the previous prompting techniques, we develop the prompt template in Prompt 2 to construct two-level fractional factorial designs using LLMs. The template has a final instruction to restate the goal of the task. That is, Prompt 2 ends with the sentence “Construct the two-level fractional factorial design with m factors and n runs that has maximum resolution and minimum aberration.” Recall that the symbols m and n must be replaced with the desired number of factors and runs, respectively. Due to its use of the Zero-shot-CoT technique, we refer to Prompt 2 as a Zero-shot-CoT prompt.

Prompt 2: Prompt template to construct a two-level design.

You are an expert in the subfield of statistics called design of experiments. Your goal is to construct a two-level fractional factorial design with maximum resolution and minimum aberration. The number of factors is m and the number of runs is n . The factors have two levels coded as ‘-1’ and ‘1’. You will think step by step about how to construct the design. However, you will only generate a table containing the design. You will not generate any text explaining the table or your answer. The table must be in a comma-separated values (CSV) format. Specifically, the values in the table must be separated by ‘,’ and each row must end with ‘\’’. In the table, the first row will be used as a header row to label the factors using the letters of the English alphabet starting with ‘A’. The first column will be called “Run” and used to count the number of runs starting at ‘1’. Each design cell (excluding the header and Run columns) must contain either ‘-1’ or ‘1’. Construct the two-level fractional factorial design with n runs and m factors that has maximum resolution and minimum aberration.

Compared to Prompt 1, Prompt 2 provides richer instructions to an LLM because it

defines its role, sets the goal and context of the task, requests to complete the task step by step, and provides a template for the output. Therefore, Prompt 2 has the potential to produce two-level fractional factorial designs with good statistical properties.

3.2 Setup for the LLMs

To construct designs with Prompt 2, we use the GPT-5.1 and Gemini 2.5 Flash models, which are the default LLMs in ChatGPT’s and Gemini’s web interfaces, respectively, at the time of writing this article. Technically, these LLMs are **gpt-5.1-chat-latest** (OpenAI, 2025) and **gemini-2.5-flash** (Google, 2025). We run them using the Python packages called `openai` (OpenAI, 2023) and `google-genai` (Google, 2023), and API keys from OpenAI and Google. An API key is a unique code that can be used to access and communicate with an API, which, in turn, provides access to an LLM. The `openai` and `google-genai` packages provide user-friendly Python functions to interact with the APIs.

Compared to ChatGPT’s and Gemini’s web interfaces, our setup allows us to perform multiple design construction tasks with Prompt 2 and the LLMs automatically. It also allows us to set up a separate conversation involving Prompt 2 and an LLM for each design construction task. In this way, we ensure the behavior of our practitioner across all tasks. That is, a user whose first query for the LLM is Prompt 2 and for whom there is no prior interaction history with the LLM. Although we can create independent conversations with the LLMs in ChatGPT’s and Gemini’s web interfaces, we must do this manually, which makes it difficult to perform all of our design construction tasks in the next section.

The Python packages and APIs enable more tuning parameters of the LLMs than ChatGPT’s and Gemini’s web interfaces. For example, through the `google-genai` package, we can set the parameters called *temperature*, *top-k*, and *top-p* of **gemini-2.5-flash**, which control the randomness and diversity of its output to multiple runs of the same prompt (Google Cloud, 2025). We can also set the parameter called *thinkingBudget*, which defines the strategy for reasoning of this LLM. The `openai` package provides access to similar tuning parameters of **gpt-5.1-chat-latest**. In our experiments, we use the default values of all tuning parameters of **gpt-5.1-chat-latest** and **gemini-2.5-flash**, except for the parameter that controls their reasoning. For **gpt-5.1-chat-latest**, we set the *reasoning-effort* parameter to “medium.” For **gemini-2.5-flash**, we set *thinkingBudget* to “-1” and

use the dynamic thinking strategy, which lets the LLM decide when and how much to think based on the complexity of the request.

Running an LLM using an API key has associated costs that depend on the number of *tokens* involved in the input and output of the model. Essentially, tokens are the data units of an LLM. They are constructed from text by the LLM using a process called *tokenization*, which considers language, punctuation, and context, among other features. For input and output text, **gpt-5.1-chat-latest** charges \$1.5 and \$10 for one million tokens, respectively (OpenAI, 2025). For **gemini-2.5-flash**, these costs are \$0.3 and \$2.5 (Google AI for Developers, 2025). Therefore, working with long prompts and outputs is more expensive than with short ones. In this regard, a benefit of our Zero-shot-CoT prompt template is that it allows us to limit the costs of our numerical experiments, because they involve many executions of Prompt 2.

4 Numerical results

We evaluate the performance of the GPT-5.1 and Gemini 2.5 Flash models for design construction tasks involving two-level fractional factorial designs with 8, 16, and 32 runs. For these run sizes, the number of factors in our tasks ranges from 4 to 7, 5 to 15, and 6 to 26, respectively. The number of tasks thus is 36. For each task, we construct 10 designs from 10 independent executions of an LLM with Prompt 2 as input. Therefore, we construct a total of $10 \times 36 = 360$ designs using each LLM. The order in which we perform the tasks with each LLM is random.

We classify the designs obtained from an LLM as compliant or non-compliant. A design is compliant if the resulting design table has complete entries and the requested number of factors, levels, and runs in the task. Otherwise, the design is non-compliant. In some of our experiments, the compliant designs did not strictly follow the output format in Prompt 2. This was mainly because they had extra commas or missed a backslash symbol at the end of a row of the table. In these cases, we manually corrected their format so that they could be evaluated using our Python code.

We evaluate the compliant designs in terms of resolution and minimum moment aberration only, since they are computationally cheap and the latter is equivalent to minimum aberration. In Section 4.1, we show the numerical results of the GPT-5.1 model for the 8-,

16-, and 32-run design tasks separately. In Section 4.2, we discuss the performance of the Gemini 2.5 Flash model on these tasks.

4.1 Designs constructed using the GPT-5.1 model

4.1.1 Tasks with eight runs

Table 2 shows the performance of the GPT-5.1 model on the design construction tasks with eight runs. Specifically, the table shows the task label and the maximum, minimum, and median resolutions of the designs obtained by the LLM. For each task, the 10 designs constructed by the LLM were compliant.

Table 2: Performance of the GPT-5.1 model on 8-run design tasks.

Task	Factors	Resolution		
		Min.	Median	Max.
1	4	4	4	4
2	5	2	3	3
3	6	2	2.5	3
4	7	3	3	3

Table 2 shows that the resolution of the designs with four and seven factors is four and three, respectively. For five and six factors, the resolution is two or three. In contrast, all two-level fractional factorial designs with eight runs and four to seven factors in Wu and Hamada (2011) have a resolution of at least three. The same is true for the 16- and 32-run fractional factorial designs, which we discuss later.

For each number of factors, the resolution of the best design obtained by the GPT-5.1 model equals that of the minimum aberration design in Wu and Hamada (2011). To further compare these designs, we computed their moment aberration patterns. Table 3 shows the patterns of the best designs obtained by the LLM for each task. The best designs obtained by this LLM have the same moment aberration patterns as the minimum aberration designs. Therefore, this LLM was able to construct the best designs for the 8-run design tasks. Regarding the consistency in obtaining them, the GPT-5.1 model found the

best 4-, 5-, 6-, and 7-factor designs in 10, 8, 5, and 10, respectively, of the 10 independent executions of the corresponding Prompt 2; see Table 3.

Table 3: Moment aberration patterns obtained by the GPT-5.1 model for the 8-run design construction tasks.

Task	Factors	Resolution	Moment aberration pattern	Frequency
1	4	4	(1.7, 3.4, 6.9, 13.7)	10
2	5	3	(2.1, 5.0, 12.4, 32.4, 87.9)	8
3	6	3	(2.6, 6.9, 18.9, 53.1, 152.6, 444.0)	5
4	7	3	(3.0, 9.0, 27.0, 81.0, 243.0, 729.0, 2187.0)	10

4.1.2 Tasks with 16 runs

Table 4 shows the resolutions of the designs obtained by the GPT-5.1 model for the 16-run design tasks. The table also shows the number of compliant designs that were generated for each task. The LLM obtained 10 compliant designs for each task, except for the tasks with 12 to 15 factors. For these tasks, it only obtained eight or nine compliant designs. A close inspection to the non-compliant designs revealed that they had missing entries or were infeasible for the LLM. For the latter case, the LLM output a sentence similar to: “I’m sorry, but I cannot generate a valid design that meets your requirements.”

Table 4 shows that the median resolutions of the 16-run designs are higher than or equal to three, except for the designs with 9, 11, 14, and 15 factors. For these designs, the median resolution was 1.5, 2.5 or 2.

For all numbers of factors, there were multiple compliant designs with the same resolution as the corresponding minimum aberration designs (Wu and Hamada, 2011). Table 5 shows the frequencies of the moment aberration patterns of these designs for each task. To enhance its layout, the table shows the first four elements of the moment aberration pattern. For each number of factors, the GPT-5.1 model obtained the minimum aberration design at least twice, except for 12 factors for which it did not find the minimum aberration design. Interestingly, for tasks with five to eight factors, this LLM obtained the minimum aberration design in at least eight repetitions of these tasks. Therefore, the GPT-5.1 model

Table 4: Performance of the GPT-5.1 model on 16-run design tasks.

Task/Factors	Resolution			# Compliant
	Min.	Median	Max.	
5	1	5	5	10
6	1	4	4	10
7	1	4	4	10
8	1	4	4	10
9	1	2	3	10
10	1	3	3	10
11	1	2.5	3	10
12	1	3	3	9
13	1	3	3	8
14	1	2	3	8
15	1	1.5	3	8

performs well in constructing two-level fractional factorial designs with 16 runs and up to eight factors.

4.1.3 Tasks with 32 runs

Table 6 shows the resolution and number of compliant designs obtained by the GPT-5.1 model for the 32-run design tasks. For 11, 13, 16, and 19 to 26 factors, this LLM did not find a compliant design with a resolution higher than two. For this reason, these numbers of factors are omitted in the table. Table 6 shows that there were few compliant designs for tasks with more than eight factors. Similar to the 16-run design tasks, the 32-run noncompliant designs had missing entries or were infeasible for the LLM.

For 6-9, 12, 17, and 18 factors, the maximum resolution in Table 6 matches that of the minimum aberration designs (Wu and Hamada, 2011). For these numbers of factors, Table 7 shows the first six elements of the moment aberration patterns of the designs with the highest resolution obtained by the GPT-5.1 model. For 6-9 and 12 factors, the LLM found the minimum aberration design at least once. This is because the best moment

Table 5: Moment aberration patterns obtained by the GPT-5.1 model for the 16-run design tasks. *: Design has minimum moment aberration.

Task/Factors	Resolution	Moment aberration pattern	Frequency
5	5	(2.3, 6.3, 18.3, 54.3)*	9
6	4	(2.8, 8.8, 28.8, 97.6)*	8
7	4	(3.3, 11.7, 42.5, 157.3)*	8
8	4	(3.7, 14.9, 59.7, 238.9)*	8
9	3	(4.2, 18.6, 84.2, 386.6)*	3
		(4.2, 18.6, 86.6, 421.8)	2
		(4.2, 18.6, 87.4, 437.8)	1
10	3	(4.7, 22.7, 113.1, 575.5)*	2
		(4.7, 22.7, 113.9, 588.3)	1
		(4.7, 22.7, 114.7, 602.7)	3
		(4.7, 22.7, 114.7, 604.3)	1
11	3	(5.1, 27.1, 146.7, 807.9)*	2
		(5.1, 27.1, 147.5, 823.9)	2
		(5.1, 27.1, 147.5, 825.5)	1
12	3	(5.6, 32, 186.4, 1105.6)	5
13	3	(6.1, 37.3, 231.7, 1456.5)*	5
14	3	(6.5, 42.9, 283.7, 1885.3)*	3
15	3	(7.0, 49, 343, 2401.0)*	3

Table 6: Performance of the GPT-5.1 model on 32-run design tasks.

Task	Factors	Resolution			# Compliant
		Min.	Median	Max.	
16	6	1	6	6	10
17	7	1	4	4	9
18	8	1	3.5	4	8
19	9	1	1	4	3
20	10	1	2	3	2
22	12	2	3	4	2
24	14	1	2	3	4
25	15	1	2.5	3	4
27	17	1	2	3	2
28	18	2	2.5	3	2

aberration pattern is equal to that of the minimum aberration design for these cases. For 17 and 18 factors, the best moment aberration patterns are worse than those of the minimum aberration designs. Overall, we conclude that the GPT-5.1 model is moderately effective for constructing the 32-run 6-factor minimum aberration design, but ineffective for generating 32-run minimum aberration designs with more factors.

4.2 Designs constructed using the Gemini-2.5 Flash model

We performed the design construction tasks with 8, 16, and 32 runs using the Gemini 2.5 Flash model. Remarkably, this LLM excelled in the 8-run design tasks because it obtained the minimum aberration designs for all factors in all independent executions of Prompt 2. This LLM also obtained 10 compliant designs for each 16-run design task. Table 8 shows the resolutions of these designs.

The Gemini 2.5 Flash model constructed 16-run designs with the same resolution as the minimum aberration designs. Table 9 summarizes the moment aberration patterns of the 16-run designs constructed by this LLM. The table is akin to Table 5 for the GPT-5.1 model, and it concerns the designs with the highest resolution obtained for each task.

Table 7: Moment aberration patterns obtained by the GPT-5.1 model for selected 32-run design tasks. *: Design has minimum moment aberration.

Task	Factors	Resolution	Moment aberration pattern	Frequency
16	6	6	(2.9, 9.7, 34.8, 131.6, 511.0, 2012.9)*	6
17	7	4	(3.4, 12.9, 52.2, 220.4, 959.5, 4278.7)*	1
			(3.4, 12.9, 52.2, 221.9, 978.9, 4437.4)	1
			(3.4, 12.9, 52.2, 223.5, 1006.0, 4735.5)	4
18	8	4	(3.9, 16.5, 74.3, 346.3, 1656.8, 8099.1)*	1
			(3.9, 16.5, 74.3, 349.4, 1703.2, 8555.9)	1
			(3.9, 16.5, 74.3, 352.5, 1765.2, 9337.8)	2
19	9	4	(4.4, 20.6, 101.9, 517.6, 2683.1, 14141.9)*	1
22	12	4	(5.8, 35.6, 223.0, 1423.0, 9243.9, 61042.1)*	1
27	17	3	(8.2, 69.6, 608.2, 5485.9, 51108.9, 491295.5)	1
28	18	3	(8.7, 77.8, 714.6, 6758.7, 65883.9, 661601.0)	1

Except for 11 and 12 factors, the Gemini 2.5 Flash model obtained the 16-run minimum aberration designs at least twice; see Table 9.

Regarding the 32-run design tasks, the LLM constructed 10 compliant designs for each number of factors, except for 17, 18, and 20 to 26 factors. For these cases, the LLM generated several non-compliant designs with missing entries or other issues. For tasks with 6-9, 10, and 14 factors, the compliant designs had a resolution higher than or equal to three. The distribution of the resolution is summarized in Table 10 for these cases. For six to nine factors, the maximum resolutions in this table equal those of comparable 32-run minimum aberration designs. Table 11, which is akin to Table 7, shows that the Gemini 2.5 Flash model obtained the minimum aberration designs with six to eight factors at least once. Remarkably, for six factors, the LLM constructed the 32-run minimum aberration design in the 10 independent executions of the corresponding Prompt 2.

Compared to the GPT-5.1 model, the Gemini 2.5 Flash model had a success rate of 100% in constructing minimum aberration designs with eight runs and four to seven factors, and with 32 runs and six factors. Moreover, its success rate for constructing 16-run minimum

Table 8: Performance of the Gemini 2.5 Flash model on the 16-run design tasks.

Task/Factors	Resolution		
	Min.	Median	Max.
5	3	5	5
6	1	4	4
7	1	4	4
8	1	4	4
9	1	3	3
10	2	3	3
11	1	2	3
12	1	2.5	3
13	1	1.5	3
14	1	1	3
15	1	1	3

aberration designs with five to eight factors was at least 80%. However, the GPT-5.1 and Gemini 2.5 Flash models were not successful in constructing minimum aberration designs of other sizes, since their success rate was 50% or less.

5 Conclusion

We assessed the quality of two-level fractional factorial designs constructed by the GPT-5.1 and Gemini 2.5 Flash models using Zero-shot-CoT prompts. To this end, we developed a prompt template to tackle design construction tasks with 8, 16, and 32 runs, and 4 to 26 factors. Using the template as input, we generated 10 designs for each task using 10 independent executions of the LLMs. We compared the designs obtained with the optimal designs in terms of resolution, minimum aberration, and minimum moment aberration. We showed that the LLMs constructed all 8- and 16-run minimum aberration designs at least twice, except for the design with 16 runs and 12 factors. They also generated 32-run minimum aberration designs with six to nine and 12 factors at least once.

Table 9: Moment aberration patterns of obtained by the Gemini 2.5 Flash model for selected 16-run design tasks. *: Design has minimum moment aberration.

Task/Factors	Resolution	Moment aberration pattern	Frequency
5	5	(2.3, 6.3, 18.3, 54.3)*	8
6	4	(2.8, 8.8, 28.8, 97.6)*	9
7	4	(3.3, 11.7, 42.5, 157.3)*	9
8	4	(3.7, 14.9, 59.7, 238.9)*	8
9	3	(4.2, 18.6, 84.2, 386.6)*	2
		(4.2, 18.6, 86.6, 421.8)	4
10	3	(4.7, 22.7, 113.1, 575.5)*	3
		(4.7, 22.7, 113.9, 588.3)	1
		(4.7, 22.7, 114.7, 602.7)	5
11	3	(5.1, 27.1, 147.5, 823.9)	3
12	3	(5.6, 32, 186.4, 1105.6)	5
13	3	(6.1, 37.3, 231.7, 1456.5)*	4
14	3	(6.5, 42.9, 283.7, 1885.3)*	3
15	3	(7.0, 49.0, 343.0, 2401.0)*	3

Table 10: Performance of the Gemini 2.5 Flash model on 32-run design tasks.

Task	Factors	Resolution		
		Min.	Median	Max.
16	6	6	6	6
17	7	2	3	4
18	8	2	3	4
19	9	1	3	4
20	10	1	2	3
24	14	1	2	3

Table 11: Moment aberration patterns obtained by the Gemini-2.5 Flash model for selected 32-run design tasks. *: Design has minimum moment aberration.

Task	Factors	Resolution	Moment aberration pattern	Frequency
16	6	6	(2.9, 9.7, 34.8, 131.6, 511.0, 2012.9)*	10
17	7	4	(3.4, 12.9, 52.2, 220.4, 959.5, 4278.7)*	2
			(3.4, 12.9, 52.2, 221.9, 978.9, 4437.4)	1
18	8	4	(3.9, 16.5, 74.3, 346.3, 1656.8, 8099.1)*	1
			(3.9, 16.5, 74.3, 349.4, 1703.2, 8555.9)	1
19	9	4	(4.4, 20.6, 101.9, 522.3, 2756.6, 14896.7)	1

Based on our results, we recommend Gemini with its Gemini 2.5 Flash model to construct 8-run designs with four to seven factors, and a 32-run design with six factors using Prompt 2. We also recommend Gemini and this prompt to generate 16-run designs with five to eight factors, but warn practitioners that there is a small chance that this chatbot will produce poor designs. On the upside, all these numbers of factors are the most common in practical applications according to the systematic surveys of Li et al. (2006) and Ockuly et al. (2017) on real factorial experiments. On the downside, the Gemini 2.5 Flash model is ineffective in constructing attractive designs of other sizes consistently using Prompt 2. For these cases, we advise practitioners to use DoE textbooks (Wu and Hamada, 2011; Montgomery, 2017), JMP, Minitab, or the FrF2 package in R.

There are two potential ways to improve the performance of the GPT-5.1 and Gemini 2.5 Flash models on our design construction tasks. One way is to use a *few-shot* prompt template with CoT-based examples instead of our Zero-shot-CoT prompt template. Despite the improvement in performance of LLMs given by Zero-shot-CoT prompts, Kojima et al. (2022) show that these prompts do not outperform few-shot prompts with CoT-based examples in general. However, developing these examples is challenging because their writing must have all the steps needed to reach the solution of a task’s instance. Moreover, the writing style of these steps influences the performance of LLMs (Kojima et al., 2022). In our setting, developing CoT-based examples would need a careful selection of two-level fractional factorial designs, as they can have different resolutions and WLPs. Moreover,

each example would need a comprehensive step-by-step explanation of their construction. Such an explanation should include the selection of basic and generated factors, the elaboration of the defining relation and WLP, and their evaluation in terms of resolution and minimum aberration. For all these reasons, we leave the use of few-shot prompts with CoT-based examples for constructing designs as a topic for future research.

Another way to improve the performance of the GPT-5.1 and Gemini 2.5 Flash models is using Retrieval-Augmented Generation (RAG; Fan et al., 2024). RAG is a methodology for enhancing the performance of LLMs using domain-specific sources provided by the user. In our context, these sources can be textbooks and research articles on DoE that are open-source (Jones-Farmer et al., 2024). RAG has three main steps. First, it stores the domain-specific sources in a database using encoding methods. Second, for an input prompt, it retrieves the elements from the database that are most similar to the prompt. Third, it informs the LLM about these elements by, for example, using them as the context of the prompt. In this way, RAG helps the LLM generate domain-specific output for a given task. We refer the reader to Megahed et al. (2024) for an application of RAG in statistical quality control. We leave the application of RAG for constructing two-level fractional factorial designs as another topic for future research.

Finally, we studied the default versions of the GPT and Gemini models used by ChatGPT’s and Gemini’s web interfaces at the time of writing this article (December, 2025). Due to the rapid development of research and applications of artificial intelligence, we expect the new versions of these LLMs—or even of other LLMs not studied here, such as the LLaMA model (Touvron et al., 2023)—to become more capable in domain-specific tasks involving Zero-shot-CoT prompts. Our set of 36 design construction tasks thus enriches the catalog of benchmarks used to assess the gains in performance of these versions and LLMs (Ni et al., 2025). We hope to see future studies showcasing the performance of these and other GenAI technologies in our set.

ACKNOWLEDGMENTS

The authors acknowledge that ChatGPT and Gemini were used for basic queries about LLMs and APIs, to construct two-level fractional factorial designs, and for language assistance in some sentences through Writefull (<https://www.writefull.com/>). They were not used to generate or explore ideas, classify literature, or provide coding assistance. The

research of Vazquez is supported by the Challenge-Based Research Fund of Tecnológico de Monterrey under the project CI-EIC-HLT-D-58.

DISCLOSURE OF INTEREST

The authors report that there are no competing interests to declare.

DECLARATION OF FUNDING

No funding was received.

DATA AVAILABILITY STATEMENT

All data and code related to this paper are available on GitHub at <https://github.com/alanrvazquez/LLMforDOE>.

References

- Alammar, J. and Grootendorst, M. (2024). *Hands-On Large Language Models: Language Understanding and Generation*. O'Reilly Media, Inc.
- Bertsimas, D. and Margaritis, G. (2024). Robust and adaptive optimization under a large language model lens. *arXiv preprint arXiv:2501.00568*.
- Chen, J., Sun, D., and Wu, C. F. J. (1993). A catalogue of two-level and three-level fractional factorial designs with small runs. *International Statistical Review*, 61:131–145.
- Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blisstein, M., Ram, O., Zhang, D., Rosen, E., et al. (2025). Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Ellis, A. R. and Slade, E. (2023). A new era of learning: Considerations for ChatGPT as a tool to enhance statistics and data science education. *Journal of Statistics and Data Science Education*, 31:128–133.

- Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.
- Frieder, S., Pinchetti, L., Griffiths, R.-R., Salvatori, T., Lukasiewicz, T., Petersen, P., and Berner, J. (2023). Mathematical capabilities of ChatGPT. *Advances in Neural Information Processing Systems*, 36:27699–27744.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT press.
- Google (2023). google-genai python package. <https://googleapis.github.io/python-genai/>. Accessed: 2025-11-28.
- Google (2025). Gemini 2.5 flash. <https://ai.google.dev/gemini-api/docs/models#gemini-2.5-flash>. Large-language model from Google AI – cost-efficient, multimodal, low-latency.
- Google (2025). Gemini Flash 2.5 (Large language model). Accessed November 25, 2025. URL: <https://google.com/gemini>.
- Google AI for Developers (2025). Gemini developer api pricing — gemini 2.5 flash. <https://ai.google.dev/gemini-api/docs/pricing#gemini-2.5-flash>. Accessed: 2025-12-15.
- Google Cloud (2025). Content generation parameters — Generative AI on Vertex AI. <https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/content-generation-parameters>. Accessed: 2025-12-01.
- Grömping, U. (2014). R package FrF2 for creating and analyzing fractional factorial 2-level designs. *Journal of Statistical Software*, 56:1–56.
- Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., Mirjalili, S., et al. (2023). Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 1:1–26.

- Jones-Farmer, L. A., Megahed, F. M., Chen, Y.-J., Zwetsloot, I., Knoth, S., Montgomery, D. C., and Capizzi, G. (2024). Editorial advice for selecting an open-source license for your next paper: Navigating copyrights for publicly facing AI chatbots. *Journal of Quality Technology*, 56:468–473.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.
- Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X., Wang, E., and Dong, X. (2023). Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702*.
- Li, X., Sudarsanam, N., and Frey, D. D. (2006). Regularities in data from factorial experiments. *Complexity*, 11:32–45.
- Megahed, F. M., Chen, Y.-J., Zwetsloot, I. M., Knoth, S., Montgomery, D. C., and Jones-Farmer, L. A. (2024). Introducing ChatSQC: Enhancing statistical quality control with augmented AI. *Journal of Quality Technology*, 56:474–497.
- Montgomery, D. C. (2017). *Design and Analysis of Experiments*. John Wiley & Sons, 9th edition.
- Ni, S., Chen, G., Li, S., Chen, X., Li, S., Wang, B., Wang, Q., Wang, X., Zhang, Y., Fan, L., Li, C., Xu, R., Sun, L., and Yang, M. (2025). A survey on large language model benchmarks. *arXiv preprint arXiv:2508.15361*.
- Ockuly, R. A., Weese, M. L., Smucker, B. J., Edwards, D. J., and Chang, L. (2017). Response surface experiments: A meta-analysis. *Chemometrics and Intelligent Laboratory Systems*, 164:64–75.
- OpenAI (2023). OpenAI Python Library. <https://github.com/openai/openai-python>.
- OpenAI (2025). ChatGPT: GPT-4 Language Model. <https://chat.openai.com/>. Accessed: 2025-05-28.

- OpenAI (2025). gpt-5.1-chat-latest. <https://platform.openai.com/docs/models/gpt-5.1-chat-latest>. OpenAI Large Language Model.
- Ronanki, K., Arvidsson, S., and Axell, J. (2025). Prompt engineering guidelines for using large language models in requirements engineering. *arXiv preprint arXiv:2507.03405*.
- Ryan, K. J. and Bulutoglu, D. A. (2010). Minimum aberration fractional factorial designs with large N . *Technometrics*, 52:250–255.
- Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927v1*.
- Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., et al. (2024). The prompt report: A systematic survey of prompt engineering techniques. *arXiv preprint arXiv:2406.06608*.
- Smucker, B., Stevens, N., Asscher, J., and Goos, P. (2023). Profiles in the teaching of experimental design and analysis. *Journal of Statistics and Data Science Education*, 31:211–224.
- Song, X., Xie, K., Lee, L., Chen, R., Clark, J. M., He, H., He, H., Min, J., Zhang, X., Zheng, S., et al. (2025). Performance evaluation of large language models in statistical programming. *arXiv preprint arXiv:2502.13117*.
- TOP500 Project (2025). Top500 supercomputer sites. <https://top500.org/>. Accessed: 2025-06-10.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

- Vazquez, A. R. and Xuan, X. (2025). A review of Design of Experiments courses offered to undergraduate students at American universities. *The American Statistician*, 79:129–139.
- Wang, N., Peng, Z., Que, H., Liu, J., Zhou, W., Wu, Y., Guo, H., Gan, R., Ni, Z., Yang, J., et al. (2024). Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14743–14777.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382v1*.
- Wu, C. J. and Hamada, M. S. (2011). *Experiments: Planning, Analysis, and Optimization*. John Wiley & Sons, 2nd edition.
- Xu, H. (2003). Minimum moment aberration for nonregular designs and supersaturated designs. *Statistica Sinica*, 13:691–708.

APPENDIX

Gemini’s output A1 shows a summary of an interaction between us and this GenAI chatbot available at <https://gemini.google.com/app>. The input to Gemini was Prompt 1 and the LLM used was the Gemini-2.5 Flash model. Gemini produced the best 2^{4-1} design in terms of the resolution and minimum aberration criteria.

Gemini's Output A1: Summary of output to Prompt 1.

A two-level fractional factorial design with four factors ($k = 4$) and eight runs ($n = 8$) is a 2^{4-1} **design**. The 2^{4-1} design is a **half-fraction** of the full 2^4 factorial design, resulting in a **Resolution IV** design.

1. Defining the Design

The full 2^4 design has $2^4 = 16$ runs. A half-fraction means we are performing $16/2 = 8$ runs.

- Factors: Let the four factors be **A**, **B**, **C**, and **D**.
- Levels: The two levels for each factor are denoted by -1 (low) and +1 (high).
- Defining Relation/Generator: To create the half-fraction, we need one generator or designate that relates the last factor, **D**, to the interactions of the previous factors **A**, **B**, and **C**. The choice of the generator determines the confounding pattern (aliasing) of the design.

⋮

3. Final Fractional Factorial Design

The resulting 2^{4-1} fractional factorial design with $D = ABC$ is:

<i>Run</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
1	−	−	−	−
2	+	−	−	+
3	−	+	−	+
4	+	+	−	−
5	−	−	+	+
6	+	−	+	−
7	−	+	+	−
8	+	+	+	+

⋮