Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

INVESTIGATING THE USE OF GENERATIVE AI IN M&S EDUCATION

James F. Leathrum, Jr.[1], Yuzhong Shen[1], Masha Sosonkina[1]

[1]Dept. of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA, USA

ABSTRACT

Large Language Models (LLMs) are rapidly creating a place for themselves in society. There are numerous reports, both good and bad, of their use in business, academia, government and society. While some organizations are trying to limit, or eliminate, their use, it appears that it is inevitable they will become a common "tool". In education, there is a fear that students will not acquire critical thinking in the future, but we argue that LLMs will become a tool to assist students with critical thinking, giving guidance, feedback, and assessment. This paper investigates how the current state of LLMs can be integrated into modeling and simulation (M&S) education. Example cases for modeling and simulation development are presented showing how an LLM can assist M&S design and education in anticipation of LLMs becoming a common tool for M&S practitioners. Current limitations are also highlighted, and where possible, short-term solutions are proposed.

1 INTRODUCTION

The introduction of ChatGPT in November 2022 has resulted in the necessity of considering how Large Language Models (LLMs) can be integrated into higher education (Dempere et al. 2023). Higher education has struggled with the acceptance of LLMs, in particular how to recognize academic fraud. (Dempere et al. 2023) highlight several different approaches to the problem including AI-generated material detection tools, online testing platforms, and data analytics to detect anomalies such as in keystrokes. They also point out the potential for biases, stereotypes, and accessibility issues. But LLMs can have a positive effect on higher education as well. There is evidence that chatbots may provide a more positive student interaction than instructors (Essel et al. 2022; Denny et al. 2024). Chatbots have also shown benefit in collaborative student interactions (Tegos et al. 2015) and in improving communication skills (Shorey et al. 2019). This paper looks to build on the potential for positive LLM interactions by investigating how LLMs could assist in the learning process for modeling and simulation engineering (MSE) education.

We view MSE education as comprising four categories, modeling, simulation, data analytics and verification and validation (V&V) as demonstrated in the topology in Figure 1. This vision is based on how LLMs impact MSE rather than on any standard M&S paradigms. Topics represented by yellow-colored ovals are not considered in this paper as they can be considered as standalone topics outside of MSE, though future work will consider the use of LLMs in these topics. In the green oval, the topic verification and validation (V&V) is also not included due to space constraints and will be considered in a separate paper.

The discussion here focuses on the authors' experiences with applying ChatGPT into current MSE educational activities, including lecture discussions and assignments. This results in showing the potential for integrating LLMs into the educational process as well as highlighting observed shortcomings. Initial results are extremely promising, especially in model development, with ChatGPT giving good initial starting points and being responsive to successive prompts to refine a model. However, the lack of control and repeatability of prompts, despite applying prompt engineering, raises issues about an LLM leading a student in the wrong direction regardless of the prompts. A discussion of the maturity of LLMs is provided as they are a moving target making it difficult to settle on a solution. Going forward, ideas on addressing the control of the learning experience are presented.

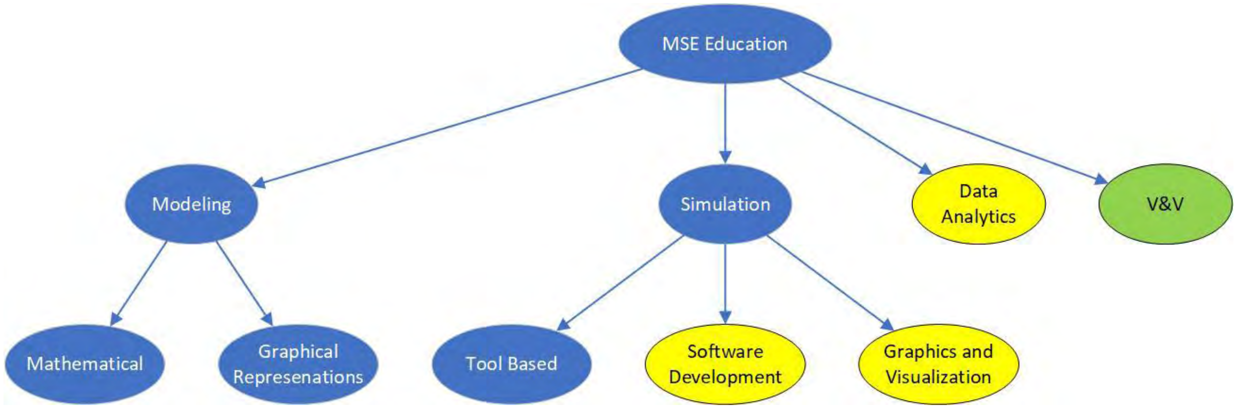


Figure 1. Modeling & Simulation Education (MSE) Toplogy.

2 BACKGROUND

2.1 Large Language Models (LLMs) and Education

Large Language Models (LLMs) represent a breakthrough in natural language processing (NLP) technology and generative artificial intelligence (AI), demonstrating remarkable capabilities in understanding and generating human-like text. The US Department of Education is examining the use of AI in teaching (Cardona et al. 2023). They consider the potential benefits of AI in the classroom as well as some of the concerns such as privacy, ethics and bias. (Edwards 2024) considers how LLMs are transforming computer programming teaching, noting benefits such as students being more comfortable interacting with an LLM than an instructor.

Of particular interest to this paper, the Department of Education report poses a key recommendation for inspectable, explainable, and overridable AI (Cardona et al. 2023) (Cardona et al. 2023) citing the work by (Khosravi et al. 2022). Insight into the decision-making made by AI is proposed to improve instructor trust in the system. They see the need for the control/monitoring of the information provided to different levels of students; for instance, is a remedial student being provided the same information repetitively with no progress (a situation observed by one author when interacting with an LLM). Therefore, they see a need for the instructor to have insight and control of the LLM to guide the learning process. It will be shown that our experience highlights concern with a level of repeatability so that an LLM does not lead a student down an inappropriate path.

A need is also identified that education of the use of LLMs should be included in a curriculum. How a student interacts with an LLM can greatly impact its benefit. An LLM's output is very sensitive to its inputs. Prompt engineering (Denny et al. 2024) should be included early in the education process to teach students how to craft a prompt. Thus, there is a need for educating the front end of the system (the students) and providing control over the back end (the instructor).

2.2 LLMs in Modeling & Simulation (M&S)

The majority of applications of LLMs in M&S appear in medical simulations, traffic simulations and training simulations. The primary uses of LLMs are for scenario generation, analysis, and integrating human models. These applications are logical based on classically requiring a high level of human involvement. LLMs can offload a lot of that work.

Scenario generation is a promising application of LLMs as a common practice is the use of subject matter experts (SMEs) to generate scenarios. (Zhang et al. 2023) propose a process of inputting a knowledge map into an LLM to create the framework for scenario generation. (Tan et al. 2023; Zhang et al. 2023) use

LLMs for traffic generation in simulations. (Tan et al. 2023) and (Deng et al. 2023) utilize LLMs to extract knowledge from test scenarios that were generated using artificial intelligence (AI) to a domain specific language. (Chung et al. 2023) generate case scenarios for trauma surgery simulations.

M&S classically has struggled with including realistic human activity into simulations. (Updyke et al. 2023) simulate the human activity using LLMs. (Li et al. 2023) created LLM-based agents to simulate an environment inspired by job fairs. (Xie et al. 2024; Li et al. 2023) found that LLMs correlate well with human trust behaviors in simulations.

Note that all the works cited are extremely new with publication in the last year, thus results are still rapidly emerging.

3 APPROACH

The premise of our inclusion of LLMs into M&S education lies in LLMs supporting student understanding and in being a tool in V&V. We feel woefully unprepared to discuss V&V at this point, a difficult concept in M&S education as it is. Therefore, we will focus on student understanding. As such, an LLM can be seen in several roles:

- Tutor: An LLM can present material in alternative media, review material, and answer questions.
- Teammate: An LLM can act as a peer, collaborating with students in assignments, providing the ability to "discuss" design ideas.
- Evaluator: An LLM can provide immediate feedback on student work, allowing them to improve the product.

However, to be useful in these roles, several properties of the LLM must be evident. For tutors and evaluators, the LLM must be:

- Reliable or accurate: when putting an LLM in the role of a tutor or evaluator, it is replacing an instructor (professor or teaching assistant). While not all instructors will give the same answer or approach when asked a question, we do assume that they will not generally lead a student astray. Likewise, we cannot have an LLM be a detriment to the educational process.
- Consistent (relatively): an educational experience should be relatively consistent. When asked the same question, we do not assume that an instructor's answer will be wildly divergent. Some variation to give alternative views is desirable, but not if it is so diverse that it confuses or misleads the student.

When acting as a teammate, it is much more acceptable, maybe even desirable, for the LLM to be somewhat obtuse. Forcing a student to "reason" with the teammate, converging on an agreed solution is part of the learning process.

4 USE CASES IN M&S EDUCATION

The authors have experimented with numerous real examples drawn from M&S courses focusing on discrete event simulation, continuous simulation, and simulation analysis. ChatGPT 3.5 was the primary LLM employed, with an eye on keeping student costs down. However, when examining potential future graphical capabilities, ChatGPT 4 was employed. The use cases are categorized as mathematical modeling, graphical modeling, and simulation.

4.1 Mathematical Modeling

The primary mathematical modeling considered was the use of ordinary differential equations (ODE). Others are considering the use of LLMs in engineering education (Tsai et al. 2023), and this work does not look to build on that. Rather we simply want to highlight that the learning paradigm may need to shift due to the availability of LLMs.

A simple example of how an LLM can quickly develop an ODE and provide an explanation of the equation is provided. Given the prompt "create ordinary differential equations for an RLC circuit", ChatGPT provides back a list of the pertinent variables (V_R, V_L, V_C, I, R, L, C, and t) and a description of each. It then derives the differential equation starting from Kirchhoff's voltage law (KVL) ($V_R + V_L + V_C = 0$). Given the relationships ($V_R = R \cdot I$), ($V_L = L \cdot dI/dt$), and ($V_L = 1/C \cdot \int I\, dt$), the final equation

$$R \cdot I + L \cdot \frac{dI}{dt} + \frac{1}{C} \cdot \int I\, dt = 0$$

is derived. The prompt can be followed up by requesting the equation be given with respect to the voltage and it walks through deriving the equation

$$V_R + L \cdot \frac{1}{R} \cdot \frac{dV_R}{dt} + \frac{1}{RC} \cdot \int V_R dt = 0$$

This likely requires a change in evaluating a student's understanding of the equation in the absence of an LLM. But it also suggests students will need to learn how to interact with the LLM, i.e., formal training in prompt engineering (Velásquez-Henao et al. 2023; Marvin et al. 2024; Cain 2024; Denny et al. 2024). The authors' university includes a required course on information literacy and it is hoped that this can be included in that course to ensure coverage for all students, not just MSE students.

4.2 Graphical

Graphical representations of systems are a common practice for model development. Originally the authors considered discrete event and continuous modeling to be separate discussions. However, it quickly became obvious that the issues in using an LLM were tightly related. It was also envisioned that LLMs would be weak in giving tutorial support for graphical models, especially if not using an LLM with graphical capabilities. On the contrary, ChatGPT 3.5 gave surprisingly useful results, though issues arose that require addressing.

4.2.1 Graphical Representation

Starting with the immediate question of the appropriateness of the graphical representation, the current status of LLMs is lacking. When requested to "create a bond graph for a mass spring damper system", ChatGPT 3.5's "graphical" representation and ChatGPT 4's are shown in Figure *2*. Surprisingly, 3.5's representation, while textually generated, is much more useful. 3.5 often has formatting issues requiring some work by the user to derive its intent. As an alternative, Google's Gemini clearly tries to search for an appropriate example. But this is not useful if the system being modeled is not a published model or the graphical model is not as common in literature such as event graphs (Schruben 1983). However, better than the graphical representations, the textual descriptions of the graphical models were found to be sufficient to construct the graphical representation. This is seen as a benefit in education as it still requires the student to understand the model at a graphical level in order to construct it from the textual description.

Considering an alternative, the generation of Petri Nets was also considered. The results were somewhat disturbing when creating a Petri Net for a single server, single queue system. In ChatGPT 3.5, the text based graphical representation was basically correct, but the text description of the model lacked a Service Complete place that was evident in the "graphic". Also, both lacked a transition allowing the server to serve another customer. Both were easily remedied by simple subsequent prompts, although adding the transition resulted in convoluted formatting of the graphic making it unreadable.

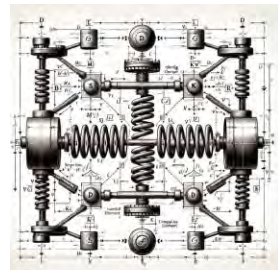
4.2.2 Prompt Engineering

When viewing the LLM as a teammate in developing a model, the LLM cannot be expected to generate the desired system on the first prompt. The authors' experience shows that ChatGPT will often create a "correct" model, but often simplified or lacking detail. To its credit, ChatGPT usually points this out, effectively prompting the student that they must refine their model. Event graph models are utilized to highlight how students would be required to interact with the LLM to get an "acceptable" model.

Event graphs have proven very useful in discrete-event simulation software development education. They provide a representation that can be directly mapped to a software implementation (Leathrum et al. 2017) and are very convenient for visualizing the execution of the software (Collins et al. 2017). Having heavily integrated them into our curriculum, they are a logical case study for use with LLMs. Two examples are used to demonstrate how students can interact with an LLM in the development of an event graph representation of a discrete system, a single-server, single-queue (SSSQ) and the dining philosophers problem.

```
Perl
     q (displacement)
      |
      |
      |                         0-junction
  ----|---- K ----|----|---- C ----|---- M ----
      |                  |
      |                  |
      q (velocity)       p (momentum)
```

ChatGPT 3.5 “graphical” representation



ChatGPT 4 graphical representation

Figure 2. ChatGPT graphical representations of the Bond Graph for a mass-spring-damper system.

4.2.2.1 Single Server, Single Queue Example

Figure *3* shows a sequence of prompts requesting ChatGPT to create and then modify an event graph model for the SSSQ example. Only a textual description of the event graph is provided including a list of the nodes, their purpose, how they modify state variables, and their transitions. But it is in sufficient detail that given a rudimentary understanding of event graphs, it is trivial to construct the graphic. Figure *3* has taken that description and formatted it to include the graphic so it easier for the reader.

The first event graph generated is not what is generally seen in the literature. It is also lacks identifying delays in the system, though ChatGPT clearly states that they are missing, and the user may want to add them. The subsequent prompt performs event reduction, combining the Arrival event and the Queue Check event into a single event, a more common representation. The third prompt fixes the lack of a service time which ChatGPT simply introduced the delay time and indicated on which transition it occurs. The last prompt addresses that the event graph does not have the ability to self-generate customers. The literature usually addresses this by having the arrival event schedule the next arrival on itself. However, this prevents combining SSSQs into a larger system, so the prompt requests the introduction of a new Source event that sends the customer to the arrival and schedules the creation of the next customer with an interarrival time delay. Note that the prompt gives little information on how the user wants the Source event integrated, but ChatGPT correctly connected the Source event to the Arrival event.

4.2.2.2 Dining Philosophers Problem

The dining philosophers problem is included as it highlights a different aspect of prompt engineering. The model was started with a simple prompt “create an event graph for the dining philosophers problem”. The result was mostly correct but lacked the intent of the problem to study the potential for deadlock or starvation and some necessary details.

The dining philosophers problem (Hoare 1978) involves *n* philosophers sitting around a table, each with a bowl of (in our case) lo mein in front of them. Between each philosopher is a single chopstick. Each philosopher starts thinking, but when he gets hungry, he tries to pick up the two chopsticks on either side of him, and if successful, he can eat. On completion of eating, he puts down the chopsticks, potentially enabling his neighboring philosophers to get them so they can eat. The process of obtaining the chopsticks allows studying approaches to resource contention.

The original model generated involved a philosopher attempting to pick up both chopsticks on completion of eating, part of an Attempt event to get the chopsticks. If successful, the philosopher could eat, if not he just waits. The problem is more interesting if the philosopher only picks up one chopstick at a time, thus creating the potential for deadlock as each philosopher may have 1 chopstick and waits on the other chopstick. In addition, the solution lacks clear indication how a philosopher gets a second attempt to eat should he fail the first time.

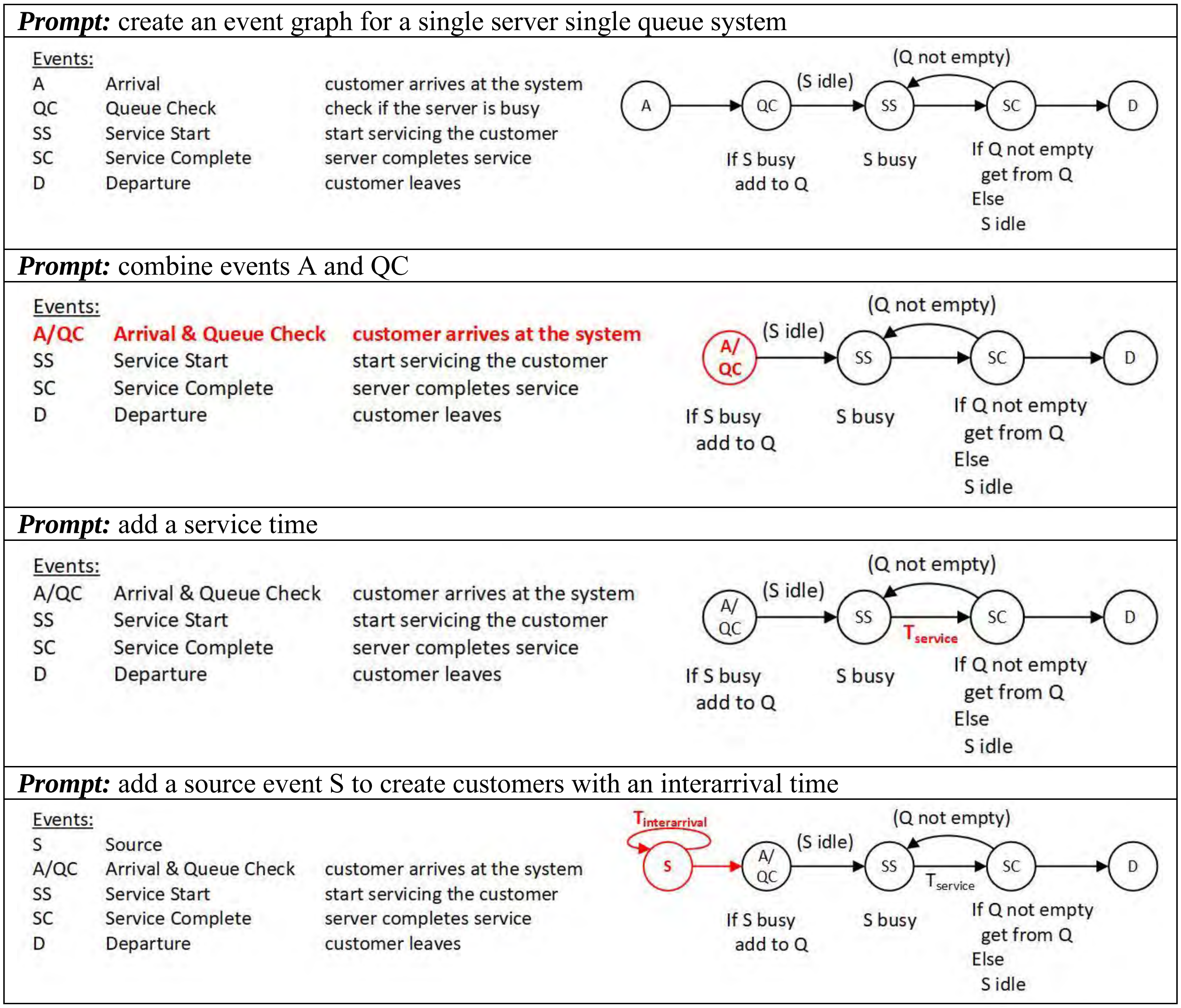


Figure 3. ChatGPT prompts for development of an event graph for a single server, single queue system.

The first update to the model is to request the addition of delay times for thinking and eating. Then adding the prompt "make separate events for acquiring each chopstick" easily partitions the Attempt event into an Attempt Left and Attempt Right event. Then a prompt is required to add the logic for a philosopher to schedule a neighbor to pick up a chopstick if they are waiting for it when it is put down. This latter prompt is more involved as it requires the model to recognize the concept of a neighbor.

4.2.3 Consistency

While the above results were promising, later attempts to repeat the process in new sessions was disturbing. The event graph example is highlighted as showing the most inconsistency, likely due to less material for the LLM to base its reasoning on. On trying to repeat the above SSSQ example, ChatGPT gave a totally different event graph shown in Figure 4. While not "wrong", it is not the common representation of an

SSSQ provided in literature. This can be a good thing, but it does demonstrate a divergence of development. However, it does provide the opportunity to show students how they can interact with the system, requiring

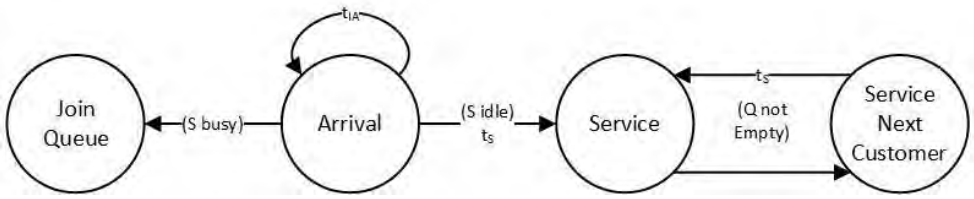


Figure 4. Alternate event graph generated by a second ChatGPT session.

them to understand the model and massage it to their needs.

But worse is that at an even later date, the system provided results that did not even show an understanding of event graphs. This resulted in one author getting in an argument with ChatGPT where ChatGPT in a passive aggressive manner kept agreeing that it made a mistake but refusing to fix it. This behavior was observed for the dining philosophers as well. The fact that the interaction got worse over time is disturbing. It also highlights the need for some control over the process in the initial stages of teaching students, a topic to be discussed later in the paper.

4.3 Simulation

The benefits (and some pitfalls) of utilizing LLMs in MSE education of model development have been highlighted. The next step is the development of a simulation. This is seen as falling in two categories, utilizing a tool or developing software. The latter is left as a separate topic as much literature is available on software generation and debugging using LLMs. But the use of tools was studied. Two tools were considered, Simulink (MathWorks 2024) for continuous simulation and Arena (Rockwell Automation 2024) for discrete event simulation, with very similar results.

When requesting the LLM to assist in developing a simulation in either Simulink or Arena, detailed step by step instructions were provided. When provided the prompt “create a Simulink model for a mass spring damper system”, the instructions, shown in Figure were provided. The Simulink example also presented a graphical representation (Figure 6) as reference which the Arena did not. The examples demonstrate that the LLM can be utilized to assist training students in the use of the tools to build a simulation. However, the Simulink example also demonstrates the need for clear prompts as in our M&S curriculum, we would expect the simulation to be built to represent state-space model equations using, e.g., Addition, Gain and Integrator blocks rather than Domain (e.g., Mechanical) blocks. Using the LLM requires clear expectations stated to the students and their understanding of prompt engineering.

In addition, little insight is gained by using the LLM. The tools already have detailed tutorials on the building of a simulation. The insight is in the model development. It is likely that the tools will consider inclusion of LLM technology in their help systems, and it is worth waiting till that time rather than putting effort into creating a useful LLM-based tool interaction.

5 LLM MATURITY

Large language models (LLMs) are evolving rapidly, marked by significant advancements in their capabilities, efficiency, and applications. This evolution reflects both the increasing sophistication of machine learning algorithms and the growing availability of vast datasets for training these models. Therefore, it is important to run the same task on LLMs multiple times over time to monitor and ensure the quality, reliability, and evolution of LLMs over time. In addition, the same task should be run on different LLMs to compare their capabilities, strengths, and weaknesses. There are several reasons and benefits for conducting such comparative analyses such as model benchmarking, application suitability, and robustness and reliability, among others. This section will use two popular LLMs, namely OpenAI’s ChatGPT and

Google's Gemini, to demonstrate LLMs' evolution over time and provide an anecdotal comparison of their capabilities in processing tasks that are related to M&S.

1. **Open Simulink:** Open MATLAB and go to the Simulink library browser.
2. **Create a New Model:** Create a new Simulink model by clicking on "File" > "New" > "Model" or using the new_system command.
3. **Add Blocks:** Drag and drop the following blocks from the Simulink library browser into your model:
   - **Sources:**
     - Add a "Step" block (Sources > Step) to represent an external force or input.
     - Add a "Scope" block (Sinks > Scope) to visualize the system response.
   - **System Components:**
     - Add a "Mass" block (Simscape > Foundation > Mechanical > Mechanical Elements > Mass) to represent the mass.
     - Add a "Spring" block (Simscape > Foundation > Mechanical > Mechanical Elements > Spring) to represent the spring.
     - Add a "Damper" block (Simscape > Foundation > Mechanical > Mechanical Elements > Damper) to represent the damper.
4. **Connect Blocks:** Connect the blocks appropriately using lines. For example:
   - Connect the output of the Step block to the input of the Mass block.
   - Connect the output of the Mass block to the input of the Spring block.
   - Connect the output of the Spring block to the input of the Damper block.
   - Connect the output of the Damper block back to the input of the Mass block.
5. **Parameterize Blocks:** Double-click on each block to set the parameters such as mass, spring constant, damping coefficient, and initial conditions.
6. **Configure Solver Settings:** Go to "Model Configuration Parameters" (Ctrl + E), navigate to the "Solver" pane, and set the solver type and options.

Run Simulation: Click the "Run" button to simulate the system. You can observe the response using the Scope block.

Figure 5. ChatGPT session for "create a Simulink simulation for a mass-spring-damper system".

```
rust
Step --> Mass --> Spring --> Damper --> Mass (Feedback loop)
                    |
                    V
                  Scope
```

Figure 6. ChatGPT 3.5 "graphical" representation of the system in Figure 5.

For comparison, both ChatGPT and Gemini were asked to "Use GLUT and OpenGL to calculate and draw the probability density histogram from an input file."

- ChatGPT generated several functions, such as readDataFromFile() and calculateHistogram(), that were called only once in the main() function. The results of these functions were used in a display() function. ChatGPT was able to generate highly efficient, if not optimal, code.
- The code generated by Gemini exhibits serious problems. The file reading and computation of maximum and minimum values of the input data are included in the display() function, which is usually called at the frame rate, e.g., 60 Hz. As the input file does not change every frame, the file

reading and computation of the data range should be kept out of the display() function and should only be called when there is a new input file.

To compare the LLMs' high-level software architecture design capabilities, both ChatGPT and Gemini were asked to "suggest the classes using MonoGame to implement a Super Mario game." MonoGame is an open-source game development framework that's widely used for creating cross-platform games. It is an evolution of Microsoft's XNA framework, designed to make game development more accessible while still offering the power and flexibility needed for advanced game development. The classes suggested by ChatGPT and Gemini are shown in Table 1.

Table 1. Classes suggested by ChatGPT and Gemini for a Super Mario game.

Classes suggested by ChatGPT	Classes suggested by Gemini
• Game1 • ScreenManager • GameScreen • Level • Player • Enemy Abstract class • Specific Enemy classes (Goomba, Koopa, etc.) • Block • Item Abstract class • Specific Item classes (Mushroom, FireFlower) • Camera • HUD • Physics • Animation • ContentManager	• Game1 • Mario • Platform • Enemy • Item • Level • Camera • AnimationManager • tileMap • SoundManager • HUD

It is seen that ChatGPT was able to generate a more sophisticated and elegant design by utilizing object-oriented programming and suggested more substantive classes, e.g., Enemy abstract class and its derived classes (Goomba, Koopa, etc.) and Item abstract class and it derived classes (Mushroom and FireFlower). More importantly, ChatGPT also suggested a Physics class that is critical for developing physically realistic games. This critical class is missing in the results generated by Gemini.

Although ChatGPT produced better results for the two examples discussed in this section, it is important to note that all LLMs are rapidly evolving, and they should be constantly evaluated in terms of their capabilities for various applications and for the overall advancement of the LLMs themselves.

6 THE PATH FORWARD

Given the issues and experiences we have outlined with LLMs, they are clearly too general and may lead to confusion if not used properly. In addition to their proper use, however, our aim of incorporating LLM's into the M&S instruction is quite specific due to the nature of the M&S problems and solutions. Hence, there is a need to fine-tune LLMs (Ferrer 2024) for this type of instruction and make reliable assistants (tutors and teammates) as described in Section 3. Following the stages of fine-tuning LLMs in other scientific and engineering domains (see, e.g., (Jeong 2024)), we envision that the most productive path forward is to fine-tune a LLM locally by considering a carefully designed subset of model parameters that will encompass the M&S instruction domain. This is an achievable goal for us as will be outlined in the next subsections. Moreover, its successful outcomes will provide multiple benefits beyond our home institution, since we plan to develop general solutions for the M&S instruction using LLM.

6.1 Availability of local computational resources

Our home institution features excellent high-performance computing (HPC) resources to accomplish LLM fine-tuning locally. Specifically, the HPC cluster, called *Wahab* (2024) is a reconfigurable computing platform based on OpenStack architecture to support diverse types of computational research workloads. The Wahab cluster consists of 158 compute nodes and 6320 computational cores using the state-of-the-art Intel's "Skylake" Xeon Gold 6148 processors (20 CPU cores per chip; 40 cores per node). Each compute node has 384 GB of RAM. To accommodate the rising computing demand from machine learning and artificial intelligence workloads, there are 18 accelerator compute nodes, each of which is equipped with four Nvidia V100 graphical processing units (GPU), for a total of 72 GPUs. A 100 Gbps EDR Infiniband high-speed interconnect provides low-latency, high-bandwidth communication between nodes to support massively parallel computing as well as data-intensive workloads. Wahab is equipped with a dedicated high-performance Lustre scratch storage (350 TB usable capacity) and is connected to the 1.9 PB university-wide home/long-term research data networked file system. Wahab also has 45 TB of CEPH block storage that can be provisioned for user data in the virtual environment. The compute nodes in Wahab cluster are provisioned as two types of resources: (1) Traditional HPC environment, using the industry standard SLURM job scheduler with access to high-speed Lustre scratch storage. (2) Virtual environment**,** which greatly expands access to high-performance computing resources for users that need customized computing environment, such as custom virtual machines or custom compute–network environments. We estimate that both types of resources will be utilized in our local LLM investigations: While the latter may reproduce easier the native LLM installation and runtime, the former is more computationally efficient for LLM training phase. We plan to expose to student the virtual environment, in the form of On-demand computing and run the training tasks as batch jobs in the queuing system of Wahab.

Recently, even newer GPUs, NDIDIA A100's, were configured into Wahab cluster as 7 nodes each with an AMD 7543 processor having 64 cores at 2.8 GHz and 2 A100 GPUs. These are high-memory GPUs (80 GB), and therefore, capable of handling large datasets for LLM training.

6.2 Availability of Open-Source LLMs and fine-tuning

There already exists a number of generative models, some of them available as open-source (see, e.g., Table 1 in (Jeong 2024)). In particular, the LLaMA2 model from META is commonly used for fine-tuning (Munoz 2023). We will install LLaMA2 locally and adapt it to the Wahab environment with the help of the ODU Research and Cloud Computing Group. The dataset for tuning will consist of specific prompts and corresponding answers for the variety of models, their graphical representations and simulation parameters and result used in our MSE programs.

Despite the excellent local computing resources, the LLMs trained on very large data sets will not fit even into the high-memory GPUs available. Hence, we will consider the fine-tuning techniques designed for efficient computation, such as Parameter Efficient Fine-Tuning (PEFT) and Low-Rank Adaptation (LoRa). The dataset for tuning and the fine-tuned model will be made publicly available, which will benefit a larger M&S educational community.

7 CONCLUSIONS

This paper captures the authors' initial efforts to assess the ability to integrate LLMs into the education of modeling and simulation engineering professionals. Our conclusions range widely, identifying the good and the bad, but with a promise of proceeding forward as technology is constantly progressing. And the Department of Education has similar concerns (Cardona et al. 2023), there is hope that the concerns will be addressed.

The potential benefits of LLMs being included as a partner in the educational process can be found in the example use cases provided. When the LLM starts on track, the experience matches the thought process involved in design, starting with an initial idea and iterating to a solution through continued modifications. ChatGPT demonstrates high insight into models and responds appropriately to requested modifications.

While ChatGPT 3.5 does not provide graphical capabilities, it does an admirable job of providing textual descriptions of the graphical representations and occasionally even an acceptable textual representation of the graphic. ChatGPT 4 does not seem to improve on the graphical representation, tending to generate very confusing graphics, but at the current time only 3.5 is being considered to keep student expenses low. The authors see great potential in an LLM participating as a tutor or teammate in the education process.

On the other hand, LLMs can lead students astray. If they do not start from a solid understanding, they only create confusion. Identifying that starting point can be difficult, requiring significant prompt engineering. This creates an additional educational component for the student, i.e., the student must first learn how to properly use the LLM. Even then, the instructor does not necessarily have the necessary control to ensure quality education. When selecting a textbook, the instructor has total control, but an LLM is constantly adapting and changing. The Department of Education identified the need for explainability in AI and the ability to override AI when integrated into education. The authors have proposed their way forward, creating a local LLM where some level of control can be maintained, but with significant overhead, both computationally and administratively. However, the issues with the use of LLMs in modeling and simulation are not seen as roadblocks as the benefits are great and the technology is rapidly changing, hopefully for the best.

REFERENCES

Cain, W. 2024. "Prompting Change: Exploring Prompt Engineering in Large Language Model AI and Its Potential to Transform Education". *TechTrends*, 68: 47-57.

Cardona, M., R. Rodríguez, and K. Ishmael. 2023. "Artificial Intelligence and the Future of Teaching and Learning: Insights and Recommendations". US Department of Education Office of Educational Technology.

Chung, P., M. Boodoo, and S. Doboli. 2023. "Case Scenario Generators for Trauma Surgery Simulation Utilizing Autoregressive Language Models". *Artificial Intelligence in Medicine*, 144.

Collins, S., N. Gonda, L. Dumalian, J. Leathrum, and R. Mielke. 2017. "Visualization of Event Execution in a Discrete Event System". In *50th Annual Simulation Symposium (ANSS 2017)*, April 23rd-26th, Virginia Beach, VA, Article 6, 1-12.

Dempere, J., K. Modugu, A. Hesham, and L. K. Ramasamy. 2023. "The Impact of ChatGPT on Higher Education". *Frontiers in Education*, 8: 1-13.

Deng, Y., J. Yao, Z. Tu, X. Zheng, M. Zhang, and T. Zhang. 2023. "TARGET: Traffic Rule-based Test Generation for Autonomous Driving Systems". *ArXiv abs/2305.06018*.

Denny, P., J. Prather, B. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, et al. 2024. "Computing Education in the Era of Generative AI". *Communications of the ACM*, 67(2): 56-67.

Edwards, C. 2024. "Teaching Transformed". *Communications of the ACM*, 67(2): 12-13.

Essel, H., D. Vlachopoulos, A. Tachie-Menson, E. Johnson, and P. Baah. 2022. " The Impact of a Virtual Teaching Assistant (Chatbot) on Students' Learning in Ghanaian Higher Education". *International Journal of Educational Technology in Higher Education*, 19: 1-19.

Ferrer, J. 2024. "An Introductory Guide to Fine-Tuning LLMs", Datacamp. https://www.datacamp.com/tutorial/fine-tuning-large-language-models.

Hoare, C. A. R. 1978. "Communicating Sequential Processes". *Communications of the ACM*, 21(8): 666-77.

Jeong, C. 2024. "Fine-tuning and Utilization Methods of Domain-Specific LLMs". https://doi.org/10.48550/arXiv.2401.02981.

Khosravi, H., S. Shum, G. Chen, C. Conati, Y.-S. Tsai, J. Kay, et al. 2022. "Explainable Artificial Intelligence in Education". *Computers and Education: Artificial Intelligence*, 3.

Leathrum, J., R. Mielke, A. Collins, and M. Audette. 2017. "Proposed Unified Discrete Event Simulation Roadmap for M&S Curricula". In *Proceedings of the 2017 Winter Simulation Conference*, 4300-11 https://doi.org/10.1109/WSC.2017.8248135.

Li, Y., Y. Zhang, and L. Sun. 2023. "Metaagents: Simulating Interactions of Human Behaviors for LLM-Based Task-Oriented Coordination Via Collaborative Generative Agents". *arXiv preprint arXiv:2310.06500*.

Marvin, G., N. Hellen, D. Jjingo, and J. Nakatumba-Nabende. 2024. "Prompt Engineering in Large Language Models". In *Proceedings of International Conference on Data Intelligence and Cognitive Informatics (ICDICI 2023)*, June 27th-28th, Singapore, 387-402.

MathWorks. 2024. "Simulink", Accessed 1st April. https://www.mathworks.com/help/simulink/.

Munoz, E. 2023. "Fine-Tuning a Llama-2 7B Model for Python Code Generation", Towards AI. https://towardsai.net/p/data-science/fine-tuning-a-llama-2-7b-model-for-python-code-generation.

ODU Research Cloud Computing. 2024. "Wahab: The 2nd Gen ODU HPC Cluster", Accessed 1st April. https://wiki.hpc.odu.edu/en/Cluster/Wahab.

Rockwell Automation. 2024. "Arena Simulation Software", Accessed 1st April. https://info.arenasimulation.com/v16-esd-0.

Schruben, L. 1983. "Simulation Modeling with Event Graphs". *Communications of the ACM*, 26(11): 957-63.

Shorey, S., E. Ang, J. Yap, E. Ng, S. Lau, and C. Chui. 2019. "A Virtual Counseling Application Using Artificial Intelligence for Communication Skills Training in Nursing Education: Development Study". *Journal of Medical Internet Research*, 21(10): e14658.

Tan, S., B. Ivanovic, X. Weng, M. Pavone, and P. Krahenbuhl. 2023. "Language Conditioned Traffic Generation". In *7th Conference on Robotic Learning (CoRL 2023)*, Nov 6-9, 2023, Atlanta, GA, 2714-52.

Tegos, S., S. Demetriadis, and A. Karakostas. 2015. "Promoting Academically Productive Talk with Conversational Agent Interventions in Collaborative Learning Settings". *Computer Education*, 87: 309-25.

Tsai, M.-L., C. W. Ong, and C.-L. Chen. 2023. "Exploring the Use of Large Language Models (LLMs) in Chemical Engineering Education: Building Core Course Problem Models with Chat-GPT". *Education for Chemical Engineers*, 44: 71-94.

Updyke, D., T. Podnar, and S. Huff. 2023. "Simulating Realistic Human Activity Using Large Language Model Directives". Carnegie Mellon University Software Engineering Institute.

Velásquez-Henao, J. D., C. J. Franco-Cardona, and L. Cadavid-Higuita. 2023. "Prompt Engineering: A Methodology for Optimizing Interactions with AI-Language Models in the Field of Engineering". *DYNA*, 90(230): 9-17.

Xie, C., C. Chen, F. Jia, Z. Ye, K. Shu, A. Bibi, et al. 2024. "Can Large Language Model Agents Simulate Human Trust Behaviors?". *arXiv:2402.04559v2*

Zhang, J., Y. Zhang, M. Chu, S. Yang, and T. Zu. 2023. "A LLM-Based Simulation Scenario Aided Generation Method". In *7th Information Technology and Mechatronics Engineering Conference (ITOEC)* September 15th-17th, Chongqing, China, 1350-54.

AUTHOR BIOGRAPHIES

JAMES F. LEATHRUM, JR. is an Associate Professor in the Department of Electrical and Computer Engineering at Old Dominion University. He earned the Ph.D. in Electrical Engineering from Duke University. His research interests include simulation software design, distributed simulation, simulation-based test & evaluation, and simulation education. His email address is jleathru@odu.edu.

YUZHONG SHEN is a Professor and Associate Chair in the Department of Electrical and Computer Engineering at Old Dominion University. He earned the Ph.D. in Electrical Engineering from the University of Delaware. His research interests include virtual reality, augmented reality, visualization and computer graphics, transportation modeling and simulation, general modeling and simulation and signal and image processing. His e-mail address is yshen@odu.edu.

MASHA SOSONKINA is a Professor in the Department of Electrical and Computer Engineering at Old Dominion University. She holds a Ph.D. in Computer Science from Virginia Polytechnic Institute and State University. Her research interests include high-performance simulations and applications, energy-efficient computing systems, computational science and engineering, and data-science infrastructure. Her email address is msosonki@odu.edu.